CNPEM
Centro Nacional de Pesquisa em Energia e Materiais

TRAD2026

TREINAMENTO EM REPRODUTIBILIDADE PARA ANÁLISE DE DADOS

Módulo III: NextFlow

Apostila

LNBio

lnbr

Sumário

1. Introdução 3
1.1. O TRAD2026 em três módulos 4
2. O que é um gerenciador de workflows? 4
2.1. Processo, tarefa e canal 4
2.2. O diretório de trabalho e o cache 5
3. Onde a análise vai rodar: a arquitetura do Marvin 6
3.1. Nó de login e nó de processamento 6
3.2. /shared e /scratch: o que fica onde 6
4. Preparando a sessão no Marvin 7
4.1. Acesso e módulo 7
4.2. As variáveis de ambiente 7
4.3. A árvore de diretórios da execução 8
4.4. O teste que confirma o ambiente 8
5. O pipeline nf-core/ampliseq 9
6. As três entradas do pipeline 10
6.1. A folha de amostras 10
6.2. Os metadados 11
6.3. O arquivo de parâmetros 12
7. Anatomia do comando 12
8. Os perfis de configuração 13
9. O script de submissão SLURM 14
9.1. O cabeçalho SBATCH 14
9.2. O script completo 14
9.3. Submeter e acompanhar 16
10. Diagnóstico de falhas e retomada 16
11. Interpretação dos resultados 17
12. Erros comuns 19
13. Boas práticas 19
14. Os dois roteiros práticos 20
15. Referências e leituras recomendadas 21
Anexo A — Tabela de configurações do módulo 21
Anexo B — Cartão de referência rápida 22

1. Introdução

Uma análise de dados ômicos raramente é um único programa. Ela é uma sequencia de ferramentas, cada uma com sua versão, seus parâmetros, suas dependências e requisitos de recursos computacionais.

Quando essa sequência é conduzida por scripts encadeados sem gerenciamento de estado, o resultado passa a depender de decisões que não ficam registradas.

O **Nextflow** é um sistema de gerenciamento de workflows científicos que descreve a análise como um fluxo de dados entre processos isolados. Ele separa a lógica científica do ambiente de execução, o que permite que o mesmo workflow seja executado em um laptop, em um cluster de alto desempenho ou em nuvem, com resultados equivalentes.

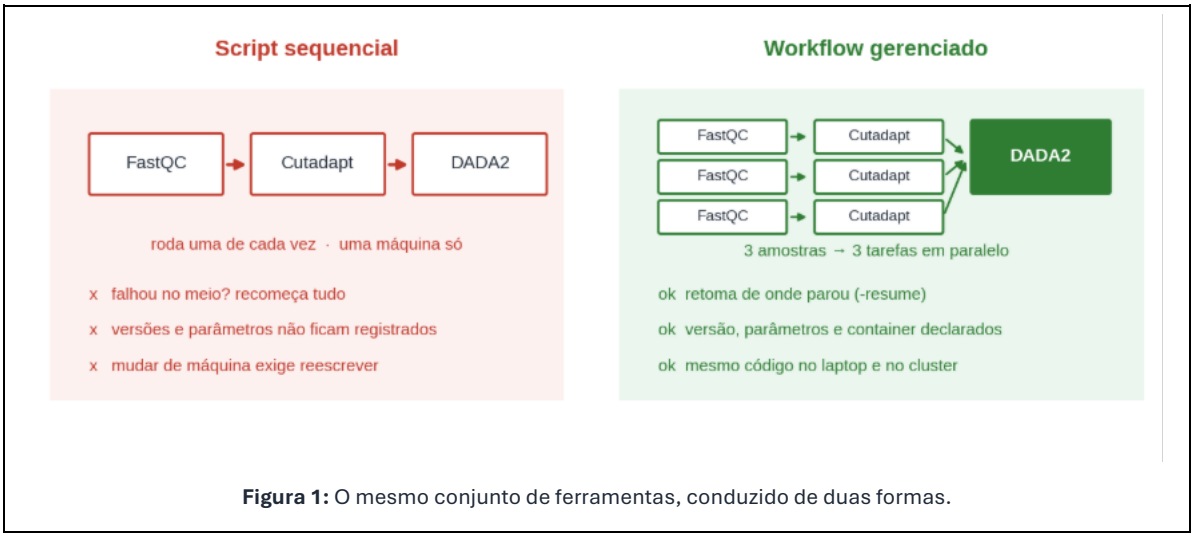

Figura 1: O mesmo conjunto de ferramentas, conduzido de duas formas.

Neste módulo, os conceitos serão aplicados ao pipeline **nf-core/ampliseq**, destinado à análise de sequenciamento de amplicons. A escolha se justifica por ser um pipeline modular e com resultados biologicamente robustos, que permite discutir tanto a mecânica da execução quanto o significado científico do que é produzido.

Nota.

O repositório do treinamento, as branches por grupo e o fluxo de contribuição por *Pull Request* permanecem os mesmos do Módulo I e serão utilizados para registrar as respostas das atividades deste módulo.

1.1. O TRAD2026 em três módulos

Reprodutibilidade computacional depende de controlar quatro elementos: **os dados, o código, o ambiente de execução e a própria execução**. O **Módulo I** tratou do código, por meio do controle de versão. O **Módulo II** tratou do ambiente, por meio de containers. O **Módulo III** — Nextflow— trata da orquestr, registrá-la e retomá-la.

> **Nota.**
>
> Toda a execução deste módulo ocorre **no cluster Marvin**, via SLURM, com containers Singularity. Não há instalação a fazer no seu computador.

2. O que é um gerenciador de workflows?

Um gerenciador de workflows é responsável por decidir **o que executar, quando executar e onde executar**, a partir de uma descrição declarativa da análise. O pesquisador declara as etapas e as dependências entre elas; o gerenciador resolve a ordem, o paralelismo, o agendamento e a recuperação de falhas.

Essa separação traz três propriedades que interessam diretamente à reprodutibilidade:

- **Reprodutibilidade:** a versão do workflow, dos parâmetros e dos containers é declarada explicitamente e pode ser recuperada meses depois.
- **Portabilidade:** o mesmo código é executado em diferentes infraestruturas, alterando apenas o perfil de configuração.
- **Escalabilidade:** o paralelismo é derivado do fluxo de dados, sem que o pesquisador precise escrever lógica de paralelização.

2.1. Processo, tarefa e canal

O Nextflow organiza a análise em **processos**, que são unidades isoladas de execução, conectados por **canais**, que transportam os dados entre eles. Um processo não chama outro

processo diretamente: ele apenas consome o que chega em seu canal de entrada e emite o resultado em seu canal de saída.

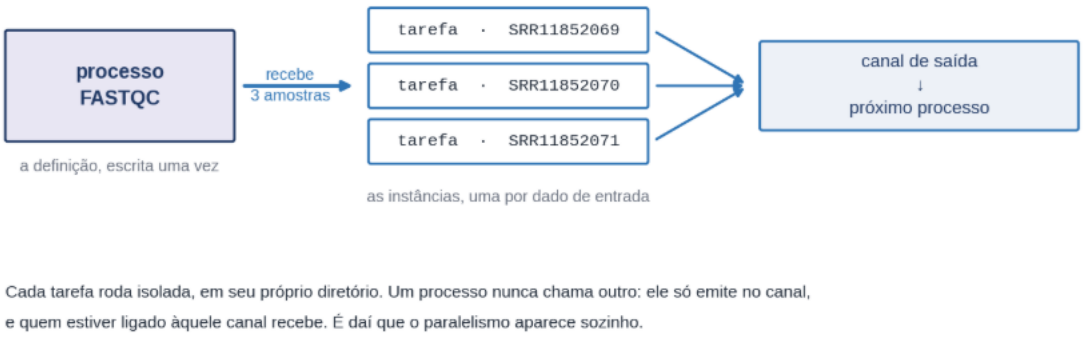

Figura 2: Processos, canais e o paralelismo derivado do fluxo de dados.

Nota.

Cada execução de um processo sobre um dado específico é chamada de **tarefa** (*task*). O processo é a definição; a tarefa é a instância. Essa distinção é importante ao interpretar os relatórios de execução e as mensagens de erro.

2.2. O diretório de trabalho e o cache

Cada tarefa é executada em um diretório próprio e isolado, criado dentro do diretório de trabalho (work/) e identificado por um *hash* como a1/3f9c2b. Esse diretório contém os arquivos de entrada preparados, os arquivos de saída e os registros da execução.

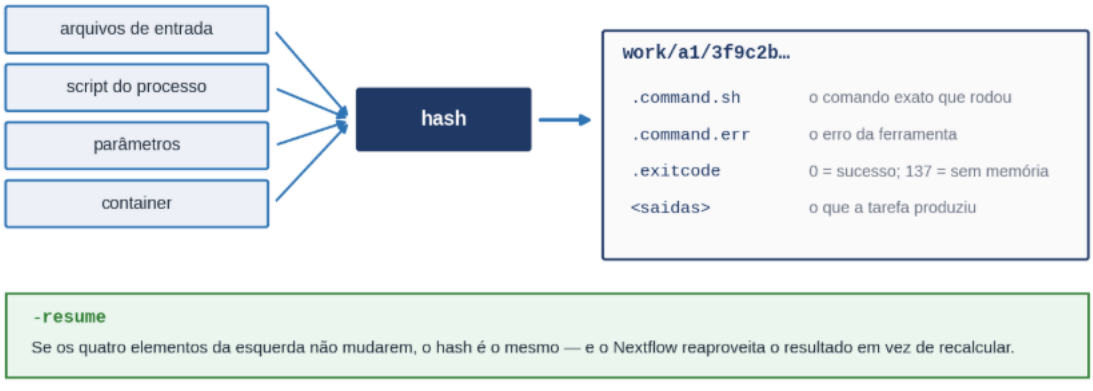

Figura 3: De onde vem o identificador do diretório e o que há dentro dele.

Atenção.

O diretório work/ cresce rapidamente e contém arquivos intermediários que não devem ser versionados nem confundidos com os resultados finais. Os resultados publicados ficam no diretório indicado por --outdir.

3. Onde a análise vai rodar: a arquitetura do Marvin

Todo o treinamento é executado no **cluster Marvin**. O Java, o Nextflow e as ferramentas científicas já estão disponíveis no ambiente do cluster — as últimas na forma de containers Singularity pré-validados. Entender como o cluster está organizado explica quase todas as decisões de configuração que virão a seguir.

3.1. Nó de login e nó de processamento

```
$ ssh usuario@marvin.cnpem.br
# no de login: preparar arquivos e submeter
$ ssh dgpu[01-03]
# no de execucao da pratica

$ module avail
$ module load nextflow/26.04.4
$ nextflow -version
    N E X T F L O W
    version 26.04.4
```

Atenção.

Nunca execute o pipeline no nó de login. Ele é compartilhado por toda a instituição; uma execução pesada ali prejudica todos os usuários. O próprio orquestrador do Nextflow também deve ir para a fila, via **sbatch**.

3.2. /shared e /scratch: o que fica onde

O Marvin separa o que é **compartilhado e estável** do que é **local e volátil**.

Caminho	Natureza	Conteúdo do treinamento
/shared/training/TRAD2026	compartilhado	dados brutos, pipeline e containers
/shared/.../pipelines/nf-core-ampliseq-2.18.0	cópia local	dispensa o download do GitHub
/shared/.../containers/nextflow/amplicon	imagens Singularity	conjunto **amplicon**, não ampliseq

Caminho	Natureza	Conteúdo do treinamento
/scratch/$USER/ampliseq	**disco local do nó**, rápido	**runs/**, **work/**, **logs/**, **metrics/**

Nota.

O pipeline roda a partir de uma **cópia local já validada** em **/shared/training/TRAD2026/pipelines/nf-core-ampliseq-2.18.0**. Isso dispensa o download do GitHub a cada execução e, com ele, a necessidade de token — o que elimina o erro de limite de requisições (**API rate limit exceeded**) que costuma travar turmas inteiras.

Atenção.

Como o **/scratch** é **disco local do nó**, o job precisa ser fixado em um nó específico (**#SBATCH --nodelist=dgpu01**). Se cair em outro nó, o cache do **-resume** não estará lá e tudo será reexecutado.

4. Preparando a sessão no Marvin

4.1. Acesso e módulo

```
$ ssh usuario@marvin.cnpem.br
# nó de login: preparar arquivos e submeter
$ ssh dgpu[01-03]
# nó de execução da pratica


$ module avail
$ module load nextflow/26.04.4
$ nextflow -version
    N E X T F L O W
    version 26.04.4
```

4.2. As variáveis de ambiente

Todos os comandos das atividades dependem destas variáveis. O script de submissão redefine as mesmas variáveis por conta própria — isso é intencional, para que o job seja autocontido.

```
$ export SHARED=/shared/training/TRAD2026
$ export SCRATCH=/scratch/$USER
$ export DATASET=Waterwaste


# containers: conjunto 'amplicon', nao 'ampliseq'
$ export NXF_SINGULARITY_CACHEDIR="$SHARED/containers/nextflow/amplicon"


$ export NXF_WORK="$SCRATCH/ampliseq/work/$DATASET"
```

```
$ export RUN="$SCRATCH/ampliseq/runs/$DATASET"
$ export LOGS="$SCRATCH/ampliseq/logs/$DATASET"
$ export METRICS="$SCRATCH/ampliseq/metrics/$DATASET"


$ mkdir -p "$RUN" "$LOGS" "$METRICS" "$NXF_WORK"
$ cd "$RUN"
$ pwd
/scratch/<usuario>/ampliseq/runs/Waterwaste
```

Atenção.

As variáveis valem **apenas nesta sessão do terminal**. Se a conexão cair, refaça esta etapa.

Crie o diretório de **logs antes do `sbatch`**: o SLURM abre os arquivos de **--output** e **--error** no instante da submissão, e um **mkdir** dentro do script executa tarde demais — o job falha sem deixar registro.

4.3. A árvore de diretórios da execução

```
/scratch/$USER/
 ├- nfx-home/
 └── ampliseq/
   ├──runs/
   │   └── Capybara/
   │       ├samplesheet.csv          # arquivo descrevendo o caminho absoluto do raw data
   │       ├ metadata.tsv             # arquivo descrevendo as condições do desenho experimental
   │       ├ Capybara_grupoX.json     # arquivo descrevendo a informação dos sequenciamentos
   │       └── results/
   ├── work/
   ├──logs/
   └── metrics/
```

Cada pasta tem um papel, e a separação entre **$RUN** e **$NXF_WORK** é o que permite **retomar sem refazer**.

4.4. O teste que confirma o ambiente

Todo pipeline nf-core tem um perfil **test**, com dados mínimos embutidos. Execute-o antes de qualquer coisa — **em um nó de processamento**, nunca no login.

```
$ nextflow run "$SHARED/pipelines/nf-core-ampliseq-2.18.0" \
    -profile test,singularity --outdir teste
executor >  local (12)
[a1/3f9c2b] NFCORE_AMPLISEQ:AMPLISEQ:FASTQC    [100%] 3 of 3
[7d/22e881] NFCORE_AMPLISEQ:AMPLISEQ:CUTADAPT  [100%] 3 of 3
-[nf-core/ampliseq] Pipeline completed successfully-
```

Dica.

Repare no **executor > local**: o Nextflow está usando os núcleos do próprio nó. Quando o job é submetido via SLURM com os perfis do treinamento, essa mesma linha dirá **slurm**. É o único sinal visível de que a orquestração mudou de camada.

5. O pipeline nf-core/ampliseq

O **nf-core** é uma iniciativa comunitária que reúne pipelines escritos em Nextflow segundo diretrizes comuns de estrutura, documentação, testes automatizados e versionamento. Adotar um pipeline nf-core em vez de escrever um do zero transfere para a comunidade a manutenção das dependências e permite que o esforço do pesquisador se concentre no desenho experimental e na interpretação.

O ampliseq responde a uma pergunta de enunciado simples e execução cara: **quem está presente nesta amostra e em que proporção**. Utiliza **DADA2** para a etapa de *denoising* e inferência de ASVs (*Amplicon Sequence Variants*) e **QIIME2** para as análises de comunidade.

Etapa	Ferramenta	O que produz
Qualidade	FastQC / MultiQC	relatório por amostra
Remoção de primers	Cutadapt	fração de leituras retidas
Denoising	DADA2	ASVs e tabela de contagem
Taxonomia	SBDI-GTDB / VSEARCH	classificação das ASVs
Comunidade	QIIME2	diversidade, barplot, ANCOM
Consolidação	summary_report	relatório único em HTML

As atividades utilizam a versão **2.18.0**, publicada em junho de 2026. Duas mudanças afetam a comparação com análises anteriores: a taxonomia padrão passou de SILVA 138.2 para **SBDI-GTDB R11-RS232-1**, e o método ANCOM-BC2 foi incorporado às análises de abundância diferencial.

Atenção.

A mudança da base de taxonomia padrão altera os resultados de classificação. Ao comparar uma análise nova com uma anterior, verifique se ambas utilizaram a mesma base de referência antes de atribuir a diferença a um efeito biológico.

6. As três entradas do pipeline

O ampliseq recebe três artefatos com papéis distintos. Entender essa divisão evita a maior parte dos erros de execução.

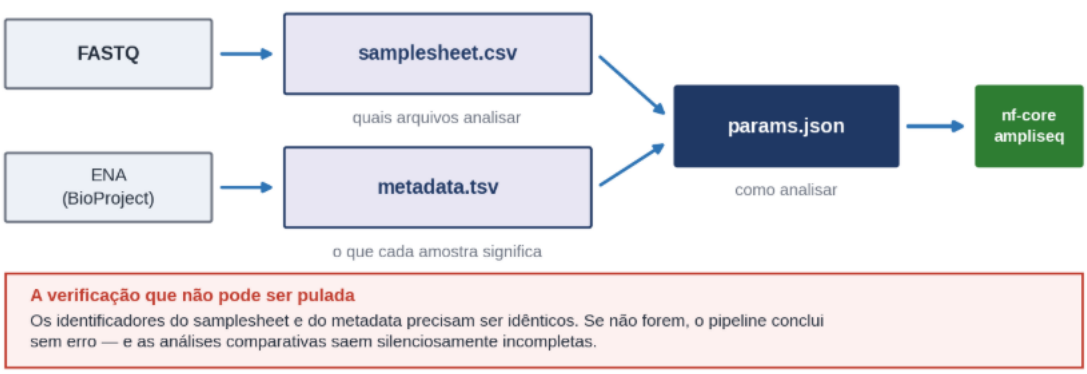

Figura 4: Os três artefatos e seus papéis. A validação cruzada entre samplesheet e metadata é o passo mais esquecido.

6.1. A folha de amostras

O pipeline não descobre as amostras automaticamente: elas são declaradas em uma folha de amostras. É nesse arquivo que a informação experimental entra na análise, e é também a origem mais frequente de erros de execução.

O formato **depende do layout da biblioteca**. Os dois conjuntos usados neste módulo exigem formatos diferentes.

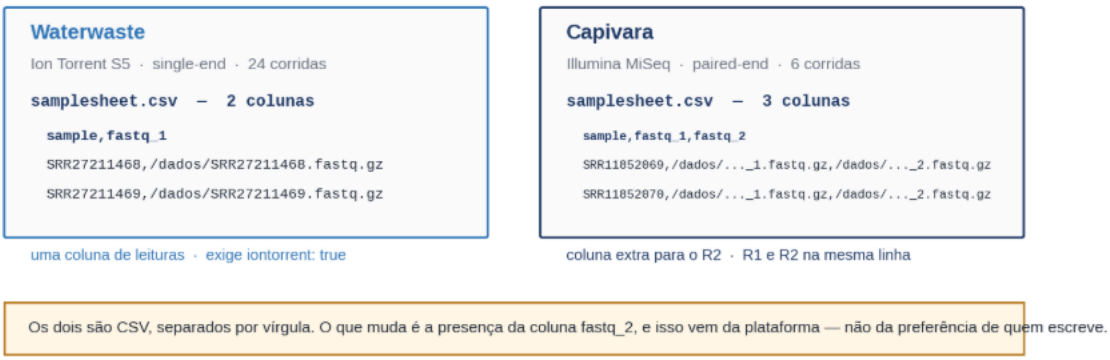

Figura 5: Os dois formatos usados no módulo. Usar o formato errado interrompe a execução na validação.

Três aspectos merecem atenção em qualquer dos formatos. Os caminhos devem ser **absolutos**. O identificador precisa ser **único por linha**. E a coluna `run`, quando existe, identifica a corrida de

sequenciamento de origem: quando as amostras provêm de corridas distintas, o perfil de erro precisa ser estimado separadamente para cada uma.

Dica.

Prefira **gerar** a folha de amostras a partir do diretório de dados, em vez de digitá-la. Assim cada linha aponta necessariamente para um arquivo que existe. Antes de submeter, verifique:

- o número de colunas, com `awk -F',' '{print NF}' samplesheet.csv | sort -u`
- identificadores duplicados, com `cut -d',' -f1 ... | sort | uniq -d`
- a existência de cada arquivo declarado

Atenção.

Arquivos TSV são separados por tabulação, e não por espaços. Editores de planilha convertem tabulações em espaços ou inserem aspas, e o erro só aparece depois, na validação do pipeline.

6.2. Os metadados

O arquivo de metadados é opcional para a execução, mas é o que viabiliza a parte comparativa da análise. Sua primeira coluna deve se chamar `ID` e corresponder exatamente aos identificadores declarados na folha de amostras.

metadata.tsv

```
ID          sample_type         site             collection_month
SRR27211473 urban_wastewater    Parque_Novo_Mundo  January
SRR27211509 hospital_wastewater HCFMUSP          March
SRR27211510 surface_water       Raia_USP         March
```

Sem metadados, o pipeline produz tabelas de abundância e métricas descritivas. Com metadados, produz diversidade beta comparativa entre grupos e análises de abundância diferencial. A qualidade da pergunta científica está, portanto, codificada neste arquivo.

Quando o conjunto está depositado publicamente, o metadata pode ser **derivado dos registros públicos** em vez de digitado — o que elimina erros de transcrição e documenta a procedência de cada categoria. O roteiro guiado demonstra esse percurso a partir da API do ENA.

Atenção.

Antes de submeter, compare os identificadores dos dois arquivos com `diff`. O ampliseq conclui a execução mesmo quando parte das amostras não encontra metadado, e as análises comparativas saem silenciosamente incompletas.

6.3. O arquivo de parâmetros

Quando o mesmo conjunto de parâmetros é reutilizado, ou quando a turma é numerosa, é preferível declarar os parâmetros em um arquivo, reduzindo erros de digitação e produzindo um registro auditável da execução.

Os parâmetros são declarados em um arquivo JSON, o que reduz erros de digitação e produz um registro auditável da execução. O arquivo deve se chamar **${DATASET}.json** e ficar dentro de **$RUN** — é assim que o script de submissão vai procurá-lo. Use **caminhos absolutos**: a tarefa executa em outro diretório.

Arquivo: Waterwaste.json

```
{
 "input":    "/scratch/<usuario>/ampliseq/runs/Waterwaste/samplesheet.csv",
 "metadata":  "/scratch/<usuario>/ampliseq/runs/Waterwaste/metadata.tsv",
 "outdir":    "/scratch/<usuario>/ampliseq/runs/Waterwaste/results",
 "FW_primer": "GTGYCAGCMGCCGCGGTAA",
 "RV_primer": "GGACTACNVGGGTWTCTAAT",
 "iontorrent": true,
 "metadata_category":    "sample_type",
 "qiime_adonis_formula": "sample_type",
 "ancom": true
}
```

```
# erro de virgula em JSON e comum; valide antes de submeter
$ python3 -m json.tool "$RUN/${DATASET}.json" > /dev/null && echo OK
OK
```

Atenção.

Parâmetros do pipeline devem ser informados por **-params-file**, e **nunca** por **-c**. A opção **-c** destina-se à configuração de execução, como recursos e executor; declarar parâmetros por ela resulta em erro.

Sem **"iontorrent": true**, o DADA2 aplica parâmetros otimizados para Illumina. A execução conclui, mas a inferência de ASVs fica comprometida — **terminar sem erro não significa resultado correto**.

7. Anatomia do comando

O comando de execução deve ser lido como um protocolo, campo por campo:

Argumento	O que determina
run <caminho>	a cópia local do pipeline em **$SHARED/pipelines/**
-profile singularity	containers e executor no Marvin
-c	configuração de execução: infraestrutura e recursos

Argumento	O que determina
-params-file	os parâmetros científicos, em JSON
-work-dir	onde ficam os arquivos intermediários e o cache
-resume	reaproveita as tarefas já concluídas com sucesso
-log	o arquivo de log da execução
-with-report / -with-timeline / -with-trace	os relatórios de desempenho

Nota.

Como usamos a **cópia local** do pipeline, **-r 2.18.0** não se aplica: a versão já está fixada pelo próprio caminho **nf-core-ampliseq-2.18.0**. Em execuções a partir do GitHub, o **-r** é obrigatório — é ele que torna a análise reproduzível.

Os nomes **--FW_primer** e **--RV_primer** correspondem à versão 2.18.0. A versão em desenvolvimento renomeou esses parâmetros.

8. Os perfis de configuração

Dois arquivos passados com **-c** descrevem, em camadas, **onde** e **com quanto** cada processo roda.

Arquivo	Papel
/scratch/nf-work/profiles/marvin_dgp.config	descreve a **infraestrutura**: executor SLURM, partição, filas e limites globais
/scratch/nf-work/profiles/ampliseq.config	**refina os recursos por processo**: CPU, memória e tempo de DADA2, QIIME2 e demais etapas

```
# confirme que os dois arquivos existem antes de submeter
$ ls -l /scratch/nf-work/profiles/
marvin_dgp.config  ampliseq.config
```

Atenção.

A ordem dos `-c` importa. O específico deve vir depois do geral: **marvin_dgp.config** primeiro, **ampliseq.config** em seguida. Invertida, a configuração geral sobrescreve a específica e os ajustes por processo se perdem.

Ajuste recursos **por arquivo de configuração**, nunca alterando o código do workflow.

9. O script de submissão SLURM

9.1. O cabeçalho SBATCH

As linhas **#SBATCH** não são comentários: são o pedido de recursos ao cluster, lido **antes** de o script começar a rodar.

Diretiva	Valor aqui	Por quê
--partition	**gui**	a fila de execução da prática
--nodelist	**dgpu01**	fixa o nó; o **/scratch** é disco local
--cpus-per-task	**2**	apenas o orquestrador; cada tarefa pede o seu
--mem	**4G**	precisa da unidade; **4** sozinho significa 4 MB
--time	**08:00:00**	ao ultrapassar, o job é encerrado mesmo inacabado
--output / **--error**	**logs/slurm-%j**	**%j** vira o número do job; a pasta precisa existir antes

Atenção.

O job submetido é o **orquestrador**, não a análise. Ele pede pouco (2 CPUs, 4G) porque seu trabalho é distribuir tarefas ao SLURM; cada tarefa solicita seus próprios recursos via **ampliseq.config**.

NXF_OPTS deve caber dentro do **--mem**: pedimos 4G ao SLURM e damos **-Xmx4G** à JVM. Se o job morrer com **código 137** logo no início, reduza o **-Xmx** ou aumente o **--mem**.

As diretivas **#SBATCH** são lidas pelo SLURM **antes** da expansão de variáveis do shell — **$USER** não funciona ali. Edite o caminho com o seu usuário.

9.2. O script completo

Arquivo: submit_ampliseq_waterwaste.sh

```
#!/bin/bash
#SBATCH --job-name=ampliseq_waterwaste
#SBATCH --partition=gui
#SBATCH --nodelist=dgpu01       # /scratch e local: fixar o no
#SBATCH --output=/scratch/nilson.coimbra/ampliseq/logs/Waterwaste/slurm-%j.out
#SBATCH --error=/scratch/nilson.coimbra/ampliseq/logs/Waterwaste/slurm-%j.err
#SBATCH --cpus-per-task=2       # apenas o orquestrador
#SBATCH --mem=4G
#SBATCH --time=08:00:00


module load nextflow/26.04.4


SHARED=/shared/training/TRAD2026
SCRATCH=/scratch/$USER
DATASET=Waterwaste
```

```
# --- containers: repositorio pre-validado, conjunto amplicon -------
export NXF_SINGULARITY_CACHEDIR="$SHARED/containers/nextflow/amplicon"
export SINGULARITY_TMPDIR="$SCRATCH/tmp/singularity/${SLURM_JOB_ID}"
export APPTAINER_TMPDIR="$SINGULARITY_TMPDIR"
mkdir -p "$SINGULARITY_TMPDIR"

# --- nextflow home e credencial do GitHub -------------------------
export NXF_HOME="/scratch/$USER/nxf-home"
export NXF_OPTS="-Xms512M -Xmx4G"
mkdir -p "$NXF_HOME"

if [[ -f "$HOME/.nextflow/scm" ]]; then
  cp "$HOME/.nextflow/scm" "$NXF_HOME/scm"
  chmod 600 "$NXF_HOME/scm"
fi

# --- diretorios da execucao ---------------------------------------
export NXF_WORK="$SCRATCH/ampliseq/work/$DATASET"
export RUN="$SCRATCH/ampliseq/runs/$DATASET"
export LOGS="$SCRATCH/ampliseq/logs/$DATASET"
export METRICS="$SCRATCH/ampliseq/metrics/$DATASET"

RUN="$SCRATCH/ampliseq/runs/$DATASET/"
WORK="$SCRATCH/ampliseq/work/$DATASET"
LOGS="$SCRATCH/ampliseq/logs/$DATASET/"
METRICS="$SCRATCH/ampliseq/metrics/$DATASET"

cd "$RUN"

# --- execucao -----------------------------------------------------
nextflow \
  -log "$LOGS/nextflow_${SLURM_JOB_ID}.log" \
  run "$SHARED/pipelines/nf-core-ampliseq-2.18.0" \
  -profile singularity \
  -c "/scratch/nf-work/profiles/marvin_dgp.config" \
  -c "/scratch/nf-work/profiles/ampliseq.config" \
  -params-file "$RUN/${DATASET}.json" \
  -work-dir "$WORK" \
  -with-report   "$METRICS/report_${SLURM_JOB_ID}.html" \
  -with-timeline "$METRICS/timeline_${SLURM_JOB_ID}.html" \
  -with-trace    "$METRICS/trace_${SLURM_JOB_ID}.tsv" \
  -resume

rm -rf "$SINGULARITY_TMPDIR"
```

Atenção.

A **barra invertida precisa ser o último caractere da linha**. Um espaço depois dela, ou uma linha em branco no meio da sequência, quebra a continuação e o **nextflow** recebe um comando truncado.

```
# checa sintaxe sem executar nada
$ bash -n submit_ampliseq_waterwaste.sh
```

9.3. Submeter e acompanhar

```
$ mkdir -p "$LOGS"
$ sbatch submit_ampliseq_waterwaste.sh
Submitted batch job 4821501

$ squeue -u $USER
 JOBID PARTITION    NAME    USER ST    TIME NODELIST
4821501     gui ampliseq  usuario  R    1:04 dgpu01

$ tail -f "$LOGS"/slurm-4821501.out
N E X T F L O W  ~  version 26.04.4
Launching `/shared/training/TRAD2026/pipelines/nf-core-ampliseq-2.18.0`
executor >  slurm (24)
[8c/1a4f2e] NFCORE_AMPLISEQ:AMPLISEQ:FASTQC   [100%] 24 of 24
[3d/9b7c01] NFCORE_AMPLISEQ:AMPLISEQ:CUTADAPT [ 66%] 16 of 24
```

Três elementos merecem atenção na saída:

- **executor > slurm** confirma que as tarefas vão para a fila, e não rodam dentro do job orquestrador.
- O contador **24 of 24** mostra **uma tarefa por amostra** — o paralelismo aparecendo na prática.
- O código **[8c/1a4f2e]** é a pasta em **work/** onde aquela tarefa rodou. Anote um deles para inspecionar.

Dica.

Para sair do **tail -f** sem interromper o job, pressione **Ctrl+C**. Para cancelar de fato, **scancel <JOBID>** — mas **não apague o `$NXF_WORK`**: ele permite retomar de onde parou.

10. Diagnóstico de falhas e retomada

Falhas fazem parte da execução de pipelines e devem ser tratadas como conteúdo, e não como acidente. O procedimento de diagnóstico é sempre o mesmo, independente do pipeline: localize a tarefa que falhou, entre no diretório dela e leia os arquivos ocultos.

```
# a mensagem de erro informa o diretorio da tarefa
$ cd "$NXF_WORK"/f2/7ab91c*
$ cat .command.sh    # o comando exato que rodou
```

```
$ cat .command.err   # o erro da ferramenta cientifica
$ cat .exitcode      # 0 = sucesso
```

Arquivo	Conteúdo
.command.sh	o comando exato executado; pode ser reproduzido manualmente
.command.err	a mensagem de erro emitida pela ferramenta científica
.command.log	a saída combinada da execução
.command.run	o invólucro gerado pelo Nextflow, com container e SLURM
.exitcode	o código de saída; zero indica sucesso

Corrigido o problema, a execução é relançada com a mesma linha de comando acrescida de -resume. As tarefas previamente concluídas são identificadas como cached e não são reexecutadas.

Atenção.

Se o **-resume** reexecutar tudo, o **diretório de trabalho mudou** entre as duas execuções. No Marvin, a causa mais comum é o job ter caído em **outro nó** — como o **/scratch** é disco local, o cache anterior não existe ali. Por isso o **--nodelist** é fixado.

11. Interpretação dos resultados

Os resultados publicados em --outdir devem ser percorridos na ordem em que a análise biológica se desenvolve, e não na ordem alfabética dos diretórios. O primeiro número a conferir é sempre quantas leituras sobreviveram a cada etapa.

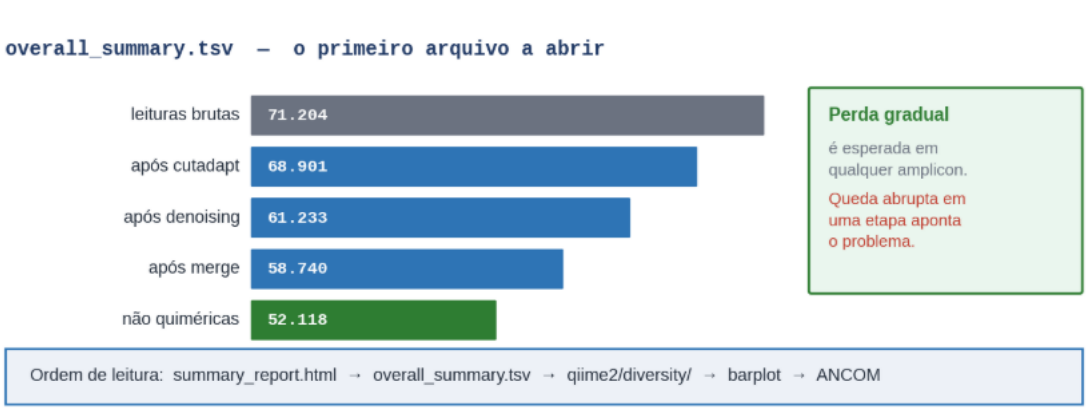

Figura 8: O funil de leituras. A proporção retida em cada etapa é o diagnóstico mais rápido da qualidade de uma análise de amplicon.

Diretório	Pergunta que responde
fastqc/ multiqc/	O dado bruto tem qualidade e profundidade suficientes?
cutadapt/	Quantas leituras sobreviveram à remoção de primers?
dada2/	Quantas ASVs foram inferidas? Quantas quimeras removidas?
qiime2/diversity/	A rarefação saturou? Os grupos se separam?
qiime2/barplot/	Qual a composição por nível taxonômico?
qiime2/ancom*/	Quais táxons diferem entre os grupos do metadata?
phyloseq/	Objeto para continuidade da análise em R
summary_report/	Visão consolidada; é por aqui que se começa

```
$ cd "$RUN" && ls results/
cutadapt  dada2  fastqc  multiqc  overall_summary.tsv
phyloseq  qiime2  summary_report

# o primeiro numero a conferir
$ column -t results/overall_summary.tsv | head -4
```

As visualizações do QIIME2 são artefatos com extensão .qzv. Diferentemente de uma figura estática, um .qzv é interativo: permite filtrar amostras, alternar métricas e exportar a figura e a tabela que a sustenta. Abra-os em https://view.qiime2.org — o site processa o arquivo no próprio navegador, sem enviar dados a servidor algum.

Atenção.

Uma ordenação de diversidade beta separa amostras por qualquer fonte de variação dominante, e não apenas pela variável de interesse. Antes de atribuir a separação observada ao seu contraste, verifique se ela não acompanha a profundidade de sequenciamento ou a réplica biológica.

```
# execute no seu computador, nao no cluster
$ scp -r usuario@marvin.cnpem.br:\
 /scratch/<usuario>/ampliseq/runs/Waterwaste/results/summary_report ./
```

Atenção.

Copiar apenas o **index.html** faz a página abrir quebrada: as visualizações dependem de arquivos auxiliares na mesma pasta. Use sempre **scp -r**. Os arquivos **.qzv** abrem em https://view.qiime2.org.

12. Erros comuns

Sintoma	Causa provável	Correção
API rate limit exceeded	sem token do GitHub configurado	usar a cópia local em **$SHARED/pipelines/**
Cannot find revision	versão inexistente ou cópia local antiga	conferir o caminho **nf-core-ampliseq-2.18.0**
Container não encontrado	**NXF_SINGULARITY_CACHEDIR** apontando para **ampliseq**	usar o conjunto **amplicon**
Erro de validação do samplesheet	separador virou espaço ao copiar de planilha	reescrever o arquivo por script
Colunas rejeitadas	formato paired-end em dado single-end	usar **sample,fastq_1**
Identificadores duplicados	rótulo biológico repetido entre réplicas	usar o acesso SRR
Missing required parameter	parâmetro grafado com um hífen só	dois hífens para o pipeline
Job falha sem log	**logs/** não existia no momento do **sbatch**	**mkdir -p "$LOGS"** antes de submeter
Código de saída 137	memória insuficiente na tarefa ou na JVM	ajustar **ampliseq.config** ou **--mem**/**-Xmx**
-resume reexecuta tudo	job caiu em outro nó; **/scratch** é local	fixar **--nodelist** e **-work-dir**
Ajustes de recurso ignorados	ordem dos **-c** invertida	**marvin_dgp.config** antes de **ampliseq.config**
Perda excessiva no cutadapt	primer incorreto ou dado sem primer	conferir a região amplificada
Comparativas incompletas	IDs do metadata não casam com o samplesheet	validar com **diff**

13. Boas práticas

Categoria	Boas práticas
Reprodutibilidade	Fixar sempre a versão do pipeline; registrar a linha de comando completa; preferir arquivo de parâmetros a argumentos avulsos
Dados	Tratar os dados brutos em **/shared** como somente leitura; não versionar dados ou resultados grandes; documentar a origem dos arquivos
Entradas	Gerar samplesheet e metadata por script, não à mão; validar colunas, duplicidade e existência dos arquivos; cruzar os IDs com **diff**
Execução	Rodar o **-profile test** antes da análise real; nunca executar no nó de login; fixar o nó com **--nodelist**; separar o **work/** por conjunto
Recursos	Ajustar recursos por configuração, não alterando o código do workflow; acompanhar **report.html** e **trace.tsv** para dimensionar execuções futuras

Categoria	Boas práticas
Diagnóstico	Preservar o **$NXF_WORK** enquanto a retomada for necessária; iniciar a investigação pelo **.command.err** da tarefa que falhou
Interpretação	Verificar as perdas ao longo das etapas antes de interpretar resultados; confirmar a base de referência utilizada; registrar as limitações do desenho

14. Os dois roteiros práticos

A parte prática do módulo está em dois documentos separados, que aplicam os conceitos desta apostila a conjuntos de dados distintos. A separação é intencional: os grupos não devem repetir o que viram, mas **transferir o procedimento** para um conjunto novo — com outro formato de entrada.

	Guiada — Waterwaste	Prática — Capivara
Conduzida por	instrutor, turma acompanhando	cada grupo, de forma autônoma
BioProject	PRJNA880881	PRJNA563062
Plataforma	Ion Torrent S5, *single-end*	Illumina MiSeq, *paired-end*
Entrada	**CSV**, 2 colunas	**CSV**, 3 colunas
Corridas	24 — desenho balanceado	6 — de apenas 3 BioSamples
Variável principal	sample_type	sample_type
Foco	construir as entradas do zero	falha provocada e diagnóstico

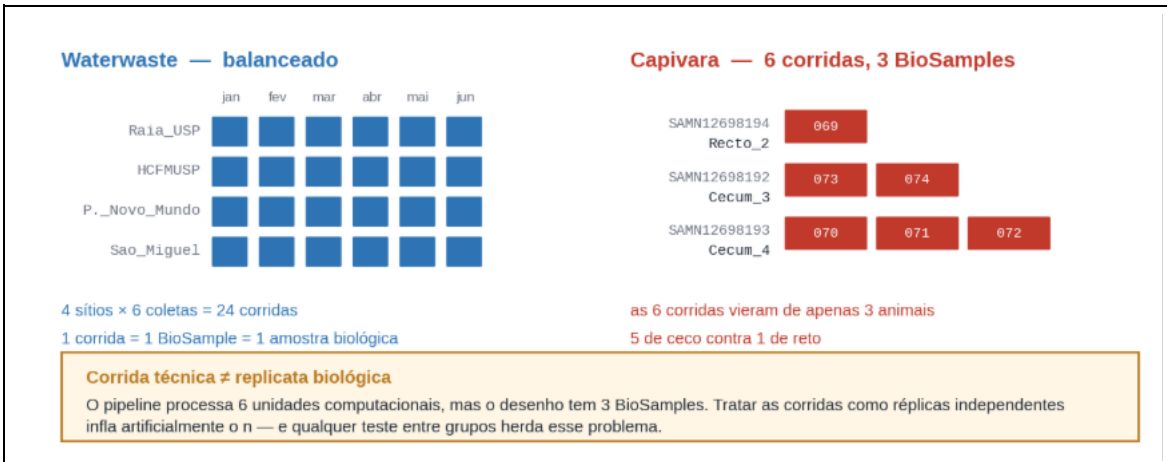

Figura 9: Os desenhos experimentais dos dois conjuntos.

Nota.

> **Corrida técnica não é réplica biológica.** No Waterwaste, 1 corrida = 1 amostra biológica. Na Capivara, as 6 corridas vieram de apenas 3 animais: o pipeline processa 6 unidades computacionais, mas o desenho tem 3 BioSamples. Tratar as corridas como réplicas independentes infla artificialmente o n.

15. Referências e leituras recomendadas

- Documentação do Nextflow — https://www.nextflow.io/docs/latest/
- Treinamento oficial Nextflow — https://training.nextflow.io/
- nf-core — https://nf-co.re/
- nf-core/ampliseq 2.18.0 — https://nf-co.re/ampliseq/2.18.0/
- Parâmetros do ampliseq — https://nf-co.re/ampliseq/2.18.0/parameters
- Visualizador do QIIME2 — https://view.qiime2.org
- Portal do ENA — https://www.ebi.ac.uk/ena/browser/
- Repositório do treinamento — https://github.com/cnpem/TRAD2026

Anexo A — Tabela de configurações do módulo

Item	Valor
Módulo	**nextflow/26.04.4**
Pipeline	**/shared/training/TRAD2026/pipelines/nf-core-ampliseq-2.18.0**
Perfil	**singularity**
Configs (nesta ordem)	**/scratch/nf-work/profiles/marvin_dgp.config** e depois **/scratch/nf-work/profiles/ampliseq.config**
Cache de containers	**$SHARED/containers/nextflow/amplicon**
Params file	**$RUN/${DATASET}.json**
SHARED	**/shared/training/TRAD2026**
SCRATCH	**/scratch/$USER**
GRUPO	**grupoN**
DATASET	**Waterwaste** (guiada) e **Capivara** (prática)
NXF_WORK	**$SCRATCH/ampliseq/work/$DATASET**
RUN	**$SCRATCH/ampliseq/runs/$DATASET**
LOGS	**$SCRATCH/ampliseq/logs/$DATASET**
METRICS	**$SCRATCH/ampliseq/metrics/$DATASET**

Item	Valor
NXF_HOME	**/scratch/$USER/nxf-home**
NXF_OPTS	**-Xms512M -Xmx4G**
SINGULARITY_TMPDIR	**$SCRATCH/tmp/singularity/${SLURM_JOB_ID}**
APPTAINER_TMPDIR	o mesmo valor de **SINGULARITY_TMPDIR**
Partição e nó	**gui** e **dgpu01**
Recursos do orquestrador	2 CPUs, 4G de memória, 08:00:00
Dados brutos	**/shared/training/TRAD2026/raw_data/16S_<DATASET>**

Anexo B — Cartão de referência rápida

```
# 1. sessao
ssh usuario@marvin.cnpem.br && ssh dgpu01
module load nextflow/26.04.4

# 2. ambiente
export SHARED=/shared/training/TRAD2026
export SCRATCH=/scratch/$USER
export GRUPO=grupoN
export DATASET=Waterwaste
export NXF_SINGULARITY_CACHEDIR="$SHARED/containers/nextflow/amplicon"
export NXF_WORK="$SCRATCH/ampliseq/work/$DATASET"
export RUN="$SCRATCH/ampliseq/runs/$DATASET"
export LOGS="$SCRATCH/ampliseq/logs/$DATASET"
export METRICS="$SCRATCH/ampliseq/metrics/$DATASET"
mkdir -p "$RUN" "$LOGS" "$METRICS" "$NXF_WORK" && cd "$RUN"

# 3. validar entradas
column -s, -t "$RUN/samplesheet.csv"
cut -d',' -f1 "$RUN/samplesheet.csv" | tail -n +2 | sort | uniq -d
python3 -m json.tool "$RUN/${DATASET}.json" > /dev/null && echo OK

# 4. submeter
bash -n submit_ampliseq_waterwaste.sh
sbatch  submit_ampliseq_waterwaste.sh

# 5. acompanhar
squeue -u $USER
tail -f "$LOGS"/slurm-<JOBID>.out

# 6. diagnosticar
cd "$NXF_WORK"/<xx>/<hash>* && cat .command.err .exitcode
```

```
# 7. resultados
cd "$RUN" && column -t results/overall_summary.tsv | head -4
```