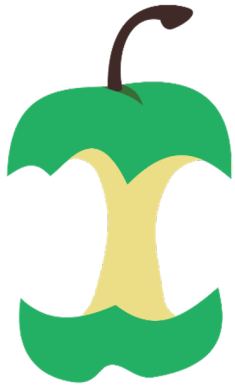
nf-core



A **community** effort to collect a curated set of analysis pipelines built using Nextflow.

nf-core in numbers

77

✅ Released pipelines

35

🔧 Under development

19

Genomics

3

Transcriptomics

9

Metagenomics

1

Metabolomics

2

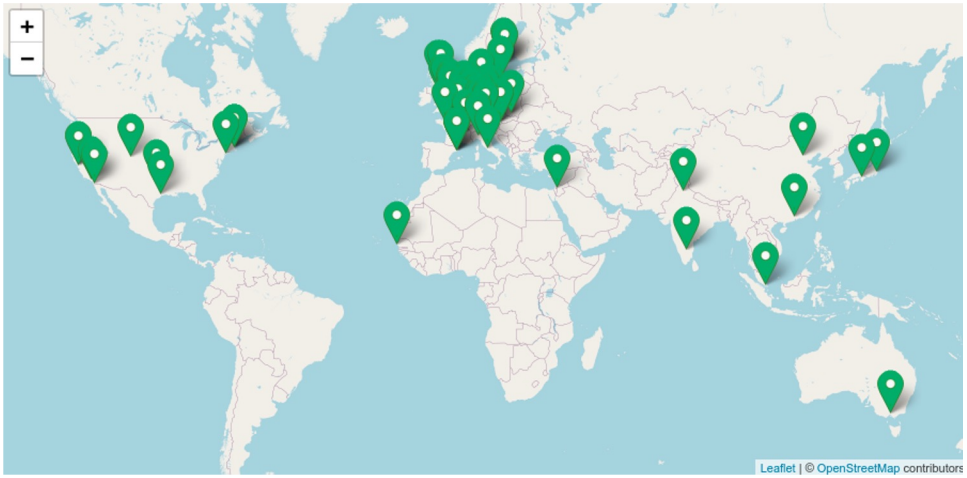
(Bio)imaging

N.B.: category numbers from Feb. 2025!

nf-core in numbers

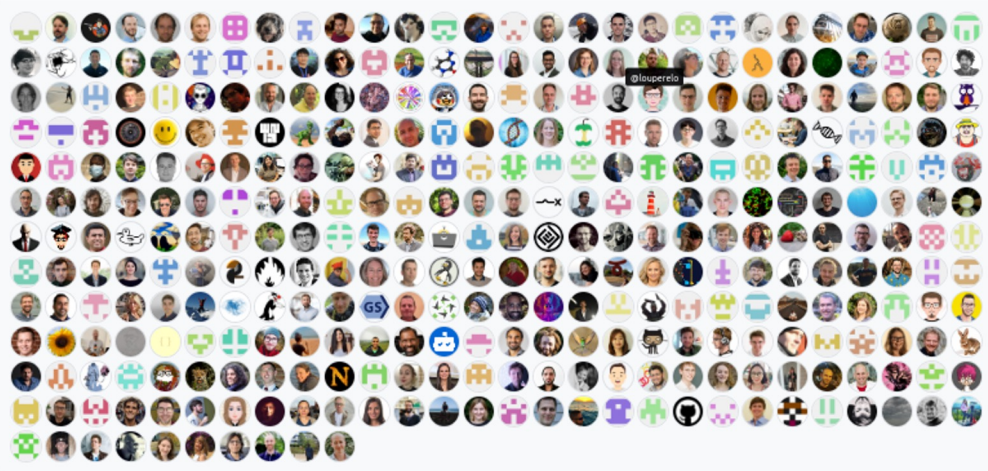
>11K

Slack Members



1.7K

GitHub Organisation Members



nf-core principles

Principles that guide the nf-core community

Cooperation

Develop with the community

Standards

Use a common template

Collaboration

No duplicate pipelines within nf-core

Helper Tools

Tools built for everyone

Compatibility

Tools work for any Nextflow pipeline

Components

Collaborate on component-level

nf-core components

Pick and choose which component you need

Pipelines

>95 pipelines and a base template

Subworkflows

>55 subworkflows

Modules

>1150 modules

Linting

Choose conventions to test for consistency

Schema

Validation, channels and user interface

Tooling

Development and deployment

nf-core/tools

Command line tools to help you build your pipeline with ease

Pipelines

Create from template, sync to get updates

Subworkflows

Create, install and update

Modules

Create, install, update, patch, test

Schema

Build your pipeline schema with a GUI

Linting

Test nf-core standards and best practices

Download

Fetch with singularity images for offline use

Why use nf-core ?

For users

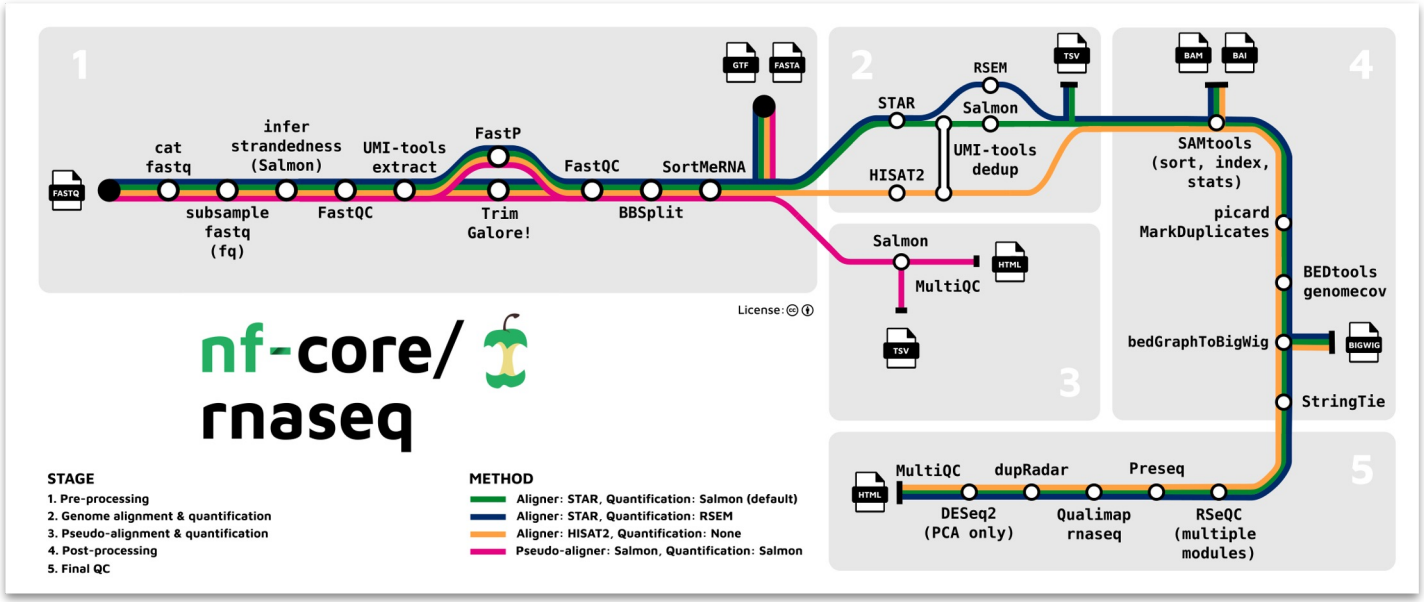
Finding Software

- What does it do?
- Does it work?
- Is it easy to use?
- Will I understand the output?
- Will it be fast enough?
- Is it easy to install?

Finding Software

- **What does it do?**
- Does it work?
- Is it easy to use?
- Will I understand the output?
- Will it be fast enough?
- Is it easy to install?

Documentation (Overview)

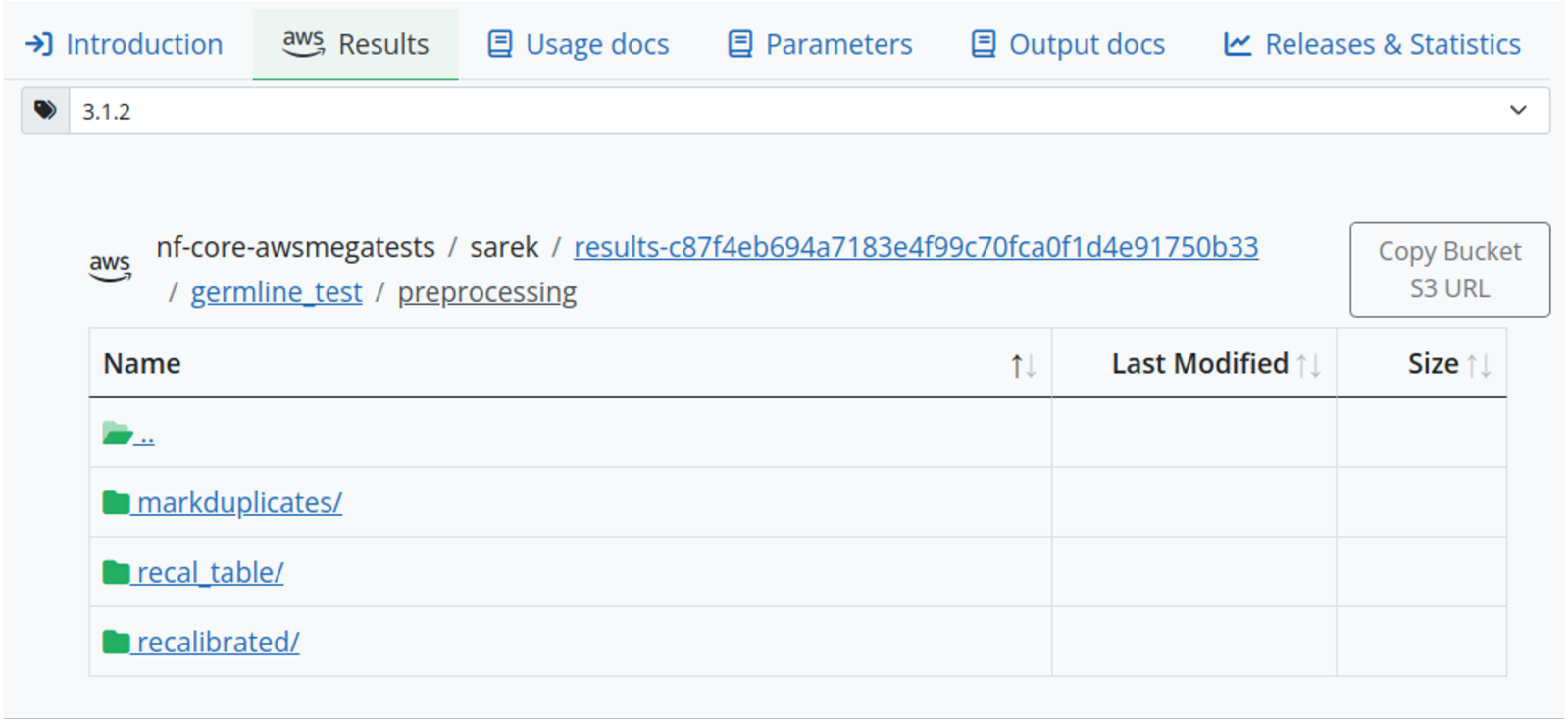


High-quality **documentation** required by nf-core guidelines
Easy to understand what and how a pipeline works

Finding Software

- What does it do?
- **Does it work?**
- Is it easy to use?
- Will I understand the output?
- Will it be fast enough?
- Is it easy to install?

Documentation (Examples)

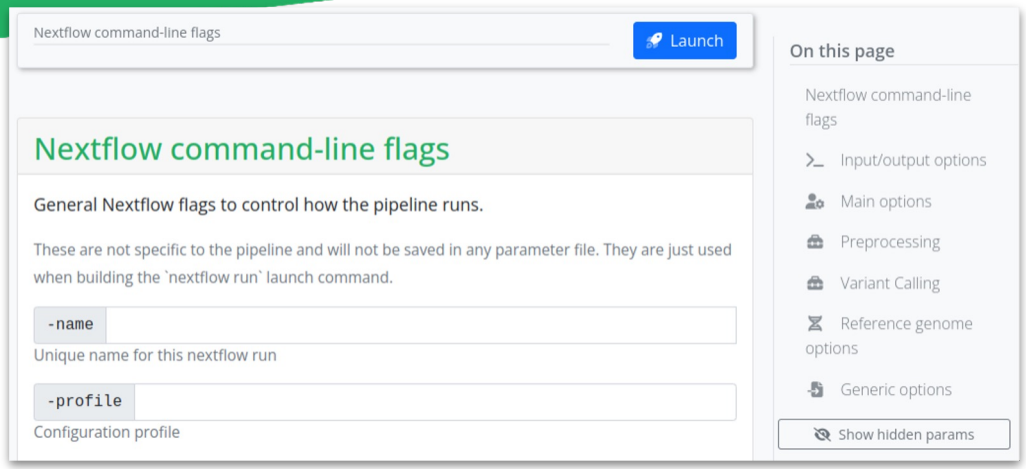


Example ‘real life’ output **automatically** and **tested** generated on every release

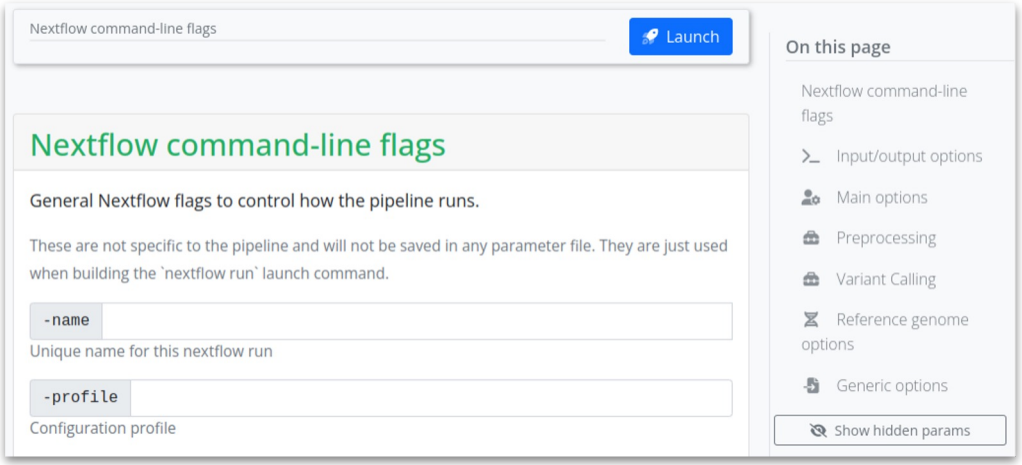
Finding Software

- What does it do?
- Does it work?
- **Is it easy to use?**
- Will I understand the output?
- Will it be fast enough?
- Is it easy to install?

Consistent (Execution)

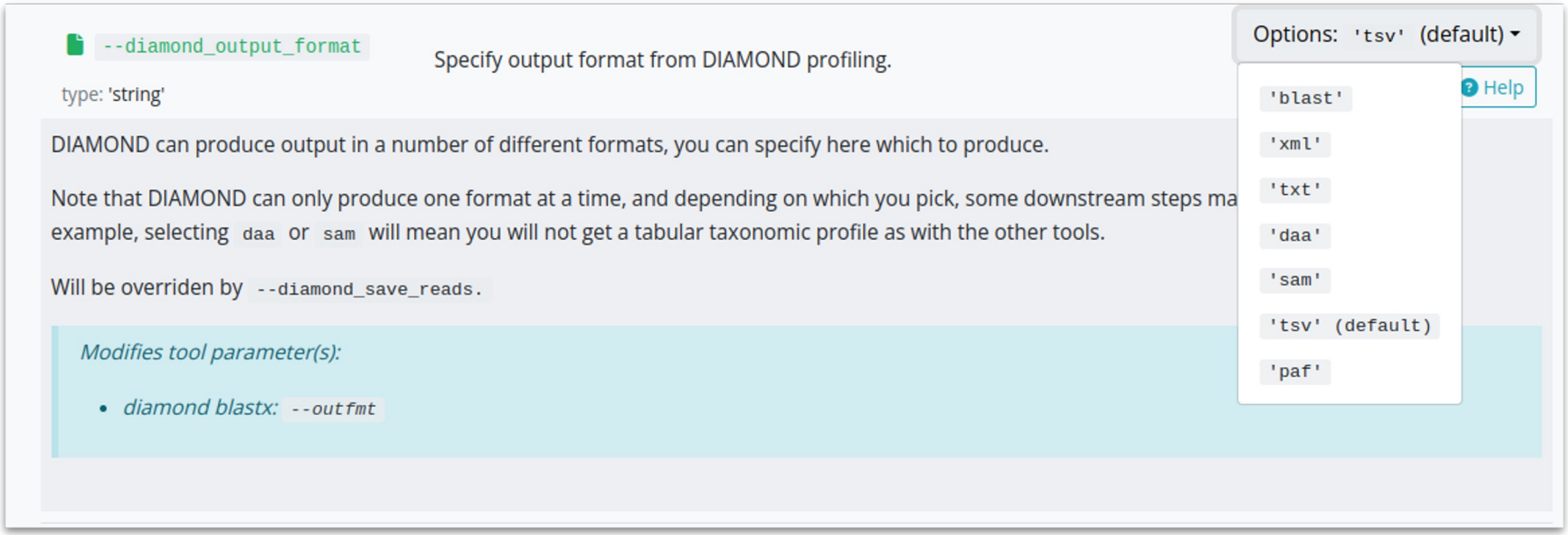



```
$ nextflow run \
>   nf-core/mag -r 2.3.0 \
>   -profile docker \
>   --input \
>   --outdir
```




```
$ nextflow run \
>   nf-core/funcscan -r 1.0.1 \
>   -profile docker \
>   --input \
>   --outdir
```

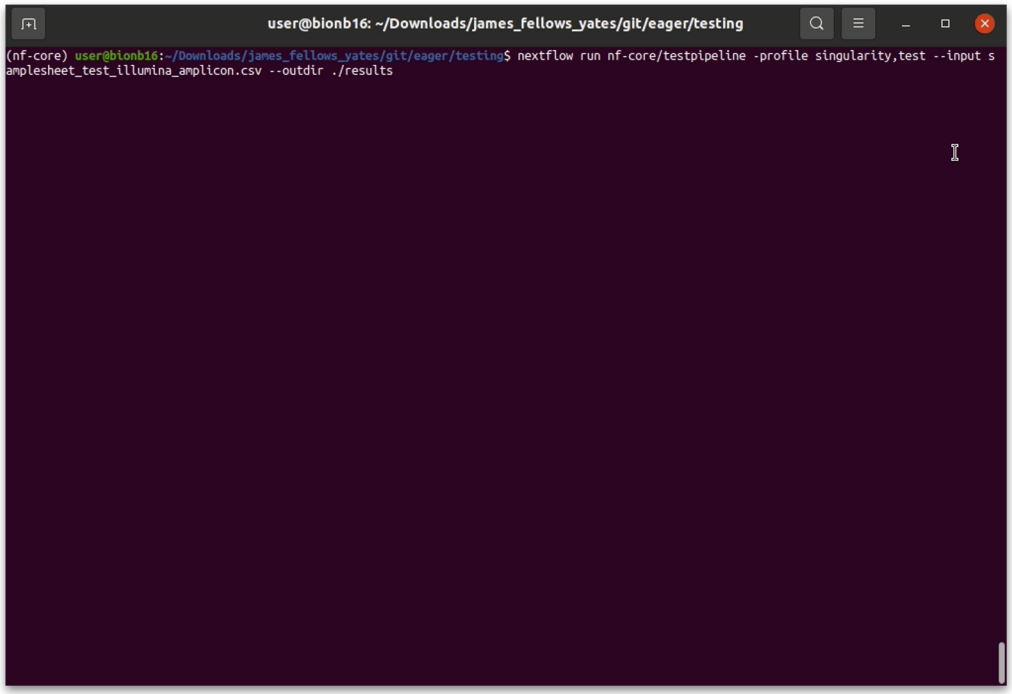
Documentation (Input)

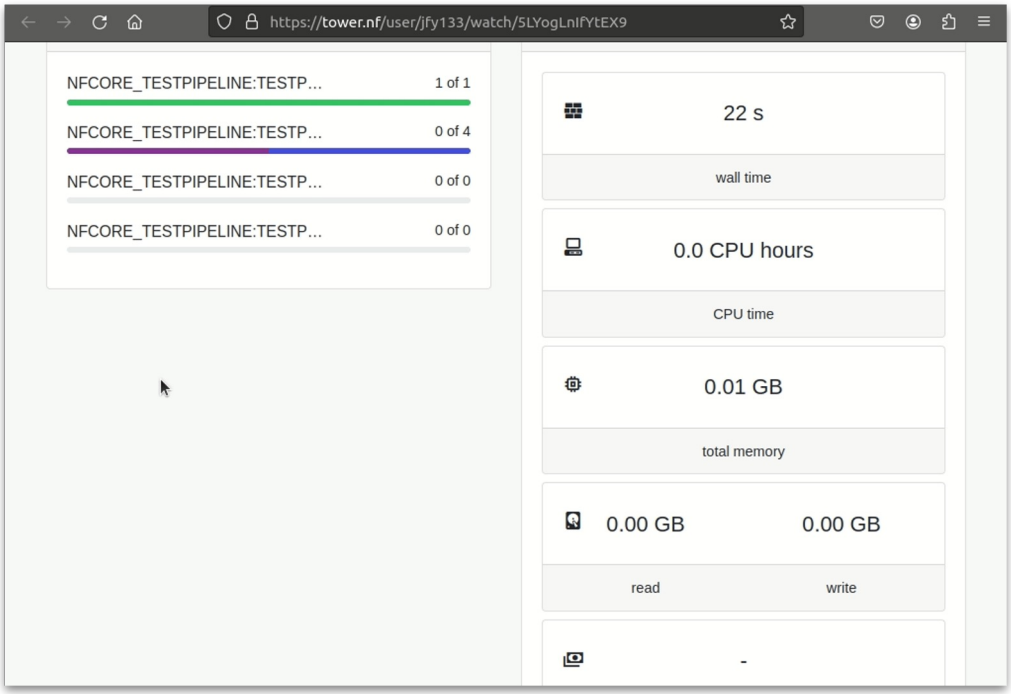


- Rich documentation & validation of inputs
- Auto-generation of commands & configuration files.
- **Consistent interface** across all pipelines for familiarity

Provenance

Real time **monitoring** of runs both in-console (and online - optional)





gif from tower.nf by Seqera labs

Finding Software

- What does it do?
- Does it work?
- Is it easy to use?
- **Will I understand the output?**
- Will it be fast enough?
- Is it easy to install?

Documentation (Output)

Porechop

Porechop is a tool for finding and removing adapters from Oxford Nanopore reads. Adapters on the ends of reads are trimmed and if a read has an adapter in its middle, it is considered a chimeric and it chopped into separate reads.

▼ Output files

- `porechop`
  - `<sample_id>.log` : Log file containing trimming statistics
  - `<sample_id>.fastq.gz` : Adapter-trimmed file

The output logs are saved in the output folder and are part of MultiQC report.You do not normally need to check these manually.

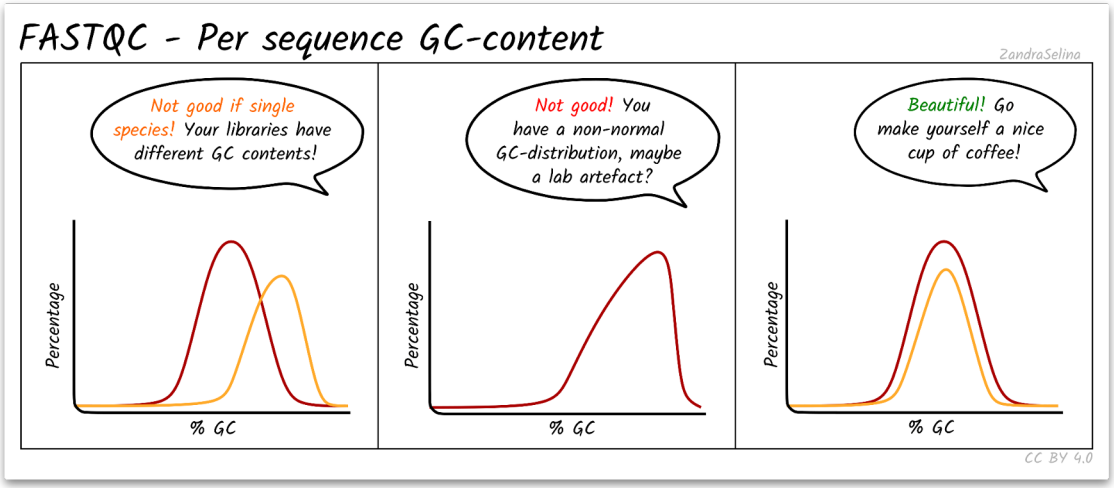
You will only find the `.fastq` files in the results directory if you provide `--save_preprocessed_reads`.

> ⚠️ *We do **not** recommend using Porechop if you are already trimming the adapters with ONT's basecaller Guppy.*

Descriptions of every output file

Additional guidance on interpretation of output

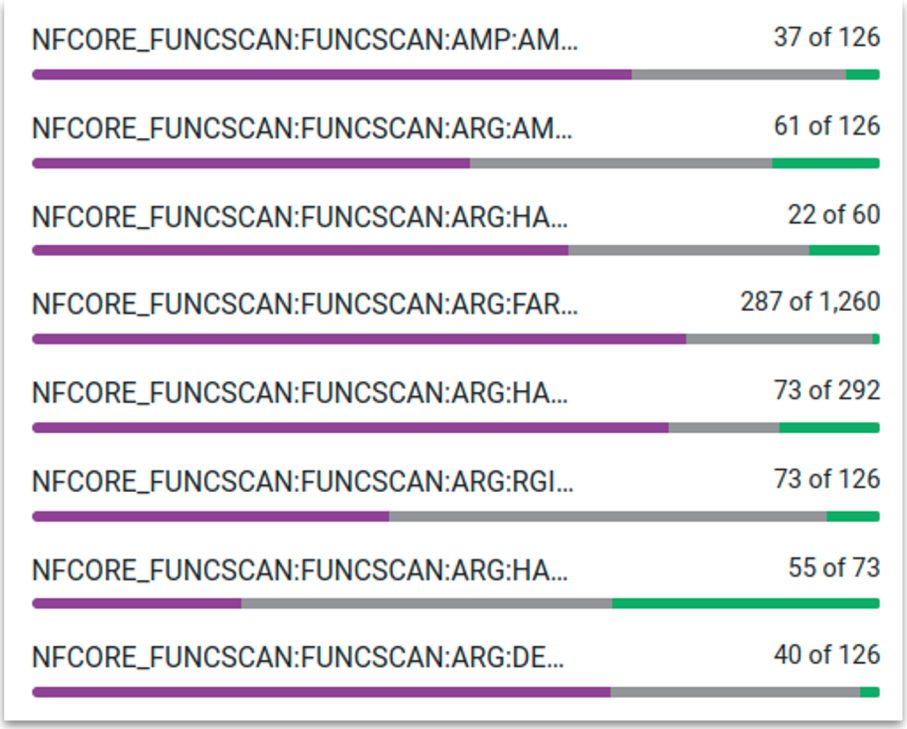
(not all pipelines)

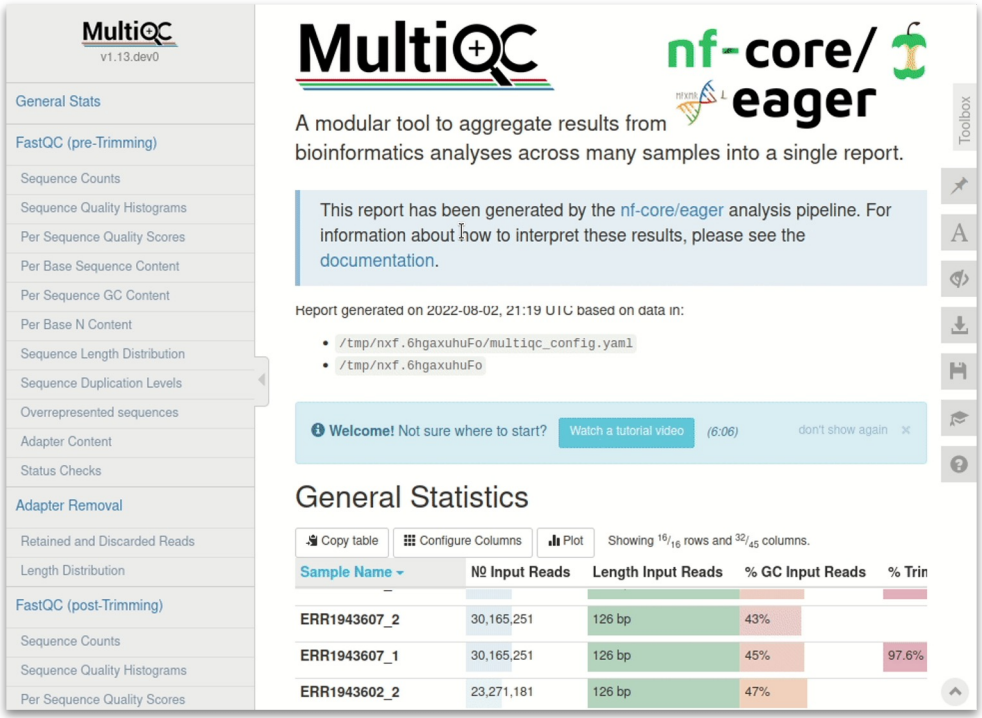


Finding Software

- What does it do?
- Does it work?
- Is it easy to use?
- Will I understand the output?
- **Will it be fast enough?**
- Is it easy to install?

Scalable





Inherent **scalability** of Nextflow means nf-core pipelines cope with 10s-1,000,000s of jobs/files

Output evaluation also scalable with MultiQC

Portable

Many **schedulers**
supported by Nextflow



& more

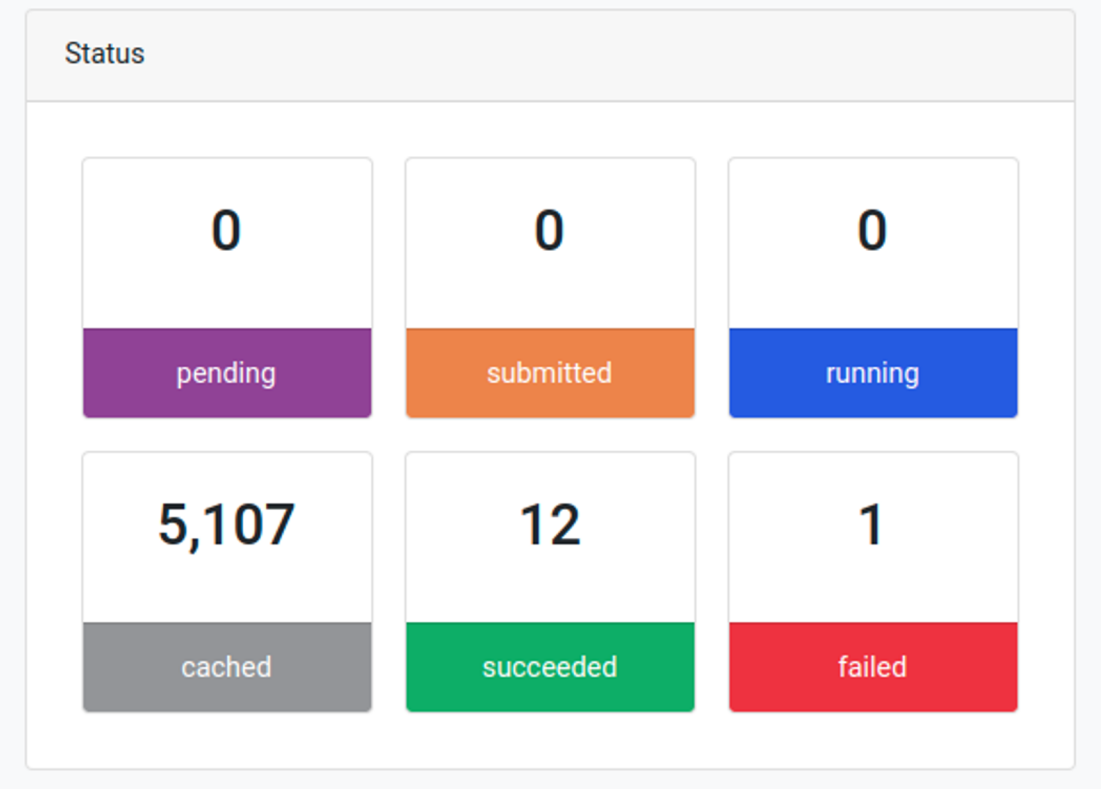
as well as the **cloud**


```
$ nextflow run nf-core/rnaseq -profile eva <...>
$ nextflow run nf-core/rnaseq -profile mpcdf,raven <...>
$ nextflow run nf-core/rnaseq -profile binac <...>
$ nextflow run nf-core/rnaseq -profile awsbatch <...>
```

Many institutional clusters/HPC already
supported by **nf-core/configs**

Fault Tolerant

- **Automatic retry** functionality
- **Resume** functionality (`-resume`)
- Ignore error functionality



Screenshot from tower.nf by Seqera labs

Finding Software

- What does it do?
- Does it work?
- Is it easy to use?
- Will I understand the output?
- Will it be fast enough?
- **Is it easy to install?**

Installation

OR

+

+

One of...

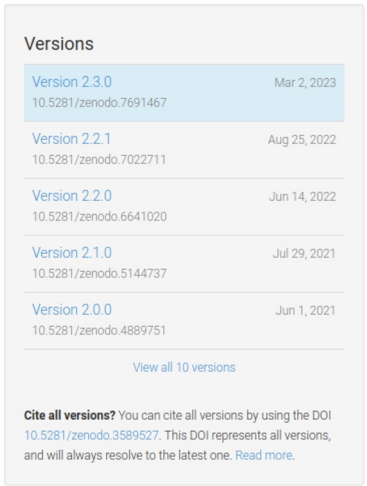

```
$ conda create -n nextflow -c bioconda nextflow

$ conda activate nextflow

$ nextflow run <...> -profile conda
```

Consistent (Execution)

Reproducible

- Strict **versioning**

```
$ nextflow run \
>   nf-core/mag -r 2.3.0 <...>
```



- Use of **container engines**/software environments

CONDA® APPTAINER docker SHIFTER SINGULARITYCE Charliecloud podman

For Users: Summary

To summarise, nf-core provides:

- Ready-to-go pipelines for many types of biological analyses
  - Community agreement
- Extremely **scalable** analyses and **portable**
- **Efficient** and **fault tolerant**
- Well **documented** and supported

Selection of relevant nf-core pipelines for metagenomics

Ancient or modern with ancient metagenomics

- nf-core/mag
- nf-core/eager
- nf-core/coproid

Modern metagenomics

- nf-core/taxprofiler
- nf-core/funcscan
- nf-core/ampliseq

Microbial metagenomics related

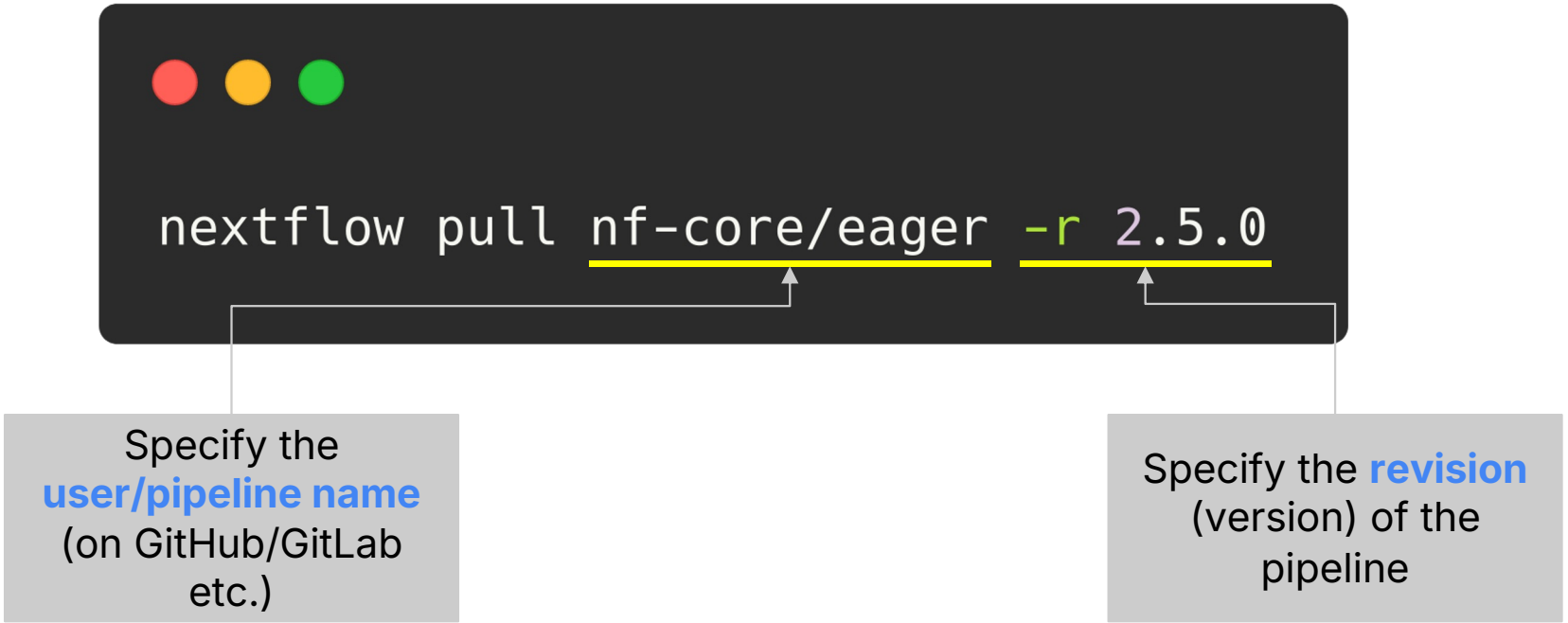
- nf-core/bacass
- nf-core/bactmap
- nf-core/viralrecon

General utility

- nf-core/fetchngs
- nf-core/differentialabundance

Installing nf-core pipelines

Any Nextflow pipeline on a web-based Git repository (GitHub, GitLab, etc.) can be downloaded with the **pull** command

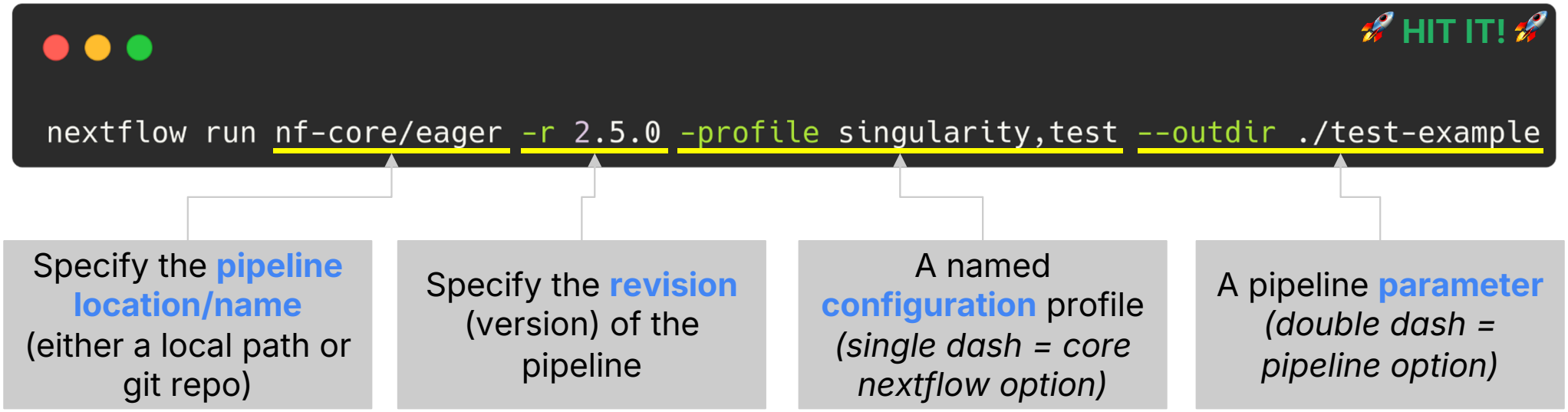


Running and monitoring nf-core pipelines

How to test an nf-core pipeline is working

Any installed Nextflow pipeline is **CLI** executed with the **run** command.

Here we will do an 'unrealistic' run with a **test** example configuration.
Every nf-core pipeline has a **test** profile containing a small input dataset.

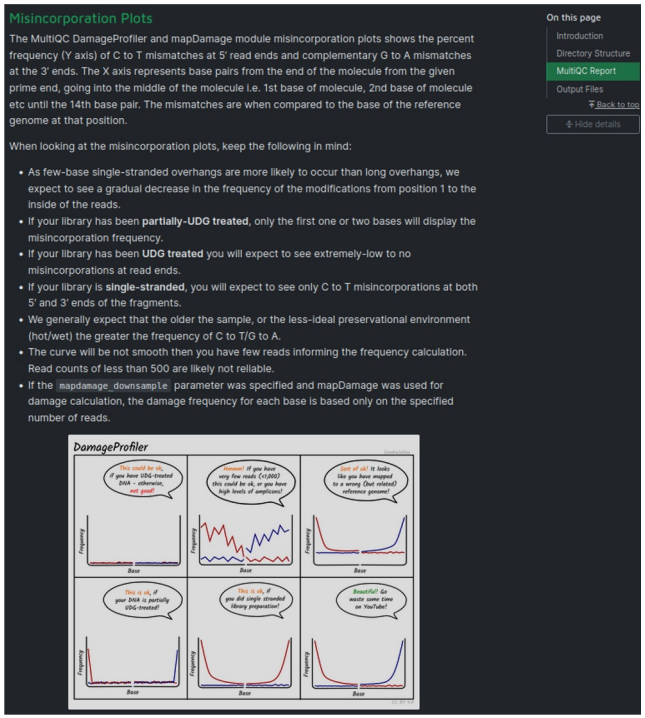


nf-core pipeline output

See the contents of `--outdir`

```
(nf-core) jfellowsy@raven01:/ptmp/jfellowsy/nextflow/eager/results> l
adapterremoval/  documentation/  mapping/       preseq/           samtools/
damageprofiler/  endorspy/       multiqc/       qualimap/
deduplication/   fastqc/         pipeline_info/  reference_genome/
(nf-core) jfellowsy@raven01:/ptmp/jfellowsy/nextflow/eager/results> tree
.
├── adapterremoval
│   └── output
│       ├── JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.combined.fq.gz
│       ├── JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.settings
│       ├── JK2802_AGAATAACCTACCA_L008_R1_001.fastq.gz.tengrand.fq_L2.se.settings
│       └── JK2802_AGAATAACCTACCA_L008_R1_001.fastq.gz.tengrand.fq_L2.se.truncated.gz
├── damageprofiler
│   ├── JK2782_rmdup
│   │   ├── 3p_freq_misincorporations.txt
│   │   ├── 3pGtoA_freq.txt
│   │   ├── 5pCtoT_freq.txt
│   │   ├── 5p_freq_misincorporations.txt
```

How to interpret? See the nf-core website!



A more realistic run command

The test **profile** uses pre-supplied data, thus is not the most realistic usage.

Most nf-core pipelines require the use of **samplesheets** to supply your own input data, and then a variety of parameters to turn on and off different steps.

A more realistic run command

Example **csv** input **samplesheet** for nf-core/eager

Sample_ Name	Library_ID	Lane	Colour_C hemistry	SeqType	Organism	Strandedness	UDG_Treatment	R1	R2	BAM
2611	ERR5766171	0	4	PE	Homo sapiens	single	none	ERR5766171_1.fastq.gz	ERR5766171_2.fastq.gz	NA
2611	ERR5766175	0	4	PE	Homo sapiens	single	none	ERR5766175_1.fastq.gz	ERR5766175_2.fastq.g z	NA
2612	ERR5766179	0	4	PE	Homo sapiens	single	none	ERR5766179_1.fastq.gz	ERR5766179_2.fastq.g z	NA
2612	ERR5766180	0	4	PE	Homo sapiens	single	none	ERR5766180_1.fastq.g z	ERR5766180_2.fastq.g z	NA

A more realistic run command

All pipeline parameters *can* be specified on the **CLI**.

⚠ DO NOT RUN ⚠

```
nextflow run nf-core/eager -r 2.5.0 -profile singularity,test --outdir ./results --input samplesheet.tsv
--fasta 'data/Mammoth_MT_Krause.fa' --skip_fastqc --clip_readlength 25 --mapper 'bowtie2'
```

Path parameter (best in quotes! Can also be URLs/S3!)

Boolean parameter (--param_name False for False)

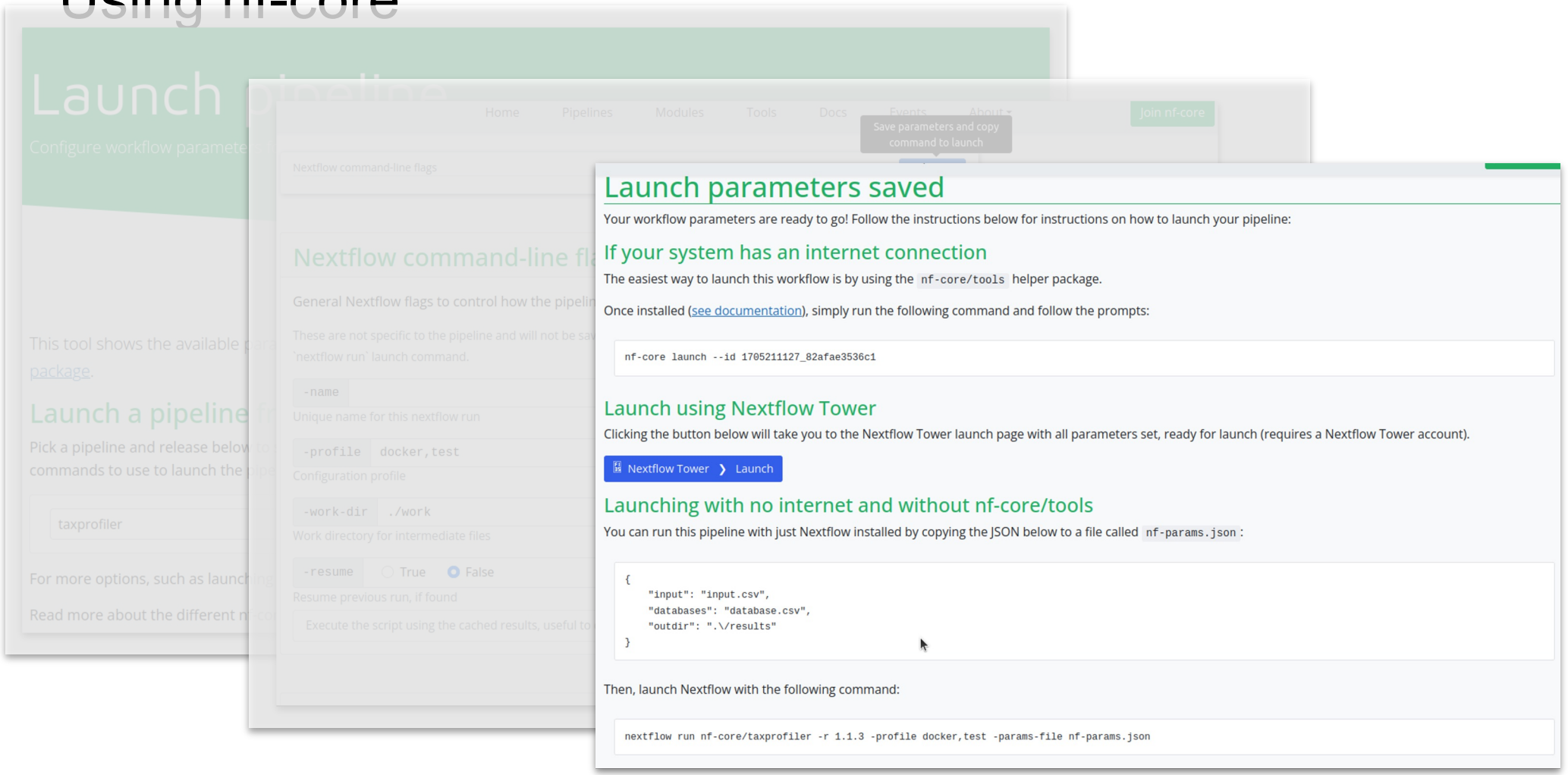
Numeric parameter

String parameter

What about GUI execution?

If you're not completely comfortable using **command line interfaces (CLI)**, We have you covered for all nf-core pipelines on: https://nf-co.re !

Using nf-core



Configuring nf-core pipelines

Adapting execution to your infrastructure

To get the workflow manager to work most efficiently, we must tell the pipeline about the infrastructure it is running on. We do this with **configs**.

You can also use these configs to create a **reusable 'settings'** file, allowing you to re-run the pipeline on different datasets.

Configs for infrastructure

Specify how your machine or HPC cluster works

- Computing resource **maximums**
- Scheduler & queues
- Job specific specifications
- Software environment
- Common resources

```
// Custom config file for ACAD ancient eDNA workshop

params {
    config_profile_contact = 'James A. Fellows Yates (@jfy133)'
    max_cpus = 16
    max_memory = '125.GB'
    max_time = 30.m
}

process {
    executor = 'slurm'

    withName: 'malt' {
      memory = '18.GB'
    }
}

singularity {
    enabled = true
    autoMounts = true
    cacheDir = '/apps/cache'
}

cleanup = true
```

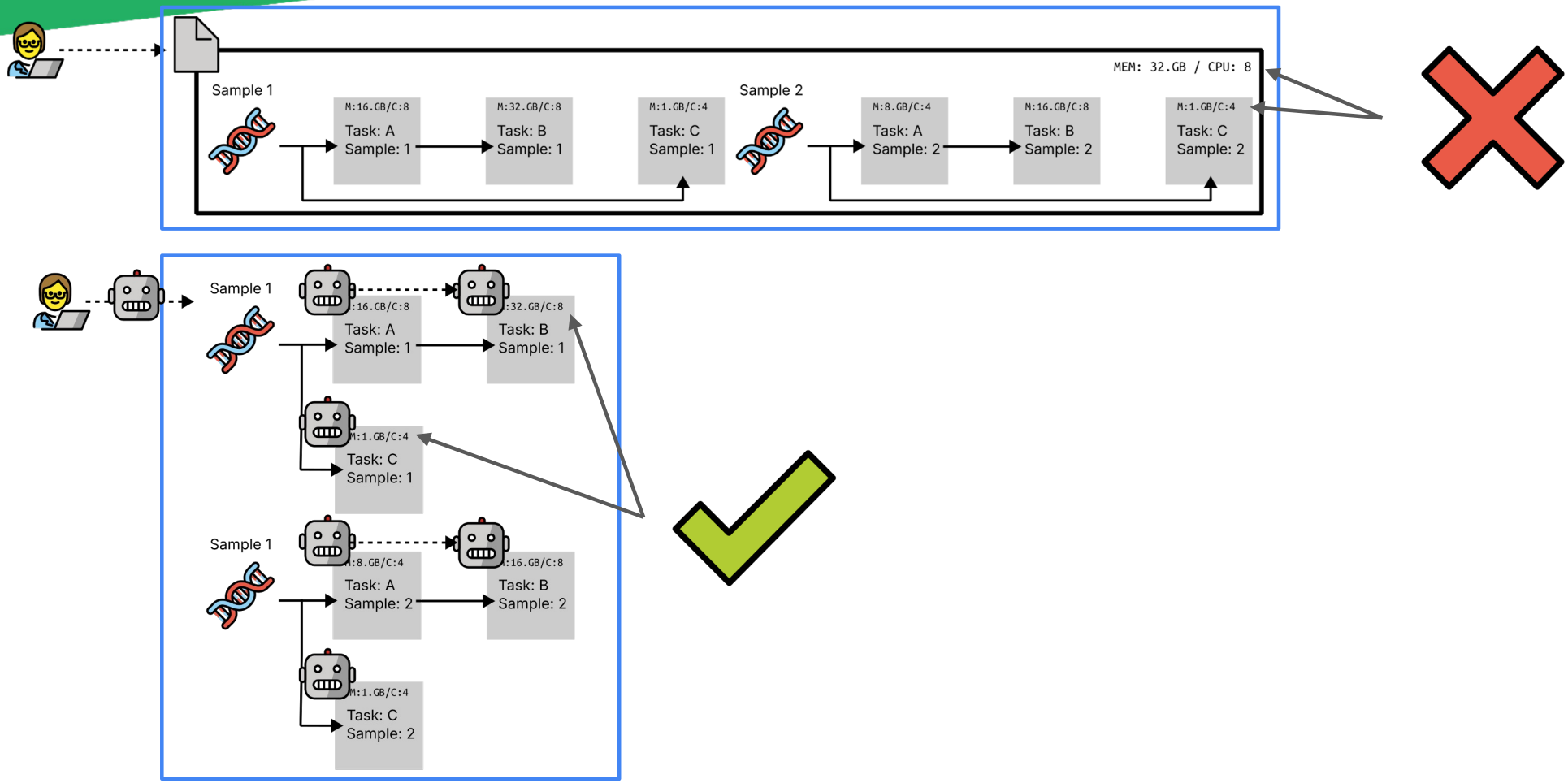
Configs for data

Optimise the run time and scheduler usage

Customise the resources used / requested of each independent **process** (a.k.a. step/job/task) of the pipeline run

```
process {

  withName: fastqc {
    memory = '1.GB'
    cpus = 2
    time = '1.h'
  }

  withName: bwa {
    memory = '32.GB'
    cpus = 16
    time = '8.h'
  }

}
```

Reminder


MEM: 32.GB / CPU: 8
Sample 1
M:16.GB/C:8
Task: A
Sample: 1
M:32.GB/C:8
Task: B
Sample: 1
M:1.GB/C:4
Task: C
Sample: 1
Sample 2
M:8.GB/C:4
Task: A
Sample: 2
M:16.GB/C:8
Task: B
Sample: 2
M:1.GB/C:4
Task: C
Sample: 2
Sample 1
Task: A
Sample: 1
Task: B
Sample: 1
Task: C
Sample: 1
Sample 1
Task: A
Sample: 2
Task: B
Sample: 2
Task: C
Sample: 2

What you specify in a config file…

```
process {

  withName: fastqc {
    memory = '1.GB'
    cpus = 2
    time = '1.h'
  }

  withName: bwa {
    memory = '32.GB'
    cpus = 16
    time = '8.h'
  }

}
```

What Nextflow does under the hood

```
(nf-core) jfellowsy@raven01: cat .command.run
#!/bin/bash
<...>
#SBATCH -c 2
#SBATCH -t 01:00:00
#SBATCH --mem 1096M
# NEXTFLOW TASK: fastqc (JK2782_L1)
<...>
singularity exec bash .command.run
<...>
```

```
(nf-core) jfellowsy@raven01: cat .command.run
#!/bin/bash
<...>
#SBATCH -c 16
#SBATCH -t 08:00:00
#SBATCH --mem 32096M
# NEXTFLOW TASK: bwa (JK2782)
<...>
singularity exec bash .command.run
<...>
```

Sharing is caring!

Configure once, run all nf-core pipelines!

Debugging nf-core pipelines

Read the error message

A **typical** Nextflow error message can give you a lot of information.

Name of part (process) of the pipeline that failed

The **Nextflow** error message!

The **bash exit code** (google this if not 0 or 255!)

The **command stdout** (what the tool prints to console)

Path to where commands were being executed

```
-[nf-core/rnaseq] Pipeline completed with errors-
ERROR ~ Error executing process > 'NFCORE_RNASEQ:RNASEQ:FASTQ_FASTQC_UMITOOLS_TRIMGALORE:TRIMGALORE (KO_REP1)'

Caused by:
  Process requirement exceeds available memory -- req: 72 GB; avail: 62.8 GB

Command executed:

  [ ! -f  KO_REP1_1.fastq.gz ] && ln -s W3_67_EKRN230070531-1A_22GTGWLT3_L4_1.fq.gz KO_REP1_1.fastq.gz
  [ ! -f  KO_REP1_2.fastq.gz ] && ln -s W3_67_EKRN230070531-1A_22GTGWLT3_L4_2.fq.gz KO_REP1_2.fastq.gz
  trim_galore \
      --fastqc_args '-t 12' \
      --cores 8 \
      --paired \
      --gzip \
      KO_REP1_1.fastq.gz \
      KO_REP1_2.fastq.gz

  cat <<-END_VERSIONS > versions.yml
  "NFCORE_RNASEQ:RNASEQ:FASTQ_FASTQC_UMITOOLS_TRIMGALORE:TRIMGALORE":
      trimgalore: $(echo $(trim_galore --version 2>&1) | sed 's/^.*version //; s/Last.*$//')
      cutadapt: $(cutadapt --version)
  END_VERSIONS

Command exit status:
  -

Command output:
  (empty)

Work dir:
  /data/RNAseq/W3_males/work/4e/2c5062870db1ce95c91c9e9d15eedc

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

 -- Check '.nextflow.log' file for details
```

Read the documentation

All nf-core pipelines come with 3 dedicated pages

- Usage
- Parameters
- Output

These are online at https://nf-co.re/<pipelinename>

The nf-core website also has general documentation including troubleshooting at https://nf-co.re/docs

Read the log files

Everytime you execute `nextflow run` a (hidden!) **.nextflow.log** file in the location you ran the pipeline!

Pipeline run failed but no error? **Inspect this file**!

This can be overwhelming(!), but can often provide more information.

```
[0a/09d560] process > NFCORE_MAG:MAG:VIRUS_IDENTIFICATION:GENOMAD_ENDTOEND (group-0)                [100%]
1 of 1, failed: 1 ✘
[31/bfd1dc] process > NFCORE_MAG:MAG:CUSTOM_DUMPSOFTWAREVERSIONS (1)                                [100%]
1 of 1 ✔
[eb/a1332d] process > NFCORE_MAG:MAG:MULTIQC                                                        [100%]
1 of 1 ✔
ERROR ~ Error executing process > 'NFCORE_MAG:MAG:VIRUS_IDENTIFICATION:GENOMAD_ENDTOEND (group-0)'

Caused by:
  /workspace/mag/work/0a/09d5603ac56e85fa66bbe6d21f021a/genomad_db


 -- Check '.nextflow.log' file for details
```

Inspecting a ‘working directory’

Remember that **‘hash’** at the beginning of each line of the console?

Represents the unique location of that task of the pipeline!

Inside has:

- Input, intermediate files
- Execution scripts (hidden!)
- Software log files
- Output files

```
/shared/jamesyates/acad-workshop/work/8c/7ec8b8202a13e13262d07cd9366498$ ll
total 556
drwxrwxr-x 2 jamesyates jamesyates   4096 Feb 12 08:30 ./
drwxrwxr-x 3 jamesyates jamesyates   4096 Feb 12 08:30 ../
-rw-rw-r-- 1 jamesyates jamesyates      0 Feb 12 08:30 .command.begin
-rw-rw-r-- 1 jamesyates jamesyates   1366 Feb 12 08:30 .command.err
-rw-rw-r-- 1 jamesyates jamesyates   1366 Feb 12 08:30 .command.log
-rw-rw-r-- 1 jamesyates jamesyates      0 Feb 12 08:30 .command.out
-rw-rw-r-- 1 jamesyates jamesyates  10532 Feb 12 08:30 .command.run
-rw-rw-r-- 1 jamesyates jamesyates    509 Feb 12 08:30 .command.sh
-rw-rw-r-- 1 jamesyates jamesyates    212 Feb 12 08:30 .command.trace
-rw-rw-r-- 1 jamesyates jamesyates      1 Feb 12 08:30 .exitcode
lrwxrwxrwx 1 jamesyates jamesyates     80 Feb 12 08:30 BWAIndex -> /shared/jamesyates/acad-
workshop/work/eb/44ce285fde0229ee6925aa91528b0e/BWAIndex/
-rw-rw-r-- 1 jamesyates jamesyates  72324 Feb 12 08:30 JK2782.sai
-rw-rw-r-- 1 jamesyates jamesyates 437388 Feb 12 08:30 JK2782_PE.mapped.bam
-rw-rw-r-- 1 jamesyates jamesyates    104 Feb 12 08:30 JK2782_PE.mapped.bam.bai
lrwxrwxrwx 1 jamesyates jamesyates    154 Feb 12 08:30
JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.combined.fq.gz -> /shared/jamesyates/acad-
workshop/work/42/b5d6b8f22ad8ff12289787713557c3/output/JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.c
ombined.fq.gz
lrwxrwxrwx 1 jamesyates jamesyates     80 Feb 12 08:30 nf-core_eager_dummy.txt ->
/shared/jamesyates/.nextflow/assets/nf-core/eager/assets/nf-core_eager_dummy.txt
(nf-core-dsl1) jamesyates@ip-10-255-2-83:/shared/jamesyates/acad-workshop/work/8c/7ec8b8202a13e13262d07cd9366498$
```

Inspecting a ‘working directory’

Exploring these files help understand how Nextflow works under the hood!

```
/shared/jamesyates/acad-workshop/work/8c/7ec8b8202a13e13262d07cd9366498$ ll
total 556
drwxrwxr-x 2 jamesyates jamesyates   4096 Feb 12 08:30 ./
drwxrwxr-x 3 jamesyates jamesyates   4096 Feb 12 08:30 ../
-rw-rw-r-- 1 jamesyates jamesyates      0 Feb 12 08:30 .command.begin
-rw-rw-r-- 1 jamesyates jamesyates   1366 Feb 12 08:30 .command.err
-rw-rw-r-- 1 jamesyates jamesyates   1366 Feb 12 08:30 .command.log
-rw-rw-r-- 1 jamesyates jamesyates      0 Feb 12 08:30 .command.out
-rw-rw-r-- 1 jamesyates jamesyates  10532 Feb 12 08:30 .command.run
-rw-rw-r-- 1 jamesyates jamesyates    509 Feb 12 08:30 .command.sh
-rw-rw-r-- 1 jamesyates jamesyates    212 Feb 12 08:30 .command.trace
-rw-rw-r-- 1 jamesyates jamesyates      1 Feb 12 08:30 .exitcode
lrwxrwxrwx 1 jamesyates jamesyates     80 Feb 12 08:30 BWAIndex -> /shared/jamesyates/acad-
workshop/work/eb/44ce285fde0229ee6925aa91528b0e/BWAIndex/
-rw-rw-r-- 1 jamesyates jamesyates  72324 Feb 12 08:30 JK2782.sai
-rw-rw-r-- 1 jamesyates jamesyates 437388 Feb 12 08:30 JK2782_PE.mapped.bam
-rw-rw-r-- 1 jamesyates jamesyates    104 Feb 12 08:30 JK2782_PE.mapped.bam.bai
lrwxrwxrwx 1 jamesyates jamesyates    154 Feb 12 08:30
JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.combined.fq.gz -> /shared/jamesyates/acad-
workshop/work/42/b5d6b8f22ad8ff12289787713557c3/output/JK2782_TGGCCGATCAACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.c
ombined.fq.gz
lrwxrwxrwx 1 jamesyates jamesyates     80 Feb 12 08:30 nf-core_eager_dummy.txt ->
/shared/jamesyates/.nextflow/assets/nf-core/eager/assets/nf-core_eager_dummy.txt
```

Ignore

STDERR of commands only

Combined **STDOUT and STDERR** of commands

STDOUT of commands only

The script submitted to the cluster

The script containing the command(s)

Ignore

The final bash exit code of the command

A symlinked input file

Output file

Try exploring a work directory yourself!

Report the error

Essential

- Command you used
- Version
- Error message
- Log files

Nice to have

- Input samplesheet

Recap: Session Objectives

To learn

- ✅ What are and the benefits of workflow managers
- ✅ What is nf-core
- ✅ What is and how to install conda
- ✅ How to install and conda correctly
- ✅ How to install Nextflow
- ✅ How to run and monitor an nf-core pipeline
- ✅ How to configure an nf-core pipeline
- ✅ How to debug an nf-core pipeline

Next session

Applying our new running Nextflow pipeline skills to taxonomic classification and profiling nf-core pipeline **nf-core/taxprofiler** It allows for in-parallel taxonomic identification of reads or taxonomic abundance estimation with multiple classification and profiling tools against multiple databases

More Info & Acknowledgements


https://nf-co.re


github.com/nf-core


nfcore.slack.com


youtube.com/nf-core


@nf_core@mstdn.science


@nf_core

Financial & Infrastructure Support:

Chan Zuckerberg Initiative

seqera

nextflow AMBASSADOR

aws

Gitpod

HackMD

Speaker's Support

nf-core/core & community

Microbiome Sciences Group @ MAX-PLANCK-INSTITUT FÜR EVOLUTIONÄRE ANTHROPOLOGIE

LEIBNIZ-HKI

WSS WERNER SIEMENS-STIFTUNG

ACAD

THE UNIVERSITY of ADELAIDE Environment Institute