

nf-core



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

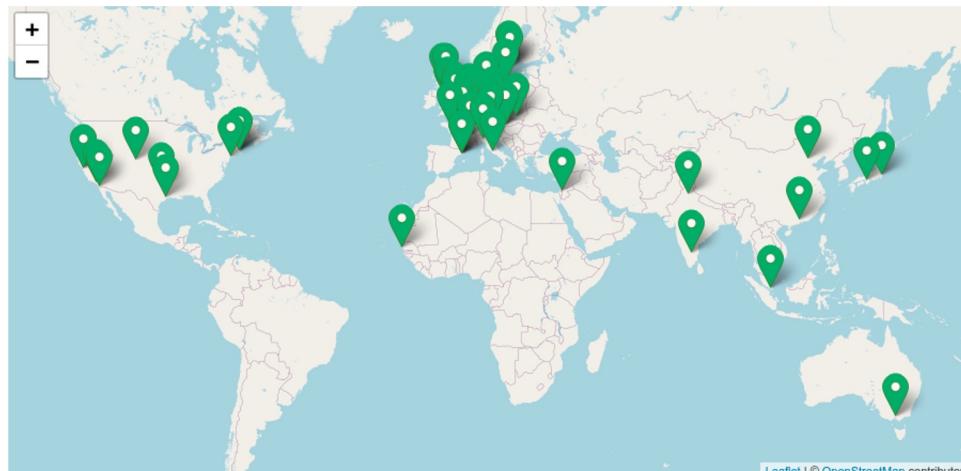
nf-core in numbers



nf-core in numbers

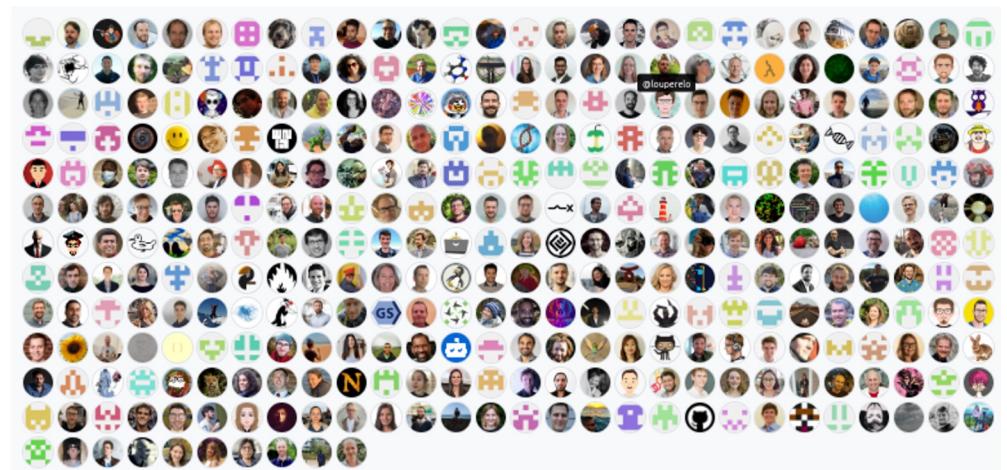
>11K

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

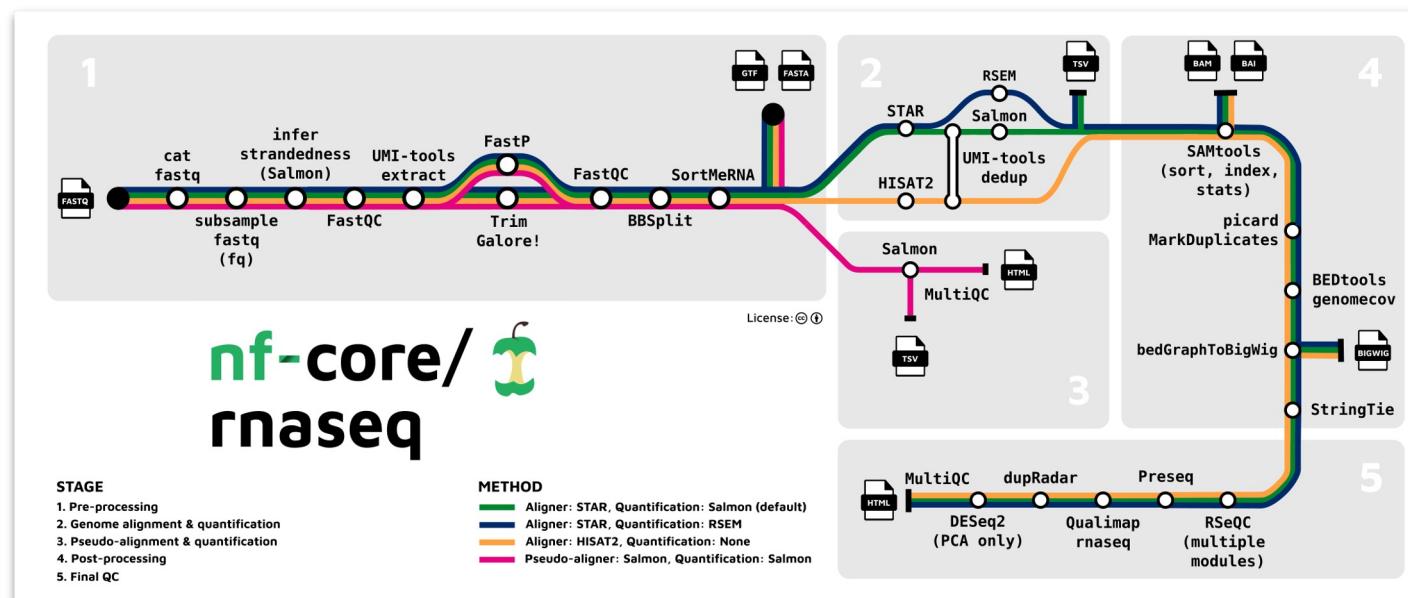
Finding Software

- 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)

The screenshot shows a web-based interface for managing AWS S3 buckets. At the top, there's a navigation bar with links: 'Introduction', 'aws Results' (which is highlighted in green), 'Usage docs', 'Parameters', 'Output docs', and 'Releases & Statistics'. Below the navigation is a dropdown menu showing '3.1.2'. The main area displays an AWS S3 file browser. The path shown is 'nf-core-awsmegatests / sarek / results-c87f4eb694a7183e4f99c70fca0f1d4e91750b33 / germline_test / preprocessing'. On the right side of this path, there are two buttons: 'Copy Bucket' and 'S3 URL'. The file browser table has columns for 'Name', 'Last Modified', and 'Size'. The data in the table is as follows:

Name	Last Modified	Size
..		
markduplicates/		
recal_table/		
recalibrated/		

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 command-line flags

General Nextflow flags to control how the pipeline runs.

These are not specific to the pipeline and will not be saved in any parameter file. They are just used when building the 'nextflow run' launch command.

-name Unique name for this nextflow run

-profile Configuration profile

Launch

On this page

- Nextflow command-line flags
- > Input/output options
- > Main options
- > Preprocessing
- > Variant Calling
- > Reference genome options
- > Generic options

Show hidden params

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

Nextflow command-line flags

General Nextflow flags to control how the pipeline runs.

These are not specific to the pipeline and will not be saved in any parameter file. They are just used when building the 'nextflow run' launch command.

-name Unique name for this nextflow run

-profile Configuration profile

Launch

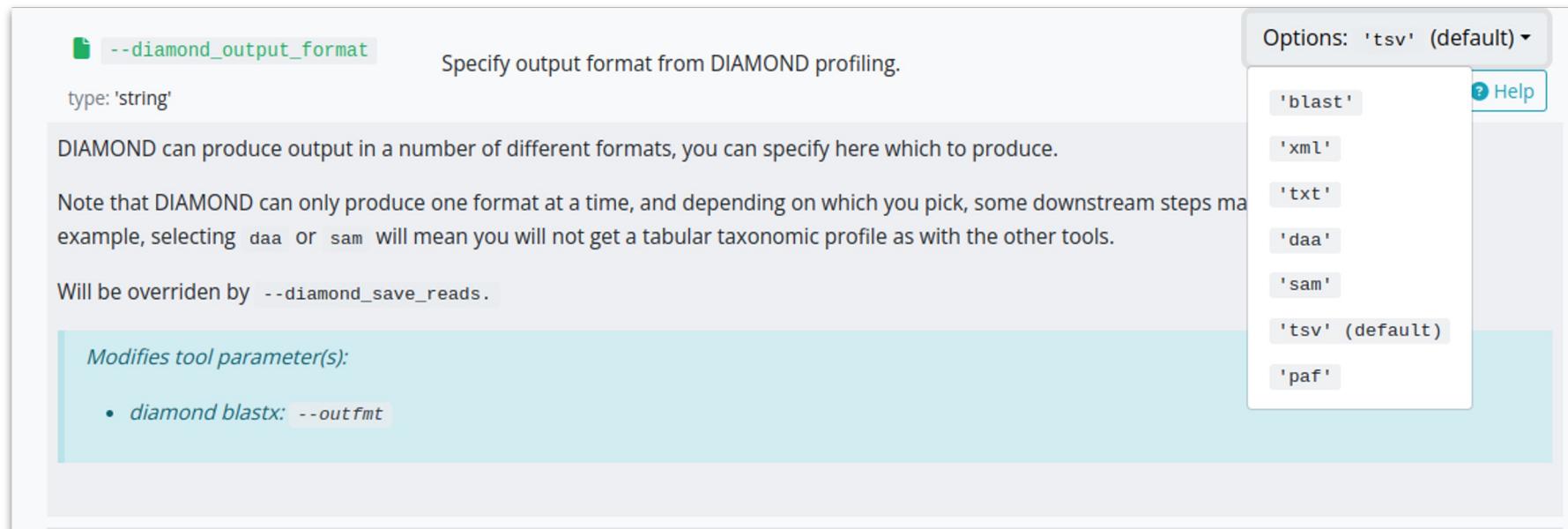
On this page

- Nextflow command-line flags
- > Input/output options
- > Main options
- > Preprocessing
- > Variant Calling
- > Reference genome options
- > Generic options

Show hidden params

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

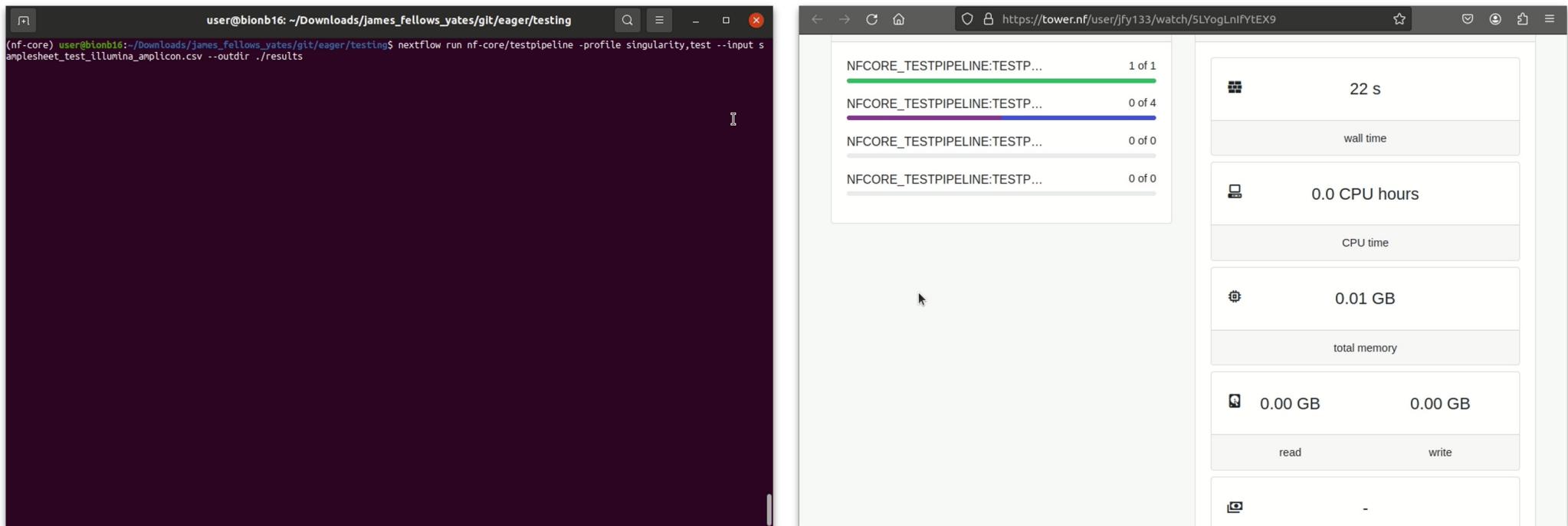
Documentation (Input)



- 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 is 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`.

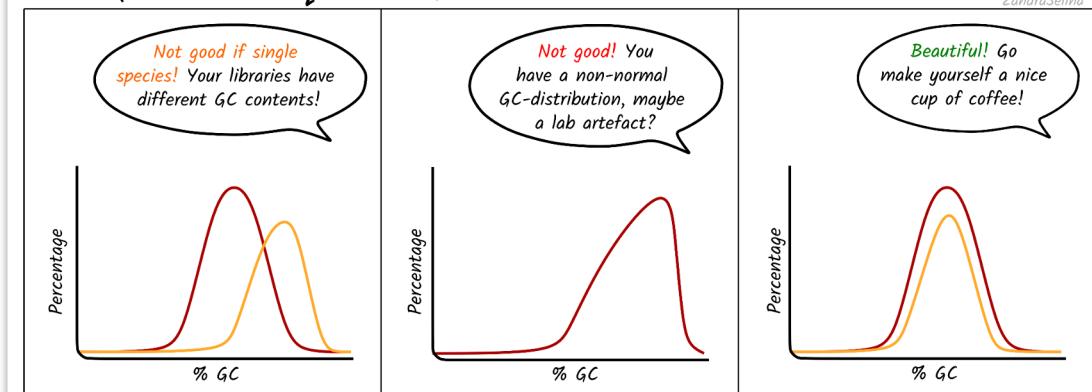
 We do not recommend using Porechop if you are already trimming the adapters with ONT's basecaller Guppy.

Descriptions of every output file

Additional guidance on interpretation of output

(not all pipelines)

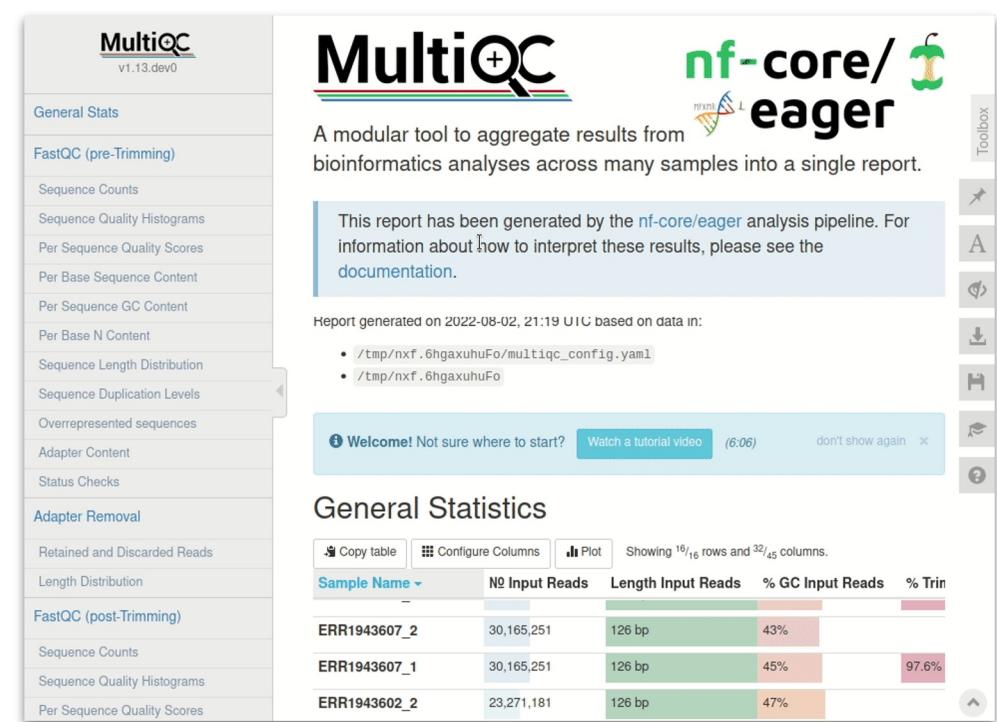
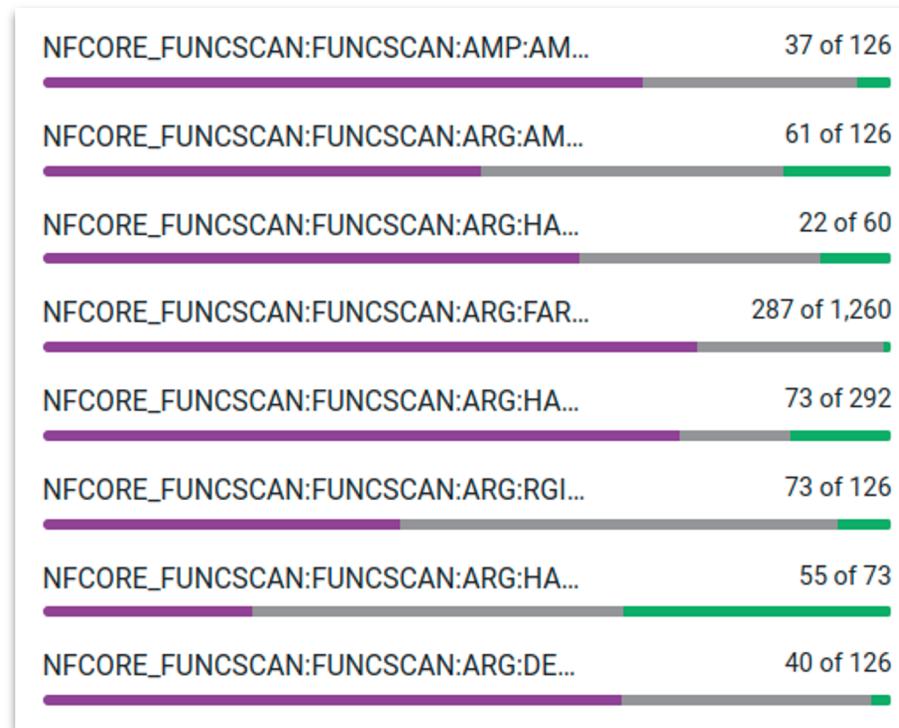
FASTQC - Per sequence GC-content



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**

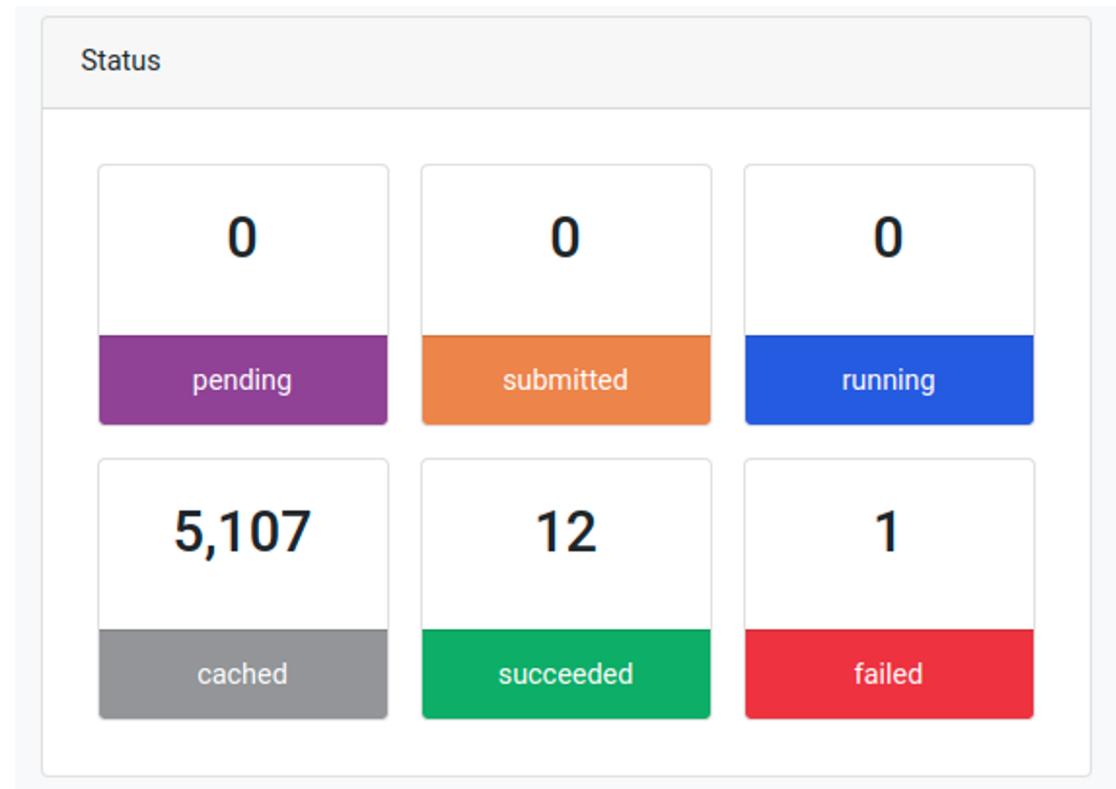


```
$ nextflow run nf-core/rnaseq -profile eva <...>
$ nextflow run nf-core/rnaseq -profile mpcdf,raven <...>
$ nextflow run nf-core/rnaseq -profile binac <...>
$ nextflow run nf-core/rnaseq -profile awsbatch <...>
```

Many institutional clusters/HPC already
supported by 
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

CONDA®

OR



nextflow

```
$ conda create -n nextflow -c bioconda nextflow  
$ conda activate nextflow  
$ nextflow run <...> -profile conda
```

CONDA®



N
SHIFTER

S
INGULARITYCE

A
APPTAINER



podman

Consistent (Execution)

Reproducible

- Strict **versioning**

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

Versions	
Version 2.3.0	Mar 2, 2023
10.5281/zenodo.7691467	
Version 2.2.1	Aug 25, 2022
10.5281/zenodo.7022711	
Version 2.2.0	Jun 14, 2022
10.5281/zenodo.6641020	
Version 2.1.0	Jul 29, 2021
10.5281/zenodo.5144737	
Version 2.0.0	Jun 1, 2021
10.5281/zenodo.4889751	
View all 10 versions	
Cite all versions? You can cite all versions by using the DOI 10.5281/zenodo.3589527. This DOI represents all versions, and will always resolve to the latest one. Read more.	

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



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

nf-core/ 
fetchngs

Modern metagenomics

nf-core/ 
taxprofiler

nf-core/ 

funcscan

nf-core/ 
ampliseq

nf-core/ 
differentialabundance

Microbial metagenomics related

nf-core/ 
bacass

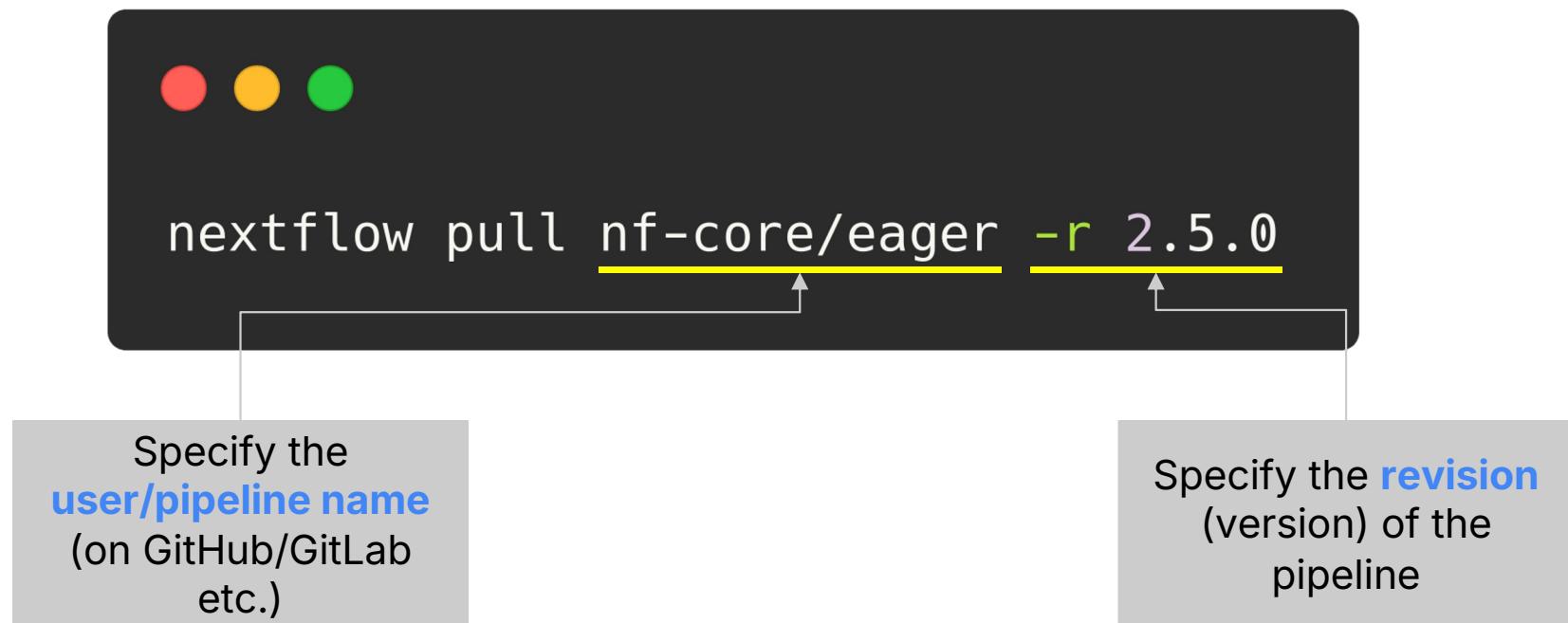
nf-core/ 
bactmap

nf-core/ 
viralrecon

General utility

Installing nf-core pipelines

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



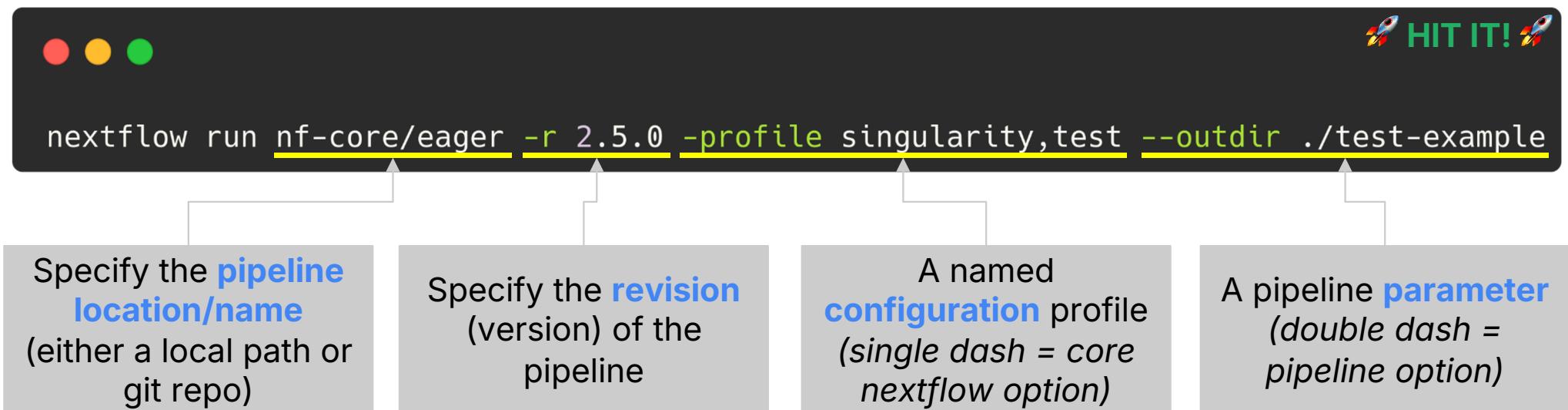
Running and monitoring nf-core pipelines

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> ls
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!

Misincorporation Plots

The MultiQC DamageProfiler and mapDamage module misincorporation plots shows the percent frequency (Y axis) of C to T mismatches at 5' read ends and complementary G to A mismatches at the 3' ends. The X axis represents base pairs from the end of the molecule from the given prime end, going into the middle of the molecule i.e. 1st base of molecule, 2nd base of molecule etc until the 14th base pair. The mismatches are when compared to the base of the reference genome at that position.

When looking at the misincorporation plots, keep the following in mind:

- As few-base single-stranded overhangs are more likely to occur than long overhangs, we expect to see a gradual decrease in the frequency of the modifications from position 1 to the inside of the reads.
- If your library has been **partially-UDG treated**, only the first one or two bases will display the misincorporation frequency.
- If your library has been **UDG treated** you will expect to see extremely-low to no misincorporations at read ends.
- If your library is **single-stranded**, you will expect to see only C to T misincorporations at both 5' and 3' ends of the fragments.
- We generally expect that the older the sample, or the less-ideal preservational environment (hot/wet) the greater the frequency of C to G/A.
- The curve will be not smooth then you have few reads informing the frequency calculation. Read counts of less than 500 are likely not reliable.
- If the `mapdamage_downsample` parameter was specified and mapDamage was used for damage calculation, the damage frequency for each base is based only on the specified number of reads.

DamageProfiler

The figure consists of six small plots arranged in a 2x3 grid, each with 'Frequency' on the Y-axis and 'Base' on the X-axis. Each plot contains a red line representing the frequency of a specific base pair across the sequence. Callouts provide context for each plot:

- Top-left: "If you have UDG-treated DNA, it should be flat"
- Top-middle: "If you have very few reads (<500), it might be ok, or you have high levels of sequencing error"
- Top-right: "Very odd if you have mapped to a wrong (but related) reference genome"
- Bottom-left: "This is ok, if your DNA is partially UDG-treated"
- Bottom-middle: "This is ok, if you did single stranded library preparation"
- Bottom-right: "Bamboo! Go waste some time on your laptop!"

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.

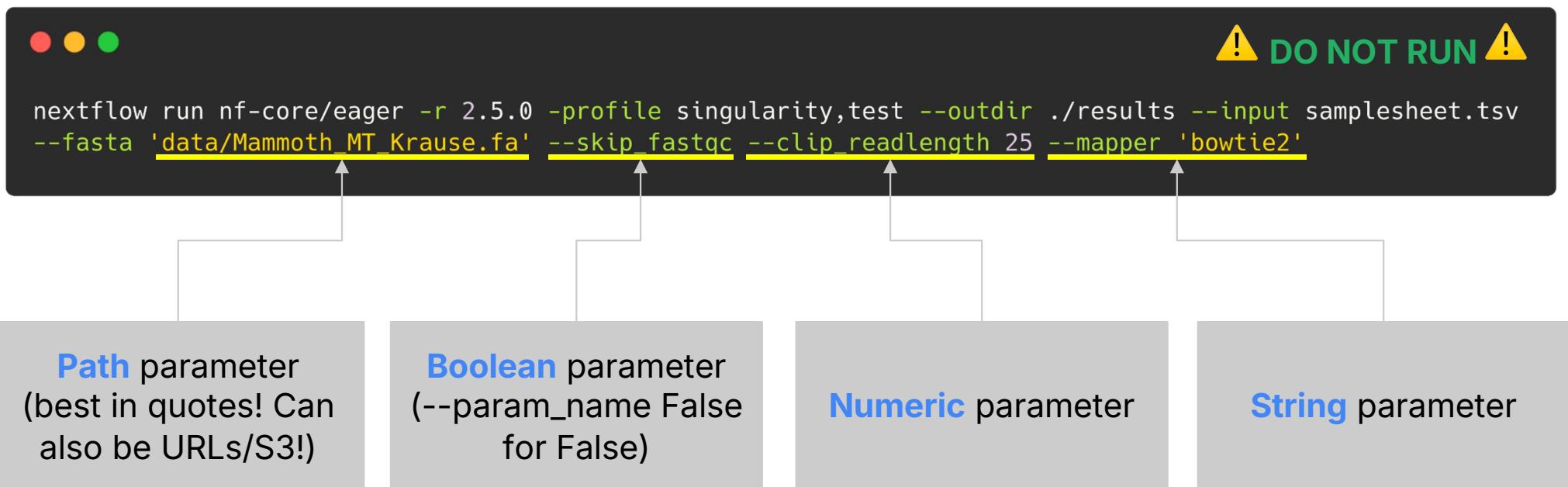
A more realistic run command

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

Sample_Name	Library_ID	Lane	Colour_Chemistry	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.gz	NA
2612	ERR5766179	0	4	PE	Homo sapiens	single	none	ERR5766179_1.fastq.gz	ERR5766179_2.fastq.gz	NA
2612	ERR5766180	0	4	PE	Homo sapiens	single	none	ERR5766180_1.fastq.gz	ERR5766180_2.fastq.gz	NA

A more realistic run command

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



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

The screenshot shows the nf-core pipeline launch interface. At the top, there's a navigation bar with links for Home, Pipelines, Modules, Tools, Docs, Events, About, and a prominent 'Save parameters and copy command to launch' button. A green 'Join nf-core' button is also visible.

Launch parameters saved

Your workflow parameters are ready to go! Follow the instructions below for instructions on how to launch your pipeline:

If your system has an internet connection

The easiest way to launch this workflow is by using the `nf-core/tools` helper package.

Once installed ([see documentation](#)), simply run the following command and follow the prompts:

```
nf-core launch --id 1705211127_82afae3536c1
```

Launch using Nextflow Tower

Clicking the button below will take you to the Nextflow Tower launch page with all parameters set, ready for launch (requires a Nextflow Tower account).

[Nextflow Tower > Launch](#)

Launching with no internet and without nf-core/tools

You can run this pipeline with just Nextflow installed by copying the JSON below to a file called `nf-params.json`:

```
{  
    "input": "input.csv",  
    "databases": "database.csv",  
    "outdir": "./results"  
}
```

Then, launch Nextflow with the following command:

```
nextflow run nf-core/taxprofiler -r 1.1.3 -profile docker,test -params-file nf-params.json
```

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
```

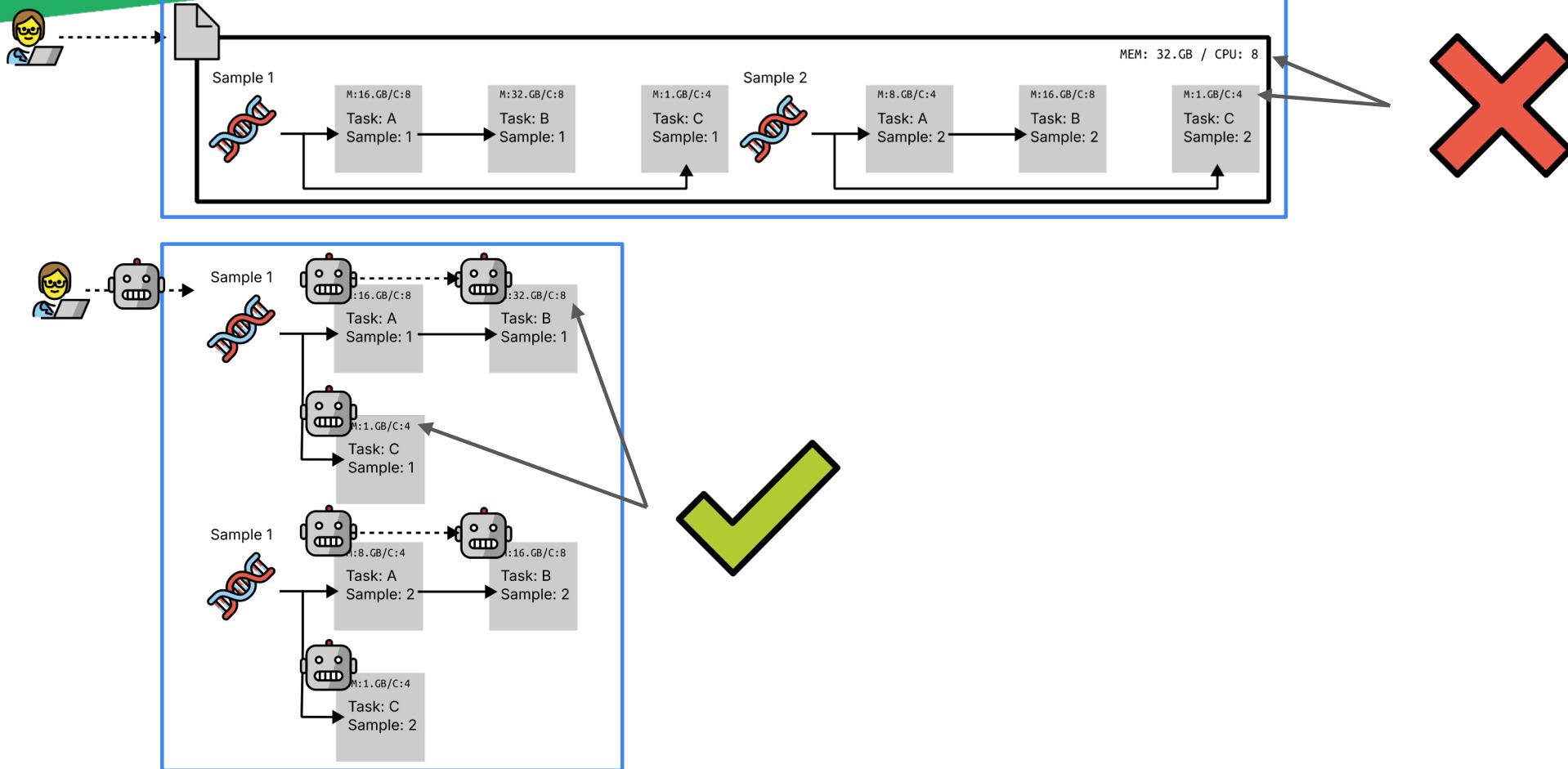
Configs for data

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



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!

nf-core/ 
configs

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_22GTGWL3_L4_1.fq.gz KO_REP1_1.fastq.gz
[ ! -f KO_REP1_2.fastq.gz ] && ln -s W3_67_EKRN230070531-1A_22GTGWL3_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_DUMP SOFTWARE VERSIONS (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_TGGCCGATCACGA_L008_R1_001.fastq.gz.tengrand.fq_L1.pe.combined.fq.gz -> /shared/jamesyates/acad-
workshop/work/42/b5d6b8f22ad8ff12289787713557c3/output/JK2782_TGGCCGATCACGA_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 → Ignore
-rw-rw-r-- 1 jamesyates jamesyates 1366 Feb 12 08:30 .command.err → Ignore
-rw-rw-r-- 1 jamesyates jamesyates 1366 Feb 12 08:30 .command.log → Combined STDOUT and STDERR of commands
-rw-rw-r-- 1 jamesyates jamesyates 0 Feb 12 08:30 .command.out → STDOUT of commands only
-rw-rw-r-- 1 jamesyates jamesyates 10532 Feb 12 08:30 .command.run → The script submitted to the cluster
-rw-rw-r-- 1 jamesyates jamesyates 509 Feb 12 08:30 .command.sh → The script containing the command(s)
-rw-rw-r-- 1 jamesyates jamesyates 212 Feb 12 08:30 .command.trace → Ignore
-rw-rw-r-- 1 jamesyates jamesyates 1 Feb 12 08:30 .exitcode → The final bash exit code of the command
lrwxrwxrwx 1 jamesyates jamesyates 80 Feb 12 08:30 BWAIndex -> /shared/jamesyates/acad- → A symlinked input file
workshop/work/eb/44ce285fde0229ee6925aa91528b0e/BWAIndex/
-rw-rw-r-- 1 jamesyates jamesyates 72324 Feb 12 08:30 JK2782.sai → Output file
-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.combined.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

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>



youtube.com/nf-core



github.com/nf-core



@nf_core@mstdn.science



nfcore.slack.com



@nf_core

Some emojis designed by [OpenMoji](#) – the open-source emoji and icon project. License: [CC BY-SA 4.0](#)

Financial &
Infrastructure
Support:

Chan
Zuckerberg
Initiative

AMBASSADOR



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