nf-core

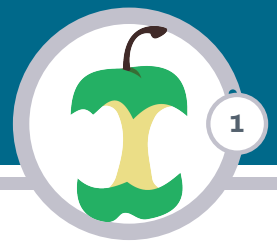
A community effort to collect a curated set of analysis pipelines built using Nextflow.

Presentation inspired from the nf-core documentation

Nicolas Fontrodona

February 1, 2022

Table of contents



What is nf-core ?

A community effort to collect analysis pipelines built with Nextflow that follow strict guidelines.

What is nf-core ?

A community effort to collect analysis pipelines built with Nextflow that follow strict guidelines.

Nextflow is a workflow manager able to manage the parallelisation of complex pipelines

What is nf-core ?

A community effort to collect analysis pipelines built with Nextflow that follow strict guidelines.

Nextflow is a workflow manager able to manage the parallelisation of complex pipelines

Nf-core as three targets:

- Facilities: Automated and efficient pipelines with reproducibility of results
- Single users: Documented, portable, and easy to use pipelines
- Developers: Write a nextflow pipeline using templates and helper tools

What is nf-core ?

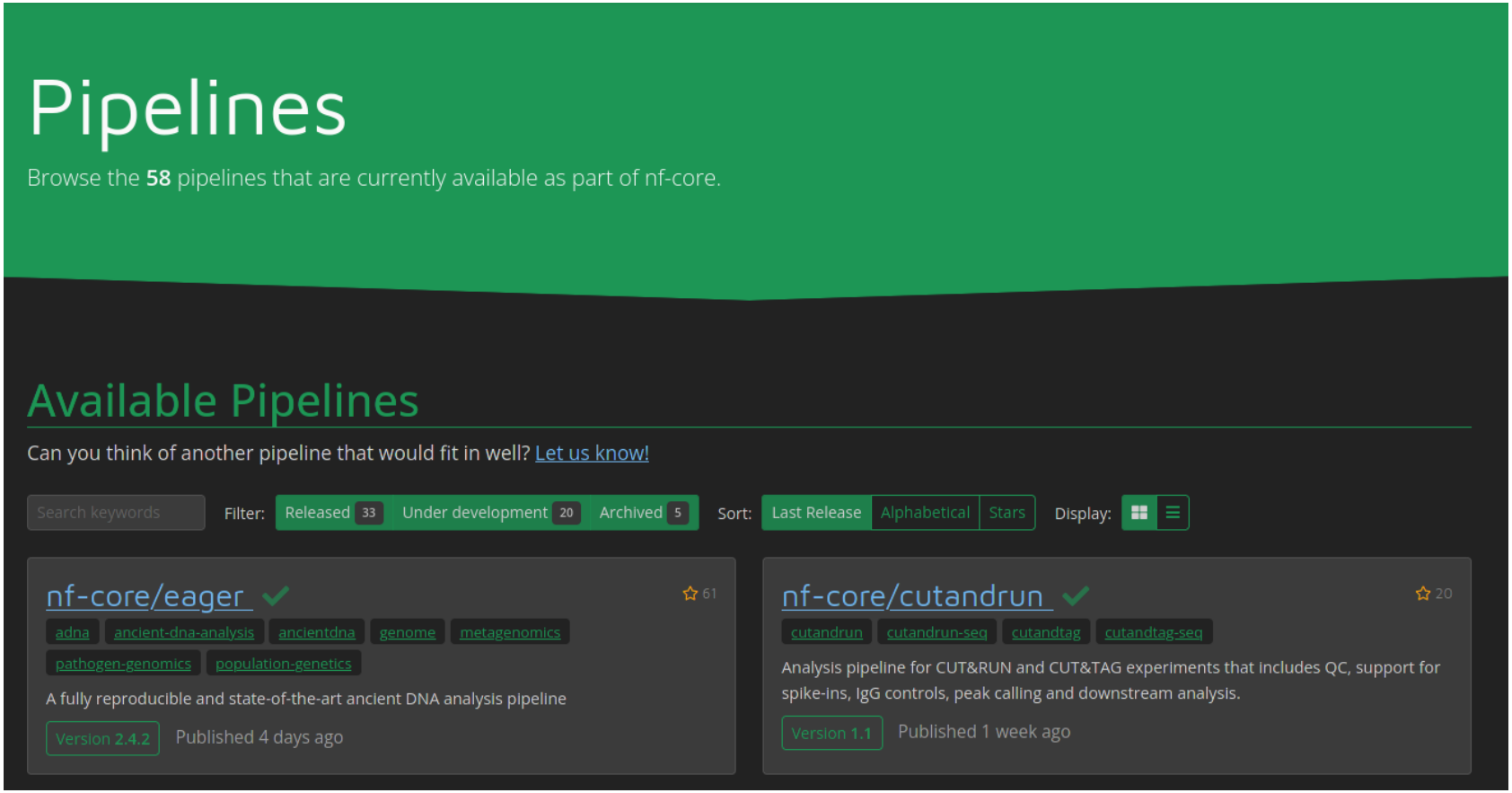
The nf-core community develops piplines for various bioinformatics analyzes that adhere to strict gidelines[1]:

- Documentation
- CI-testing
- Stable release
- Packaged processes
- Portable and reproducible

[1] Source: https://nf-co.re/

nf-core pipelines

The nf-core pipelines are listed here: https://nf-co.re/pipelines:

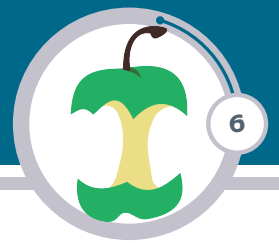


Launching a nf-core pipeline

Prerequisites to launch a nf-core pipeline:

- Nextflow
- A software packaging tool (that can be managed by Nextflow)
  - Pipeline sofwares don't need to be installed on your computer
  - They are packaged inside containers (or environements) and downloaded when the pipeline needs them
  - Key step for reproducibility
  - All nf-core pipelines can work with Docker and Singularity and most of them have support for 'Conda'

Launching a nf-core pipeline



Launching the Rnaseq pipeline on test data:

```
$ # Install nextflow
$ curl -s https://get.nextflow.io | bash
$ mkdir bin; mv nextflow ./bin/
$ # Launching a rnaseq pipeline with test inputs
$ bin/nextflow run nf-core/rnaseq -profile test,singularity
N E X T F L O W  ~  version 21.10.6
Launching `nf-core/rnaseq` [festering_fermi] - revision: 646723c70f
    [master]
...
[c7/8318de] process > NFCORE_RNASEQ:RNASEQ:MULTIQC (1)

    [100%] 1 of 1
Pulling Singularity image https://depot.galaxyproject.org/
    singularity/multiqc:1.11--pyhdfd78af_0 [cache /media/Data/
    Projects/cours/nf-core/work/singularity/depot.galaxyproject.org-
    singularity-multiqc-1.11--pyhdfd78af_0.img]
...
-[nf-core/rnaseq] Pipeline completed successfully-
...
Completed at: 28-janv.-2022 17:53:44
Duration    : 17m 58s
CPU hours   : 0.4
Succeeded   : 202
```

Launching a nf-core pipeline
Additional data

By default the command `nextflow run mypipe [options]` runs the latest version of the pipeline mypipe.

To run a particular version of the pipeline you can use the `-r/-revision` flag. For example with the pipeline rnaseq:

```
bin/nextflow run nf-core/rnaseq -profile test,singularity -r 3.4
```

Launching a nf-core pipeline
Additional data

Also note that the code of the pipeline is not downloaded in your working directory:

```
$ bin/nextflow list
nf-core/rnaseq
$ bin/nextflow info nf-core/rnaseq
project name: nf-core/rnaseq
repository  : https://github.com/nf-core/rnaseq
local path  : /home/nicolas/.nextflow/assets/nf-core/rnaseq
main script : main.nf
description : Nextflow RNA-Seq analysis pipeline, part of the nf-
    core community.
author      : Phil Ewels, Rickard Hammaren
```

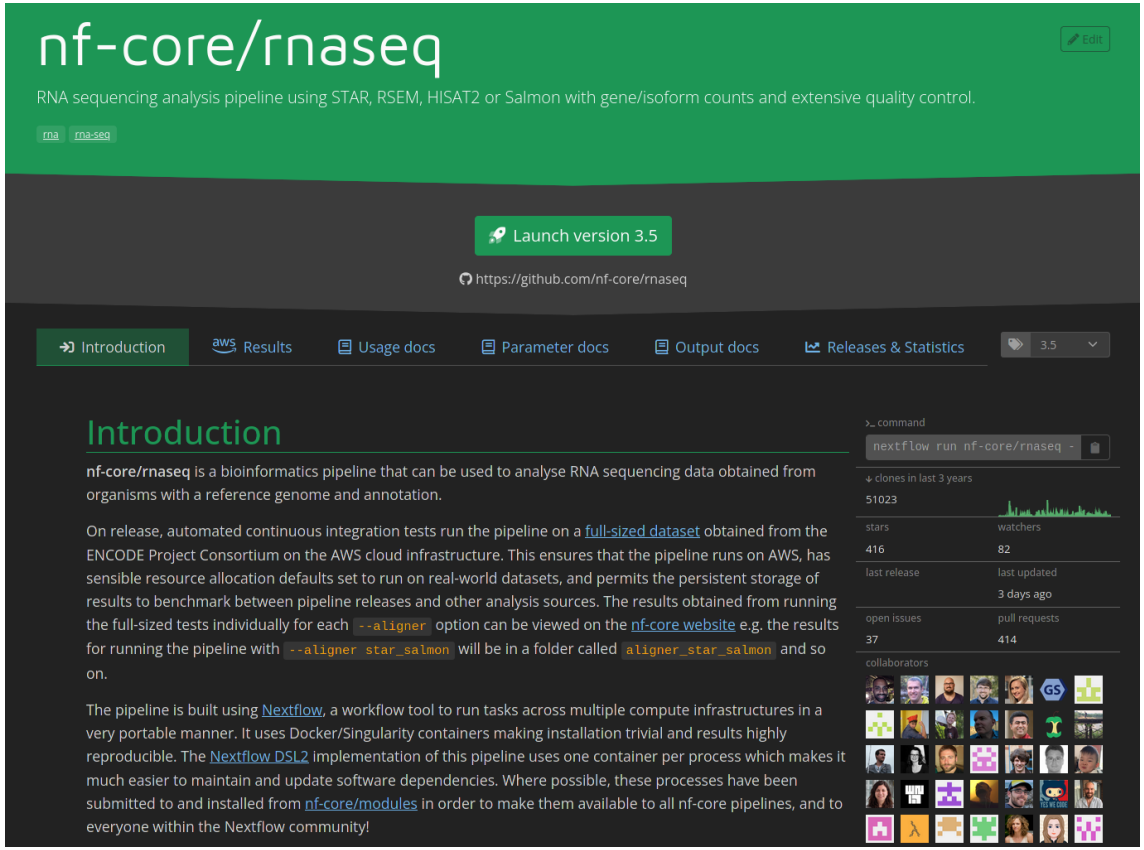
To do so you can use :

```
bin/nextflow clone nf-core/rnaseq -r 3.4
```

Additional data about nextflow's command line interface can be found here

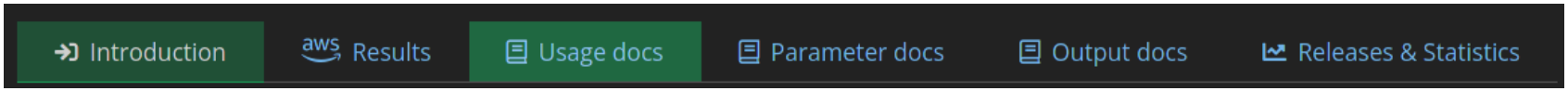
Documentation of nf-core pipelines

The documentation of a nf-core pipeline can be easily found by going on this url ‘https://nf-co.re/pipelines‘ and clicking on a pipeline name.



Documentation of nf-core/rnaseq pipeline

Documentation of nf-core pipelines



- Introduction: Describes how the program works and what are its steps
- Usage docs: Describes how to use the pipeline
- Parameters docs: Documentation for all available parameters
- Output docs: Describes the output produced by the pipeline

You can display the parameters doc with the following command:

```
$ bin/nextflow run nf-core/rnaseq  -r 3.5 --help
```

nf-core helper tools

To ease the management of nf-core pipelines the nf-core team has created the Helper Tools. They can help you to:

- List available pipelines and versions
- Run a pipeline with interactive parameter prompt
- View software licences in a pipeline
  `nf-core licences PIPELINE_NAME`
- Create a pipeline that follow the nf-core best practices
- And more...

Let's review some commands of these tools

To learn more about these tools, check the Helper Tools documentation

nf-core helper tools
installation

Three possible kind of installations:

- conda

```
$ # installation with conda
$ conda install nf-core
$ # create an environnment with nf-core/tools and nextflow
$ conda create --name nf-core python=3 nf-core nextflow
$ # Activate the new environment to run nf-core tools commands
$ conda activate nf-core
```

- python package index

```
$ pip install nf-core
```

- Docker

```
$ # Display the help of nf-core/tools where PATH is a folder
    that will contain .nfcore and .nextflow cache directories
$ docker run -it -v $PWD:$PWD -w $PWD -u $(id -u):$(id -g) --rm
    -e HOME=$PATH nfcore/tools
$ # Creating an alias
$ alias nfcore="docker run -it -v $PWD:$PWD -w $PWD -u $(id -u):
    $(id -g) --rm -e HOME=$PWD nfcore/tools"
$ nfcore --help
```

nf-core helper tools

Listing pipelines


```
$ nfcore list | head -n 50
|-----------------------------------------------------------------------|
| Pipeline     | Stars |      Latest |    Released | Last Pulled |
| Name         |       |     Release |             |             |
|--------------|-------|-------------|-------------|-------------|
| eager        |    61 |       2.4.2 |  6 days ago |           - |
|--------------|-------|-------------|-------------|-------------|
| eager        |    61 |       2.4.2 |  6 days ago |           - |
| cutandrun    |    20 |         1.1 | 1 weeks ago |           - |
| mhcquant     |    19 |       2.2.0 | 2 weeks ago |           - |
$ # You can display only the pipelines with a given list of keywords
    in title description and topics
$ nf-core list rna-seq rnaseq
| Pipeline Name | Stars | Latest Release |      Released |
|---------------|-------|----------------|---------------|
| rnaseq        |   417 |            3.5 |  1 months ago |
| smrnaseq      |    35 |          1.1.0 |  8 months ago |
| dualrnaseq    |     5 |          1.0.0 | 12 months ago |
| scflow        |    12 |            dev |             - |
|---------------|-------|----------------|---------------|
$ # The pipelines are sorted by the column Released by default. You
    can change that
$ nf-core list -s pulled # sort by when you last pulled a local copy
$ nf-core list -s name # sort by name
$ $ nf-core list -s stars # sort by github stars
```

nf-core helper tools

Launch a pipeline

Nf-core pipelines can have a lot of parameters. Nf-core helper tools help you to launch a pipeline and chose its parameters by using either a web-based graphical interface or an interactive command-line wizard tool.

1. Those tools display the documentation for each parameters
2. They validate your inputs
3. They save out parameters into a file named `nf-params.json` and used by nextflow with the flag `-params-file`

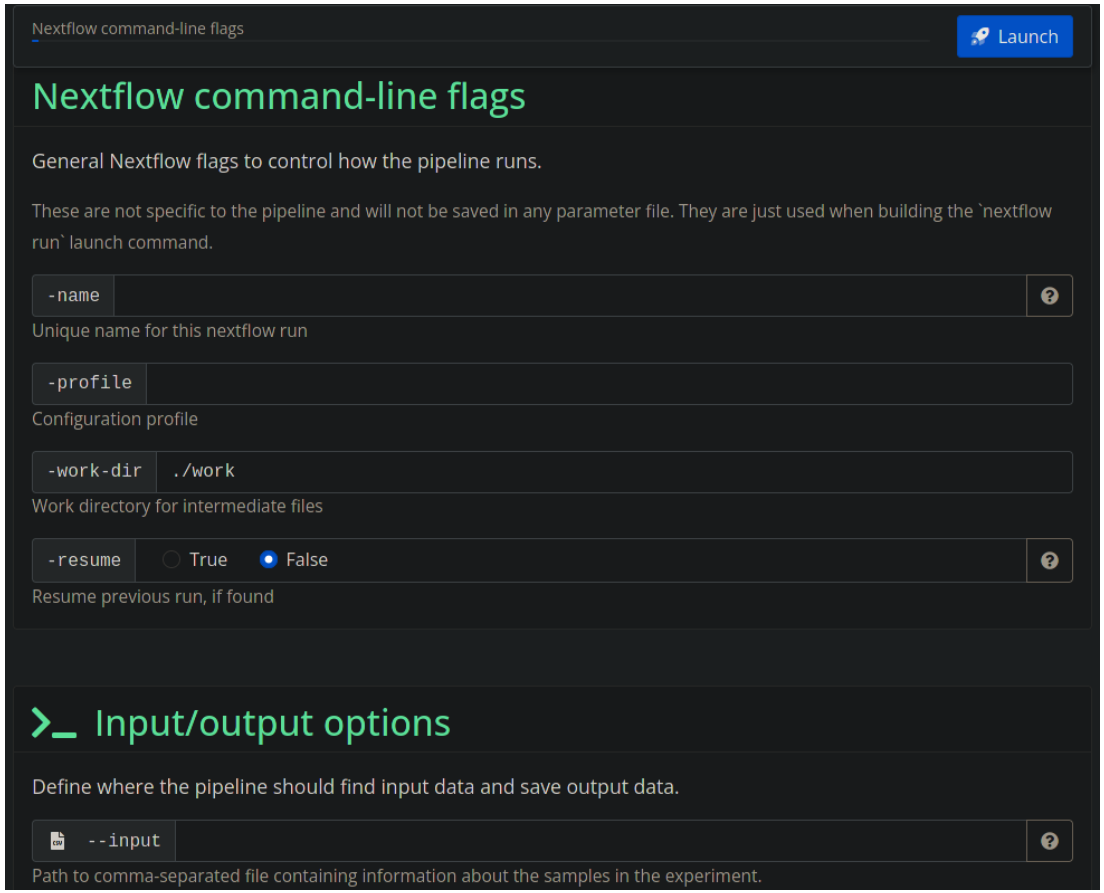
Using the wizard tool:

```
$ nf-core launch rnaseq
...
? Select release / branch: 3.5  [release]
?  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.
(Use arrow keys)
Continue >>
...
?  Input/output options
Define where the pipeline should find input data and save output
    data.
(Use arrow keys)
input
...
?  --input
Path to comma-separated file containing information about the
    samples in the experiment.
You will need to create a design file with information about the...

?
```

Using the web interface:

```
$ nf-core launch rnaseq
...
? Select release / branch: 3.5  [release]
? Choose launch method Web based
```



Creating a nf-core pipeline
Guidelines

If you want to create a nf-core pipeline it has to follow some guidelines:

- Workflow specificity
- Workflow size
- Use the template `nf-core create`
- Built with Nextflow
- MIT Licence
- Software packaged with docker
- Continuous integration testing
- Stable release tags
- Common pipeline structure and usage (Standard file, folder, and parameters names)
- A unique workflow that runs in a single command (Not multiple separate workflows)
- Great documentation
- Contact to the person responsible for the pipepline
- No failure with `nf-core lint` tests. Checks for all nf-core community guidelines
- Workflow name in lowercase without ponctuation
- Credits and Acknowledgements
- Be in touch with the community
- Other recommended features

Check this page for more details

nf-core pipelines on the PSMN

Example with the rnaseq pipeline:

```
$ curl -s https://get.nextflow.io | bash # installation of nextflow
$ mkdir bin; mv nextflow ./bin/ # moving it into ./bin
$ bin/nextflow clone nf-core/rnaseq -r 3.5 .# clonning the pipeline
```

Now, we need to define a config file to run the pipeline on the PSMN. Let's create a config file `psmn.config` into the `rnaseq/conf` folder.

Note

The config files of nf-core pipelines are always located into the directory `pipeline_name/conf` (exept for the main config file `nextflow.config` located in the root directory of the pipeline).

nf-core pipelines on the PSMN

conf/psmn.config

```
singularity.enabled = true
singularity.cacheDir = "./bin"
singularity.runOptions = "--bind /Xnfs,/scratch"
errorStrategy = 'retry'
maxRetries = 3
params.singularity_pull_docker_container = true // build images from docker containers
process{
  ext.singularity_pull_docker_container = true // build images from docker containers
  withLabel: process_high {
    // see next slide
  }
  withLabel: process_medium {
    // see next slide
  }
  withLabel: 'process_low|process_long' {
    // see next slide
  }
}
```

nf-core pipelines on the PSMN


```
 withLabel: process_high {
  executor = "sge"
  clusterOptions = "-cwd -V"
  memory = "150GB"
  cpus = 32
  time = "24h"
  queue = "CLG6242deb384*,CLG5218deb192*,CLG6226Rdeb192*"
  penv = "openmp32"
}
withLabel: process_medium {
  executor = "sge"
  clusterOptions = "-cwd -V"
  memory = "20GB"
  cpus = 8
  time = "15h"
  queue = "CLG6242deb384C,CLG6226Rdeb192D,CLG5218deb192D,
    SLG6142deb384C"
  penv = "openmp8"
}
withLabel: 'process_low|process_long' {
  executor = "sge"
  clusterOptions = "-cwd -V"
  memory = "16GB"
  cpus = 1
  time = "30h"
  queue = "monointeldeb128"
}
```

nf-core pipelines on the PSMN

To run a test on the psmn, it's highly advisable to set the `queue` and `cpus` parameters to `monointeldeb128` and `1` respectively for all processes. Remove also the `penv` parameter. Let's create another file `psmn_test.config` dedicated to test the pipeline. This file has the following content:

```
 //--snip (same thing as the file psmn.config)
 withLabel: process_high {
  //--snip
  memory = "16GB" // reduce the available memory for the process
  cpus = 1
  time = "2h" // You can also set the maximum duration to 2h
  queue = "monointeldeb128"
}
withLabel: process_medium {
  //--snip
  memory = "16GB"
  cpus = 1
  time = "2h" // You can also set the maximum duration to 2h
  queue = "monointeldeb128"

}
```

nf-core pipelines on the PSMN

To be able to combine our newly defined config file to those defined in `conf` directory, for example test input parameters, we can add the following line inside `rnaseq/nextflow.conf` under the `profiles` section:

```
profiles {
  psmn_test { includeConfig 'conf/psmn_test.config' }
  psmn { includeConfig 'conf/psmn.config' }
  // other profiles
}
```

Then we can run a test with:

```
$ bin/nextflow rnaseq/main.nf -profile psmn_test,test
```

Then you can run the pipeline with your file:

```
$ bin/nextflow rnaseq/main.nf -profile psmn [PARAM]
```

It can be annoying to create those configuration files yourself, that's why I developped a tool nf-core utility.

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`
2. Clone the pipeline code into a folder named `pipeline_name_version`

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`
2. Clone the pipeline code into a folder named `pipeline_name_version`

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`
2. Clone the pipeline code into a folder named `pipeline_name_version`
3. Create configuration files under the `pipeline_name_version/conf` folder
   - psmn_test.config to test the pipeline on the psmn
   - psmn.config to launch the pipeline on the psmn with your own inputs

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`
2. Clone the pipeline code into a folder named `pipeline_name_version`
3. Create configuration files under the `pipeline_name_version/conf` folder
   - psmn_test.config to test the pipeline on the psmn
   - psmn.config to launch the pipeline on the psmn with your own inputs
4. Add those profiles inside `nextflow.config` file

nf-core utility

Utility dedicated to automate the creation of config file in the psmn.
Git repository: https://gitbio.ens-lyon.fr/nfontrod/nfcore_utility

Steps of the script

1. Download nextflow into a `bin` folder inside the `output_folder`
2. Clone the pipeline code into a folder named `pipeline_name_version`
3. Create configuration files under the `pipeline_name_version/conf` folder
   - psmn_test.config to test the pipeline on the psmn
   - psmn.config to launch the pipeline on the psmn with your own inputs
4. Add those profiles inside `nextflow.config` file
5. Give you examples of commands to test the pipeline or to run it with your data

nf-core utility

The tool is packaged into a singularity image on the PSMN.
To easily use the tool, create an alias on your `$HOME/.profile` :

```
# To add in \code{~/.profile}
alias nfutil='singularity exec --pwd `pwd` -B `pwd`:`pwd` /Xnfs/abc/
    singularity/lbmc-nfcore_utility-latest.img python3 /script/nf-
    core_4_psmn.py'
# source your .profile then with
# source ~/.profile
```

nf-core utility


```
$nfutil --help
usage: nf-core_4_psmn.py [-h] -p STR [-o STR] [-s STR] [-l STR] [-v
    STR]

Download a nf-core pipeline in a given folder along with nextflow
      and the singularity image needed to make the pipeline work.
    Then,
create a config file to launche the pipeline in the psmn.

Optional arguments:
-h, --help                        show this help message and exit
-o, --output_folder STR           The folder where the pipeline will
    be downloaded (default '.')
-s, --singularity_cachedir STR  The folder where the singularity
    image will be downloaded (default
".")
-l, --level STR                   The level of information to display
    (default INFO)
-v, --version STR                 The version of the pipeline to
    download (default 'latest')

Required arguments:
-p, --pipeline_name STR           The name of the pipeline to download
    . It can be preceded by nf-core/ but it's not
mandatory.
```

nf-core utility

Preparing the pipeline rnaseq v3.4:

```
$ nfutil -p rnaseq -s ./bin -o . -v 3.4
downloading nextflow...
downloading the pipeline rnaseq
...
Creating a psmn config files
The next step is to test the pipeline:

-To run a test on your computer using docker enter :
export NXF_SINGULARITY_CACHEDIR=bin; bin/nextflow rnaseq_3.4/main.nf
     -c rnaseq_3.4/nextflow.config -profile test,docker
-To run a test on your computer using singularity enter :
export NXF_SINGULARITY_CACHEDIR=bin; bin/nextflow rnaseq_3.4/main.nf
     -c rnaseq_3.4/nextflow.config -profile test,singularity --
    singularity_pull_docker_container
-To run a test on the psmn enter:
export NXF_SINGULARITY_CACHEDIR=bin; bin/nextflow rnaseq_3.4/main.nf
     -profile psmn_test,test

To run the pipeline with your data run:
export NXF_SINGULARITY_CACHEDIR=bin; bin/nextflow rnaseq_3.4/main.nf
     -profile psmn [INPUT_PARAMS]
Where [INPUT_PARAMS] are the input parameter for the pipeline. Check
     the documentation at https://nf-co.re/rnaseq
```

nf-core utility

Launch a test with :

```
$ bin/nextflow rnaseq_3.4/main.nf -profile psmn_test,test
```

Launch the pipeline with your data:

```
$ bin/nextflow rnaseq_3.4/main.nf -profile psmn [INPUT_PARAMS]
```