coproID Documentation

Release 1.1.1

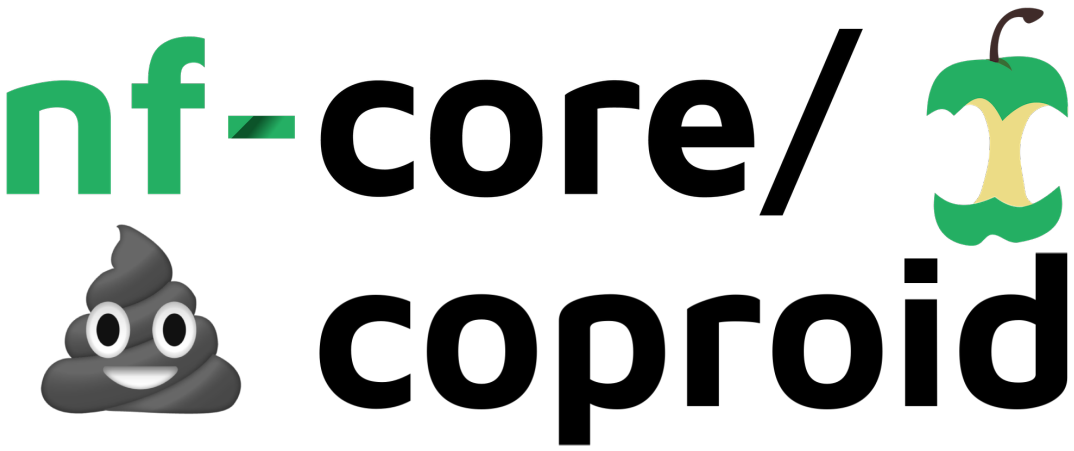
Maxime Borry

Apr 19, 2023

Contents:

CHAPTER 1

Introduction



nf-core/coproid

1.1 A fully reproducible pipeline for COPROlite and paleofeces host IDentification

GitHub Actions CI Status GitHub Actions Linting Status Nextflow install with bioconda Docker Singularity Container available Documentation Status DOI Joins us on Slack Published in PeerJ

CoproID helps you to identify the *"true maker"* of Illumina sequenced Coprolites/Paleofaeces by checking the microbiome composition and the endogenous DNA.

It combines the analysis of putative host ancient DNA with a machine learning prediction of the feces source based on microbiome taxonomic composition:

- (**A**) First coproID performs a comparative mapping of all reads agains two (or three) target genomes (genome1, genome2, and eventually genome3) and computes a host-DNA species ratio (*NormalizedRatio*)
- (**B**) Then coproID performs a metagenomic taxonomic profiling, and compares the obtained profiles to modern reference samples of the target species metagenomes. Using machine learning, coproID then estimates the host source from the metagenomic taxonomic composition (*prop_microbiome*).
- Finally, coproID combines **A** and **B** to predict the likely host of the metagenomic sample.

The coproID pipeline is built using Nextflow, a workflow tool to run tasks across multiple compute infrastructures in a very portable manner. It comes with docker containers making installation trivial and results highly reproducible.

A detailed description of coproID can be found in the article published in PeerJ.

1.2 Quick Start

i. Install `nextflow`

ii. Install either `Docker` or `Singularity` for full pipeline reproducibility (please only use `Conda` as a last resort; see docs)

iii. Download the pipeline and test it on a minimal dataset with a single command

```
nextflow run nf-core/coproid -profile test,<docker/singularity/conda/institute>
```

> Please check nf-core/configs to see if a custom config file to run nf-core pipelines already exists for your Institute. If so, you can simply use `-profile institute` in your command. This will enable either `docker` or `singularity` and set the appropriate execution settings for your local compute environment.

iv. Start running your own analysis!

```
nextflow run maxibor/coproid --genome1 'GRCh37' --genome2 'CanFam3.1' --name1 'Homo_
↪sapiens' --name2 'Canis_familiaris' --reads '*_R{1,2}.fastq.gz' --krakendb 'path/to/
↪minikraken_db' -profile docker
```

This command runs coproID to estimate whether the source of test samples (`--reads '*_R{1,2}.fastq.gz'`) are coming from a human (`--genome1 'GRCh37' -name1 'Homo_sapiens'`) or a dog (`--genome2 'CanFam3.1' --name2 'Canis_familiaris'`), and specifies the path to the minikraken database (`--krakendb 'path/to/minikraken_db'`).

> NB: The example above assumes access to iGenomes.

See *usage docs* for all of the available options when running the pipeline.

1.3 Documentation

The nf-core/coproid pipeline comes with documentation about the pipeline, found in the `docs/` directory:

The nf-core/coproid pipeline comes with documentation about the pipeline, found in the `docs/` directory and at the following address: coproid.readthedocs.io

1. Installation
2. Pipeline configuration
    - Local installation

1.4 Credits

nf-core/coproid was written by Maxime Borry.

1.5 Contributions and Support

If you would like to contribute to this pipeline, please see the contributing guidelines.

For further information or help, don't hesitate to get in touch on Slack (you can join with this invite).

1.6 Citing

coproID has been published in peerJ. The bibtex citation is available below:

```
@article{borry_coproid_2020,
 title = {{CoproID} predicts the source of coprolites and paleofeces using microbiome composition and host {DNA} content},
 volume = {8},
 issn = {2167-8359},
 url = {https://peerj.com/articles/9001},
 doi = {10.7717/peerj.9001},
 language = {en},
 urldate = {2020-04-20},
 journal = {PeerJ},
 author = {Borry, Maxime and Cordova, Bryan and Perri, Angela and Wibowo, Marsha and Honap, Tanvi Prasad and Ko, Jada and Yu, Jie and Britton, Kate and Girdland-Flink, Linus and Power, Robert C. and Stuijts, Ingelise and Salazar-García, Domingo C. and Hofman, Courtney and Hagan, Richard and Kagoné, Thérèse Samdapawindé and Meda, Nicolas and Carabin, Helene and Jacobson, David and Reinhard, Karl and Lewis, Cecil and Kostic, Aleksandar and Jeong, Choongwon and Herbig, Alexander and Hübner, Alexander and Warinner, Christina},
 month = apr,
 year = {2020},
 note = {Publisher: PeerJ Inc.},
 pages = {e9001}
}
```

1.7 Contributors

James A. Fellows Yates

1.8 Tool references

- **AdapterRemoval v2** Schubert, M., Lindgreen, S., & Orlando, L. (2016). AdapterRemoval v2: rapid adapter trimming, identification, and read merging. BMC Research Notes, 9, 88. https://doi.org/10.1186/s13104-016-1900-2
- **FastQC** https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
- **Bowtie2** Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. Nature methods, 9(4), 357. https://dx.doi.org/10.1038%2Fnmeth.1923
- **Samtools** Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., ... 1000 Genome Project Data Processing Subgroup. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics , 25(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352
- **Kraken2** Wood, D. E., Lu, J., & Langmead, B. (2019). Improved metagenomic analysis with Kraken 2. BioRxiv, 762302. https://doi.org/10.1101/762302
- **PMDTools** Skoglund, P., Northoff, B. H., Shunkov, M. V., Derevianko, A. P., Pääbo, S., Krause, J., & Jakobsson, M. (2014). Separating endogenous ancient DNA from modern day contamination in a Siberian Neandertal. Proceedings of the National Academy of Sciences of the United States of America, 111(6), 2229–2234. https://doi.org/10.1073/pnas.1318934111
- **DamageProfiler** Judith Neukamm (Unpublished): 10.5281/zenodo.1064062
- **Sourcepredict** Borry, M. (2019). Sourcepredict: Prediction of metagenomic sample sources using dimension reduction followed by machine learning classification. The Journal of Open Source Software. https://doi.org/10.21105/joss.01540
- **MultiQC** Ewels, P., Magnusson, M., Lundin, S., & Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics , 32(19), 3047–3048. https://doi.org/10.1093/bioinformatics/btw354

CHAPTER 2

Installation

This part of the documentation (common to all nf-core pipelines) is hosted on nf-co.re

nf-co.re/usage/installation

CHAPTER 3

Configuration

This part of the documentation (common to all nf-core pipelines) is hosted on nf-co.re

3.1 Pipeline configuration

- Local installation
- Adding your own system config
- Reference genomes

CHAPTER 4

Usage

4.1 Introduction

Nextflow handles job submissions on SLURM or other environments, and supervises running the jobs. Thus the Nextflow process must run until the pipeline is finished. We recommend that you put the process running in the background through `screen` / `tmux` or similar tool. Alternatively you can run nextflow within a cluster job submitted your job scheduler.

It is recommended to limit the Nextflow Java virtual machines memory. We recommend adding the following line to your environment (typically in `~/.bashrc` or `~./bash_profile`):

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

4.2 Running the pipeline

The typical command for running the pipeline is as follows:

```
nextflow run nf-core/coproid --genome1 'GRCh37' --genome2 'CanFam3.1' --name1 'Homo_
↪sapiens' --name2 'Canis_familiaris' --reads '*_R{1,2}.fastq.gz' --krakendb 'path/to/
↪minikraken_db' -profile docker
```

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

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

```
work            # Directory containing the nextflow working files
results         # Finished results (configurable, see below)
.nextflow_log   # Log file from Nextflow
# Other nextflow hidden files, eg. history of pipeline runs and old logs.
```

4.2.1 Updating the pipeline

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

```
nextflow pull nf-core/coproid
```

4.2.2 Reproducibility

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

First, go to the nf-core/coproid releases page and find the latest version number - numeric only (eg. `1.3.1`). Then specify this when running the pipeline with `-r` (one hyphen) - eg. `-r 1.3.1`.

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

4.3 Main arguments

4.3.1 `-profile`

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

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

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

The pipeline also dynamically loads configurations from https://github.com/nf-core/configs when it runs, making multiple config profiles for various institutional clusters available at run time. For more information and to see if your system is available in these configs please see the nf-core/configs documentation.

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

If `-profile` is not specified, the pipeline will run locally and expect all software to be installed and available on the `PATH`. This is *not* recommended.

- `docker`
  - A generic configuration profile to be used with Docker
  - Pulls software from dockerhub: `nfcore/coproid`
- `singularity`
  - A generic configuration profile to be used with Singularity
  - Pulls software from DockerHub: `nfcore/coproid`
- `conda`

  - Please only use Conda as a last resort i.e. when it's not possible to run the pipeline with Docker or Singularity.
  - A generic configuration profile to be used with Conda
  - Pulls most software from Bioconda
- `test`
  - A profile with a complete configuration for automated testing
  - Includes links to test data so needs no other parameters

4.3.2 `--reads`

Use this to specify the location of your input FastQ files. For example:

```
--reads 'path/to/data/sample_*_{1,2}.fastq'
```

Please note the following requirements:

1. The path must be enclosed in quotes
2. The path must have at least one `*` wildcard character
3. When using the pipeline with paired end data, the path must use `{1,2}` notation to specify read pairs.

If left unspecified, a default pattern is used: `data/*{1,2}.fastq.gz`

4.3.3 `--single_end`

By default, the pipeline expects paired-end data. If you have single-end data, you need to specify `--single_end` on the command line when you launch the pipeline. A normal glob pattern, enclosed in quotation marks, can then be used for `--reads`. For example:

```
--single_end --reads '*.fastq'
```

It is not possible to run a mixture of single-end and paired-end files in one run.

4.3.4 `--name1`

Name of the first candidate species. Example : `"Homo_sapiens"`

4.3.5 `--name2`

Name of the second candidate species. Example : `"Canis_familiaris"`

4.3.6 `--krakenDB`

Path to the directory containing the Kraken2 MiniKraken2_v2_8GB database files. The MiniKraken2_v2_8GB database can be downloaded here

```
--krakendb "path/to/kraken2_db_dir"
```

4.4 Reference genomes

The pipeline config files come bundled with paths to the illumina iGenomes reference index files. If running with docker or AWS, the configuration is set up to use the AWS-iGenomes resource.

4.4.1 `--fasta1`

Reference genome1 can be specified by using the full path to the genome fasta file. Must be provided if `--genome1` is not provided.

```
--fasta1 'path/to/fasta/reference1.fa'
```

4.4.2 `--fasta2`

Reference genome2 can be specified by using the full path to the genome fasta file. Must be provided if `--genome2` is not provided.

```
--fasta2 'path/to/fasta/reference2.fa'
```

4.4.3 `--genome1` (using iGenomes)

Alternatively, reference genomes can be specified using pre-index genomes available through the iGenomes service. Must be provided if `--fasta1` is not provided.

There are 31 different species supported in the iGenomes references. To run the pipeline, you must specify which to use with the `--genome` flag.

You can find the keys to specify the genomes in the iGenomes config file. Common genomes that are supported are:

- Human
  - `--genome GRCh37`
- Dog
  - `--genome CanFam3.1`

  There are numerous others - check the config file for more.

Note that you can use the same configuration setup to save sets of reference files for your own use, even if they are not part of the iGenomes resource. See the Nextflow documentation for instructions on where to save such a file.

The syntax for this reference configuration is as follows:

```
params {
  // illumina iGenomes reference file paths
  genomes {
    'GRCh37' {
      fasta   = "${params.igenomes_base}/Homo_sapiens/Ensembl/GRCh37/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Homo_sapiens/Ensembl/GRCh37/Sequence/
↪Bowtie2Index/genome"
    }
    'GRCm38' {
      fasta   = "${params.igenomes_base}/Mus_musculus/Ensembl/GRCm38/Sequence/
↪WholeGenomeFasta/genome.fa"
```

(continues on next page)

(continued from previous page)

```
      bowtie2 = "${params.igenomes_base}/Mus_musculus/Ensembl/GRCh37/Sequence/
↪Bowtie2Index/genome"
    }
    'UMD3.1' {
      fasta   = "${params.igenomes_base}/Bos_taurus/Ensembl/UMD3.1/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Bos_taurus/Ensembl/UMD3.1/Sequence/
↪Bowtie2Index/genome"
    }
    'CanFam3.1' {
      fasta   = "${params.igenomes_base}/Canis_familiaris/Ensembl/CanFam3.1/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Canis_familiaris/Ensembl/CanFam3.1/Sequence/
↪Bowtie2Index/genome"
    }
    'EquCab2' {
      fasta   = "${params.igenomes_base}/Equus_caballus/Ensembl/EquCab2/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Equus_caballus/Ensembl/EquCab2/Sequence/
↪Bowtie2Index/genome"
    }
    'Galgal4' {
      fasta   = "${params.igenomes_base}/Gallus_gallus/Ensembl/Galgal4/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Gallus_gallus/Ensembl/Galgal4/Sequence/
↪Bowtie2Index/genome"
    }
    'Mmul_1' {
      fasta   = "${params.igenomes_base}/Macaca_mulatta/Ensembl/Mmul_1/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Macaca_mulatta/Ensembl/Mmul_1/Sequence/
↪Bowtie2Index/genome"
    }
    'CHIMP2.1.4' {
      fasta   = "${params.igenomes_base}/Pan_troglodytes/Ensembl/CHIMP2.1.4/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Pan_troglodytes/Ensembl/CHIMP2.1.4/Sequence/
↪Bowtie2Index/genome"
    }
    'Rnor_6.0' {
      fasta   = "${params.igenomes_base}/Rattus_norvegicus/Ensembl/Rnor_6.0/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Rattus_norvegicus/Ensembl/Rnor_6.0/Sequence/
↪Bowtie2Index/genome"
    }
    'Sscrofa10.2' {
      fasta   = "${params.igenomes_base}/Sus_scrofa/Ensembl/Sscrofa10.2/Sequence/
↪WholeGenomeFasta/genome.fa"
      bowtie2 = "${params.igenomes_base}/Sus_scrofa/Ensembl/Sscrofa10.2/Sequence/
↪Bowtie2Index/genome"
    }
  }
}
```

4.4.4 --genome2 (using iGenomes)

Name of iGenomes reference for candidate organism 2. Must be provided if `--fasta2` is not provided.

```
--genome2 'CanFam3.1'
```

4.4.5 --igenomes_ignore

Do not load `igenomes.config` when running the pipeline. You may choose this option if you observe clashes between custom parameters and those supplied in `igenomes.config`.

4.5 Settings

4.5.1 --adna

Specified if data is modern (false) or ancient DNA (true). Default = true

```
--adna true
```

or

```
--adna false
```

4.5.2 --phred

Specifies the fastq quality encoding (33 | 64). Defaults to 33

```
--phred 33
```

or

```
--phred 64
```

4.5.3 --collapse

Specifies if AdapterRemoval should merge the paired-end sequences or not. Default = true

```
--collapse true
```

or

```
--collapse false
```

4.5.4 --identity

Identity threshold to retain read alignment. Default = 0.95

```
--identity 0.95
```

4.5.5 --pmdscore

Minimum PMDscore to retain read alignment. Default = 3

```
--pmdscore 3
```

4.5.6 --library

DNA preparation library type (classic | UDGhalf). Default = classic

```
--library classic
```

or

```
--library UDGhalf
```

4.5.7 --bowtie

Bowtie settings for sensivity (very-fast | very-sensitive). Default = very-sensitive

```
--bowtie very-fast
```

or

```
--bowtie very-sensitive
```

4.5.8 --minKraken

Minimum number of Kraken hits per Taxonomy ID to report. Default = 50

```
--minKraken 50
```

4.5.9 --endo1

Proportion of Endogenous DNA in organism 1 target microbiome. Must be between 0 and 1. Default = 0.01

```
--endo1 0.01
```

4.5.10 --endo2

Proportion of Endogenous DNA in organism 2 target microbiome. Must be between 0 and 1. Default = 0.01

```
--endo2 0.01
```

4.5.11 sp_embed

SourcePredict embedding algorithm. One of mds, tsne, umap. Default to mds from coproID version 1.1

```
--sp_embed mds
```

More information is available in the Sourcepredict documentation

4.5.12 sp_norm

Sourcepredict normalization method. One of 'rle', 'gmpr', 'subsample'. Default = 'gmpr'

```
--sp_norm 'gmpr'
```

More informations are available in the Sourcepredict documentation

4.5.13 sp_neighbors

Sourcepredict numbers of neighbors for KNN ML. Integer or all. Default = all

```
--sp_neighbors all
```

More informations are available in the Sourcepredict documentation

4.6 Other coproID parameters

4.6.1 --name3

Name of candidate species 3.

`--name3 Sus_scrofa`

4.6.2 --fasta3

Reference genome3 can be specified by using the full path to the genome fasta file. Must be provided if `--genome3` is not provided.

```
--fasta3 'path/to/fasta/reference3.fa'
```

4.6.3 --genome3 (using iGenomes)

Name of iGenomes reference for candidate organism 3. Must be provided if `--fasta3` is not provided.See `--genome1` above for more details.

```
--genome3 'Sscrofa10.2'
```

4.6.4 --endo3

Proportion of Endogenous DNA in organism 3 target microbiome. Must be between 0 and 1. Default = 0.01

```
--endo3 0.01
```

4.6.5 --index1

Path to Bowtie2 pre-indexed genome candidate 1 Coprolite maker's genome

```
--index1 'path/to/bt_index/basename1'
```

4.6.6 --index2

Path to Bowtie2 pre-indexed genome candidate 2 Coprolite maker's genome

```
--index2 'path/to/bt_index/basename2'
```

4.6.7 --index3

Path to Bowtie2 pre-indexed genome candidate 3 Coprolite maker's genome

```
--index3 'path/to/bt_index/basename3'
```

4.7 Job resources

4.7.1 Automatic resubmission

Each step in the pipeline has a default set of requirements for number of CPUs, memory and time. For most of the steps in the pipeline, if the job exits with an error code of `143` (exceeded requested resources) it will automatically resubmit with higher requests (2 x original, then 3 x original). If it still fails after three times then the pipeline is stopped.

4.7.2 Custom resource requests

Wherever process-specific requirements are set in the pipeline, the default value can be changed by creating a custom config file. See the files hosted at `nf-core/configs` for examples.

If you are likely to be running `nf-core` pipelines regularly it may be a good idea to request that your custom config file is uploaded to the `nf-core/configs` git repository. Before you do this please can you test that the config file works with your pipeline of choice using the `-c` parameter (see definition below). You can then create a pull request to the `nf-core/configs` repository with the addition of your config file, associated documentation file (see examples in `nf-core/configs/docs`), and amending `nfcore_custom.config` to include your custom profile.

If you have any questions or issues please send us a message on Slack.

4.8 AWS Batch specific parameters

Running the pipeline on AWS Batch requires a couple of specific parameters to be set according to your AWS Batch configuration. Please use `-profile awsbatch` and then specify all of the following parameters.

4.8.1 --awsqueue

The JobQueue that you intend to use on AWS Batch.

4.8.2 --awsregion

The AWS region in which to run your job. Default is set to `eu-west-1` but can be adjusted to your needs.

4.8.3 --awscli

The AWS CLI path in your custom AMI. Default: `/home/ec2-user/miniconda/bin/aws`.

Please make sure to also set the `-w/--work-dir` and `--outdir` parameters to a S3 storage bucket of your choice - you'll get an error message notifying you if you didn't.

4.9 Other command line parameters

4.9.1 --outdir

The output directory where the results will be saved.

4.9.2 --email

Set this parameter to your e-mail address to get a summary e-mail with details of the run sent to you when the workflow exits. If set in your user config file (`~/.nextflow/config`) then you don't need to specify this on the command line for every run.

4.9.3 --email_on_fail

This works exactly as with `--email`, except emails are only sent if the workflow is not successful.

4.9.4 --max_multiqc_email_size

Threshold size for MultiQC report to be attached in notification email. If file generated by pipeline exceeds the threshold, it will not be attached (Default: 25MB).

4.9.5 -name

Name for the pipeline run. If not specified, Nextflow will automatically generate a random mnemonic.

This is used in the MultiQC report (if not default) and in the summary HTML / e-mail (always).

NB: Single hyphen (core Nextflow option)

4.9.6 -resume

Specify this when restarting a pipeline. Nextflow will used cached results from any pipeline steps where the inputs are the same, continuing from where it got to previously.

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

NB: Single hyphen (core Nextflow option)

4.9.7 -c

Specify the path to a specific config file (this is a core NextFlow command).

NB: Single hyphen (core Nextflow option)

Note - you can use this to override pipeline defaults.

4.9.8 --custom_config_version

Provide git commit id for custom Institutional configs hosted at `nf-core/configs`. This was implemented for reproducibility purposes. Default: `master`.

```
## Download and use config file with following git commid id
--custom_config_version d52db660777c4bf36546ddb188ec530c3ada1b96
```

4.9.9 --custom_config_base

If you're running offline, nextflow will not be able to fetch the institutional config files from the internet. If you don't need them, then this is not a problem. If you do need them, you should download the files from the repo and tell nextflow where to find them with the `custom_config_base` option. For example:

```
## Download and unzip the config files
cd /path/to/my/configs
wget https://github.com/nf-core/configs/archive/master.zip
unzip master.zip

## Run the pipeline
cd /path/to/my/data
nextflow run /path/to/pipeline/ --custom_config_base /path/to/my/configs/configs-
↪master/
```

> Note that the nf-core/tools helper package has a `download` command to download all required pipeline files + singularity containers + institutional configs in one go for you, to make this process easier.

4.9.10 --max_memory

Use to set a top-limit for the default memory requirement for each process. Should be a string in the format integer-unit. eg. `--max_memory '8.GB'`

4.9.11 --max_time

Use to set a top-limit for the default time requirement for each process. Should be a string in the format integer-unit. eg. `--max_time '2.h'`

4.9.12 --max_cpus

Use to set a top-limit for the default CPU requirement for each process. Should be a string in the format integer-unit. eg. `--max_cpus 1`

4.9.13 --plaintext_email

Set to receive plain-text e-mails instead of HTML formatted.

4.9.14 --monochrome_logs

Set to disable colourful command line output and live life in monochrome.

4.9.15 --multiqc_config

Specify a path to a custom MultiQC configuration file.

CHAPTER 5

Output

This document describes the output produced by the coproID pipeline. Results are found in the `results` directory (default, specified by `--outdir`).

5.1 MultiQC report

File `multiqc_report.html`

5.1.1 FastQC section

FastQC gives general quality metrics about your reads. It provides information about the quality score distribution across your reads, the per base sequence content (%T/A/G/C). You get information about adapter contamination and other overrepresented sequences.

For further reading and documentation see the FastQC help.

> **NB:** The FastQC plots displayed in the MultiQC report shows *untrimmed* reads. They may contain adapter sequence and potentially regions with low quality.

5.1.2 AdapterRemoval section

AdapterRemoval searches for and removes remnant adapter sequences from High-Throughput Sequencing (HTS) data and (optionally) trims low quality bases from the 3' end of reads following adapter removal. AdapterRemoval can analyze both single end and paired end data, and can be used to merge overlapping paired-ended reads into (longer) consensus sequences.

- *Retained and Discarded Paired-End Collapsed*: This plot shows the number/proportion of reads that passed adapter removal and trimming filters.
- *Length Distribution Paired End Collapsed*: This plot shows the length distribution of the different read categories.

5.1.3 Bowtie2

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. This plot shows the number of reads aligning to the reference in different ways.

5.1.4 DamageProfiler

DamageProfiler calculates damage profiles of mapped reads. These plots represents the damage patterns and read length distribution.

5.1.5 nf-core/coproid Software Versions

This section shows the version of the different softwares used in this pipeline.

5.2 coproID_report.html

This file contains the coproID report

5.2.1 coproID summary table

This table summarizes the read ratios and microbiome source proportions as computed by coproID and sourcepredict. You can download the table in `.csv` format by clicking on the green "Download" button.

5.2.2 coproID summary plot

This plot summarizes the coproID prediction.

> **Note:** This plot is only available when coproID is used with 2 organisms

5.2.3 microbiome profile embedding

This interactive plot shows the embedding of the microbiome samples by sourcepredict

5.2.4 Damage plots

These plots represent the damage patterns computed by DamageProfiler

5.3 coproID_result.csv

This table summarizes the read ratios and microbiome source proportions as computed by coproID and sourcepredict.

5.4 coproID_bp.csv

This table contains the mapped base pair counts (ancient and modern reads) for each sample.

5.5 kraken

This directory contains the merged OTU count for all samples of the run, as counted by Kraken2

5.6 damageprofiler

This directory contains all the output files of DamageProfiler (see multiqc section above)

5.7 alignments

This directory contains the alignment `.bam` files for aligned and aligned sequences to each target genome.

5.8 pmdtools

This directory contains the alignment `.bam` files for aligned and aligned **ancient DNA** sequences to each target genome, according to PMDTools.

5.9 pipeline_info

This directory contains all the informations about the pipeline run.

5.9.1 execution_report.html

Interactive report showing resources used in the execution of the pipeline

5.9.2 execution_timeline.html

Timeline of pipeline execution

5.9.3 execution_trace.txt

Log of pipeline execution

5.9.4 pipeline_dag.svg

Pipeline workflow overview

5.9.5 pipeline_report.html

nf-core log of pipeline metadata and execution

5.9.6 pipeline_report.txt

Same as above, in text format

5.9.7 results_description.html

The content of this page

5.9.8 software_versions.csv

List of softwares and their versions.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search