nf-core: Community curated bioinformatics pipelines

Philip A Ewels[1], Alexander Peltzer[2], Sven Fillinger[2], Johannes Alneberg[1], Harshil Patel[3], Andreas Wilm[4], Maxime Ulysse Garcia[5], Paolo Di Tommaso[6], Sven Nahnsen[2]

1. Science for Life Laboratory (SciLifeLab), Department of Biochemistry and Biophysics, Stockholm University, Stockholm, Sweden
2. Quantitative Biology Center (QBiC), University of Tübingen, Tübingen, Germany
3. Bioinformatics and Biostatistics, The Francis Crick Institute, London, United Kingdom
4. Computational & Systems Biology, Genome Institute of Singapore, Singapore
5. Department of Oncology-Pathology, Karolinska Institutet, Stockholm, Sweden
6. Centre for Genomic Regulation (CRG), The Barcelona Institute for Science and Technology, Barcelona, Spain.

Abstract

The standardization, portability, and reproducibility of analysis pipelines is a renowned problem within the bioinformatics community. Bioinformatic analysis pipelines are often designed for execution on-premise, and this inevitably leads to a level of customisation and integration that is only applicable to the local infrastructure. More notably, the software required to run these pipelines is also tightly coupled with the local compute environment, and this leads to poor pipeline portability, and reproducibility of the ensuing results - both of which are fundamental requirements for the validation of scientific findings. Here we introduce *nf-core*, a framework that provides a community-driven platform for the creation and development of best practice analysis pipelines written in the Nextflow language. Nextflow has built-in support for pipeline execution on most computational infrastructures, as well as automated deployment using container technologies such as Conda, Docker, and Singularity. Therefore, key obstacles in pipeline development such as portability, reproducibility, scalability and unified parallelism are inherently addressed by all *nf-core*

pipelines. Furthermore, to ensure that new pipelines can be added seamlessly, and existing pipelines are able to inherit up-to-date functionality the *nf-core* community is actively developing a suite of tools that automate pipeline creation, testing, deployment and synchronization. The peer-review process during pipeline development ensures that best practices and common usage patterns are imposed and therefore, adhere to community guidelines. Our primary goal is to provide a community-driven platform for high-quality, excellent documented and reproducible bioinformatics pipelines that can be utilized across various institutions and research facilities.

Introduction

Being able to reproduce scientific results is the central tenet of the scientific method. To move towards FAIR (Findable, Accessible, Interoperable and Reusable) research methods[1] in data-driven science is complex[2,3]. Central repositories such as qPortal[4], omictools[5], and the Galaxy toolshed[6] make it possible to find existing pipelines and their associated tools. However, it is still notoriously challenging to develop analysis pipelines that are fully reproducible and interoperable across multiple systems and institutions, not only due to differences in hardware, operating systems and software versions.

Although some analysis pipelines have become standardised (e.g. GATK best practices), the actual implementations are usually developed by individual researchers. As such, they are rarely thoroughly tested, well documented or shared with others. This can give rise to multiple groups reinventing the wheel, creating highly individualized in-house pipelines. This not only hampers sustainable sharing of data and tools but also results in a proliferation of heterogeneous analysis pipelines, making it difficult for newcomers to find what they need for a specific analysis question.

As the scale of ‘omics data and its associated tools has grown dramatically, the scientific community is increasingly moving towards the use of specialized workflow management systems to build analysis pipelines. They separate the requirements of the underlying computing infrastructure from the analysis and workflow description, and therefore introduce a higher degree of portability compared to custom in-house scripts. One such popular tool is Nextflow[7]; a workflow manager able to abstract pipeline logic from its underlying compute infrastructure. Software packages can be bundled with analysis pipelines using built-in integration with package managers like Conda and container environments, such as Docker and Singularity. Support for most common High Performance Computing (HPC) batch schedulers and cloud providers allows simple deployment of analysis pipelines in almost any infrastructure in a transparent and portable manner.

Pipelines within *nf-core* implement the same framework principles, for example, the way package dependencies need to be resolved or how the execution on a test data set works. Once a researcher is familiar with one *nf-core* analysis pipeline, all others will work on the same system with the same logic. Detailed documentation and templates for new analysis pipelines make it trivial to start a new project, making *nf-core* an excellent entry point for Nextflow pipeline development. Whilst current *nf-core* analysis pipelines currently focus on life sciences, the concept is also applicable to other data-intensive fields such as chemistry and physics.

Other high-level approaches to facilitate the creation of analysis pipelines are available: Flowcraft[8] and Pipeliner[9] are meta-tools to dynamically build Nextflow pipelines from template blocks. The awesome-pipelines website lists community Nextflow pipelines, though these do not follow consistent implementation patterns. Other workflow tools have collections of pipelines, although with less extensively defined rules as *nf-core*. These include Snakemake Workflows[10]; an early-stage community project attempting to collect best practice analysis pipelines based on the Snakemake[11] workflow manager, the Galaxy Toolshed[6] for Galaxy[12] and the ENCODE project pipelines. Other than Snakemake or Nextflow, the CWL[13] project aims at providing researchers with a toolset for creating analysis pipelines.

nf-core is not only a collection of Nextflow pipelines maintained by a single institution or research unit but an entire user community of research institutions that all contribute to a larger objective. With this philosophy in mind, *nf-core* provides a set of framework tools that can help users to develop, build and maintain pipelines in a way that has not been met by other, similar attempts for the creation of pipeline communities. While being strongly represented in the analysis of next-generation sequencing data in genomics, the community also provides a more general framework for building and maintaining analysis pipelines in proteomics, metabolomics, and other life science applications, following best-practices in the field of scientific data analysis[14–16].

Results

User community

The heart of *nf-core* is its community. More than just a static collection of analysis pipelines, the *nf-core* community strives to provide a framework for developer best-practices, code standardization, and excellent documentation. The open nature of this community allows discussion and collaboration on both general best-practices and also domain-specific pipelines by experts across the world. Technical issues are characteristically resolved by a group of people, ensuring those resulting solutions are reliable, well designed and thoroughly tested. This open process breaks down traditional barriers and silos between research groups, resulting in high-quality pipelines that anyone can use. The early feedback and successive reviewing process allow developers to improve their analysis pipelines and ensures that they adhere to community standards. Through discussions and collaborative efforts, new guidelines and tools naturally evolve and are made available to all *nf-core* developers.

The hub of the community is the *nf-core* website (https://nf-co.re). It contains a listing of all analysis pipelines as well as documentation and tutorials, serving as the first point of entrance for newcomers, and a point of reference for more experienced developers. For communication, we have a mailing list "nf-core" on Google Groups as well as a Slack workspace (https://nf-core.slack.com/) that can provide access to help and discussion with other members of *nf-core*. Announcements, updates, and news to the community are distributed via Twitter (https://twitter.com/nf_core) to reach a broader audience.

All code is hosted on GitHub in the *nf-core* organization (https://github.com/nf-core/) with issue tracking and code collaboration via pull-requests. GitHub assures accurate attribution of work and gives a public forum for code, bug reports, and feature requests. At the time of writing, the *nf-core* community has 63 contributing team members with 19 listed contributing organizations and more than 500 twitter followers.

The *nf-core* project ties in well with other communities, notably Nextflow[7], Bioconda[17] and conda-forge (https://conda-forge.org/). Wherever possible, we encourage users and developers to contribute to these projects and communities. For example, we package missing software in the bioconda or conda-forge repositories, helping the wider bioinformatics community. We have started to integrate with initiatives such as bio.tools and the GA4GH-compliant Dockstore[18] and hope to work together with the Biocontainers project in the future to further simplify software packaging.

nf-core analysis pipelines

Currently, *nf-core* encompasses a total of 26 analysis pipelines that span multiple life-science domains and analysis types. For example; `nf-core/rnaseq` generates gene counts and quality metrics for RNA-Seq data, `nf-core/mhcquant` identifies and quantifies peptides from mass spectrometry raw data and `nf-core/eager`[19] processes genomics data from ancient DNA samples. New features and tools are added to existing pipelines through the use of configurable parameters, the use of which is logged for every analysis pipeline execution.

All *nf-core* pipelines are released under the MIT license, making them available for anyone and encouraging downstream modification and reuse. Indeed, a key feature of the *nf-core* analysis pipelines is to provide “base” pipelines that can be built upon to provide more niche project-specific outputs. We require all pipelines to specify a contact person who is responsible for curating the pipeline. All pipelines need to provide high-quality documentation: both pipeline usage documentation with examples and also a description of the generated results files.

Since *nf-core* pipelines use Nextflow, they can be used across a huge range of environments. All of the software dependencies required for any given analysis pipeline are available using either Conda, Docker or Singularity, and Nextflow supports virtually all job scheduling software. A centralized repository holding cluster-specific configuration files means that a new cluster-specific config profile can be made available for all pipelines instantly, through a single update.

To ensure reproducibility, *nf-core* pipelines are tagged with release numbers which link pipeline code to software dependencies. Pipelines contain a conda environment with pinned software versions and a reference to a tagged Docker image containing the exact analysis environment. Users can provide Nextflow with a command line parameter to specify the release version, ensuring that the very same software dependencies and the same pipeline are used to generate results.

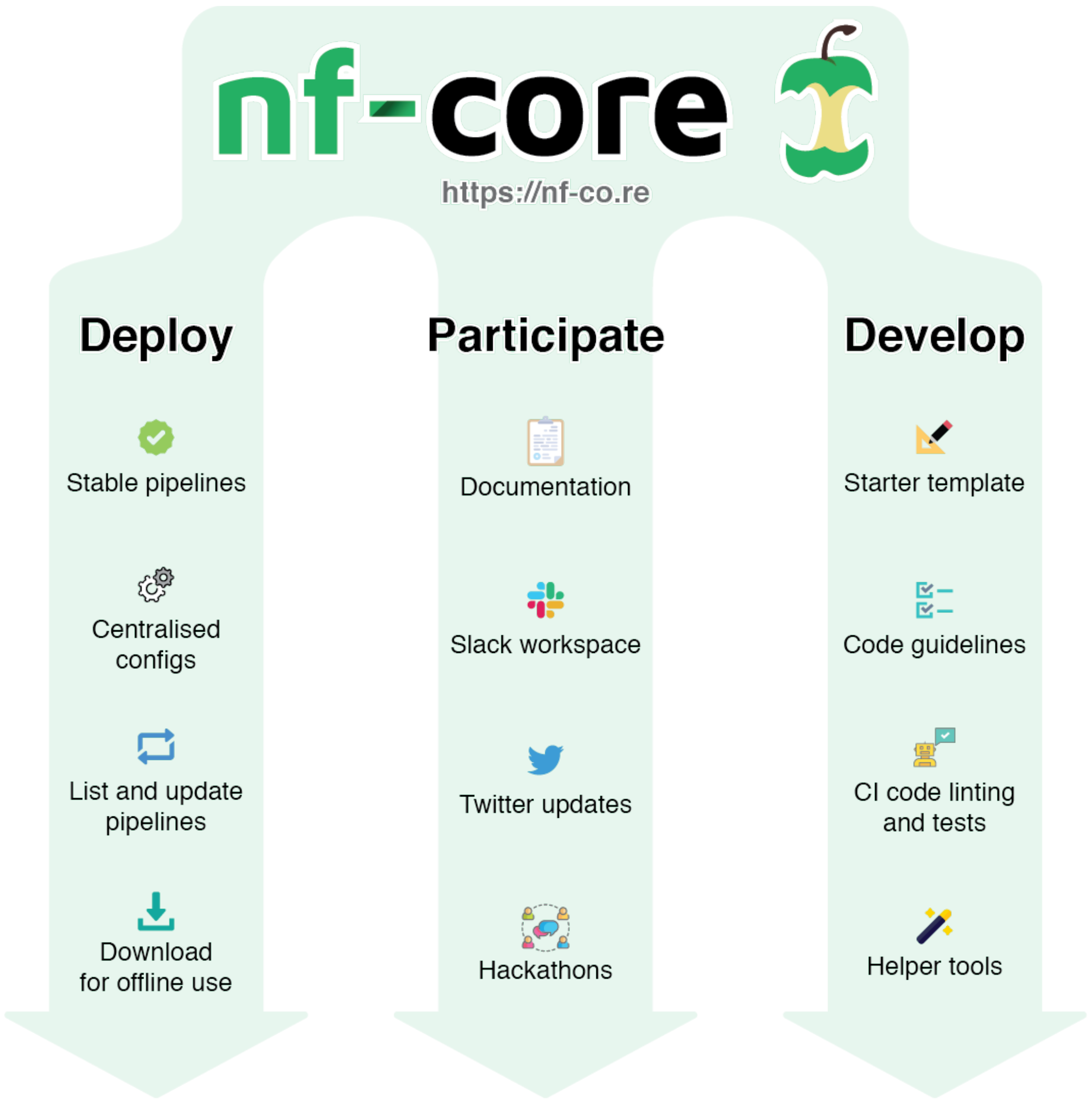


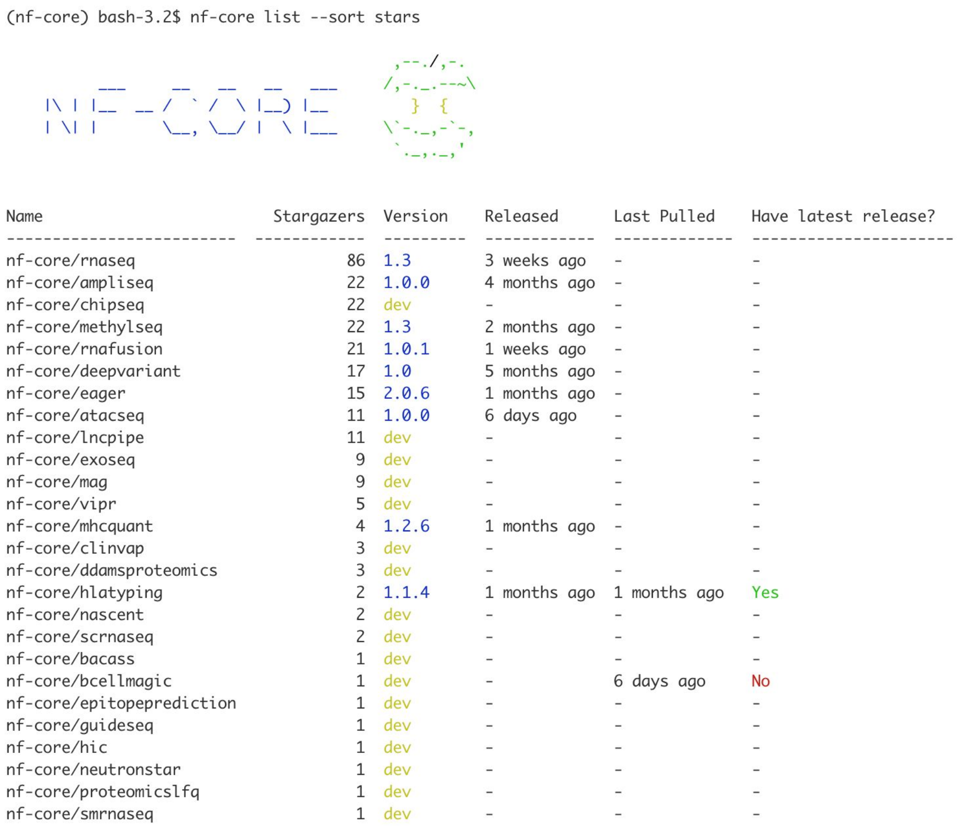
Figure 1: Main concepts of nf-core

The *nf-core* framework

To help *nf-core* users and developers, we provide a set of command-line tools to automate common tasks (Fig. 2). Users can list available analysis pipelines and check that their versions are up to date, as well as downloading all required files for offline use in a single command. Software licenses can be printed for each pipeline. Developers have a suite of commands to build, test and release new analysis pipelines.

When working on a pipeline, a linting command checks the code against current best-practice recommendations defined by the *nf-core* community. Various configuration variables and essential pipeline components are tested, such as a minimum required Nextflow version and consistent version numbers. Additional linting checks also provide helpful warnings, such as notifications about new versions of software available on bioconda.

Linting tests and minimal test-dataset runs are automatically triggered whenever there is a change to a pipeline, using continuous integration with Travis CI[20]. This ensures that developers can guarantee a working pipeline at all times.



```
(nf-core) bash-3.2$ nf-core list --sort stars

                                          ,--./,-.
          ___     __   __   __   ___     /,-._.--~\
    |\ | |__  __ /  ` /  \ |__) |__         }  {
    | \| |       \__, \__/ |  \ |___     \`-._,-`-,
                                          `._,._,'

Name                         Stargazers  Version    Released      Last Pulled    Have latest release?
-------------------------  ------------  ---------  ------------  -------------  ----------------------
nf-core/rnaseq                       86  1.3        3 weeks ago   -              -
nf-core/ampliseq                     22  1.0.0      4 months ago  -              -
nf-core/chipseq                      22  dev        -             -              -
nf-core/methylseq                    22  1.3        2 months ago  -              -
nf-core/rnafusion                    21  1.0.1      1 weeks ago   -              -
nf-core/deepvariant                  17  1.0        5 months ago  -              -
nf-core/eager                        15  2.0.6      1 months ago  -              -
nf-core/atacseq                      11  1.0.0      6 days ago    -              -
nf-core/lncpipe                      11  dev        -             -              -
nf-core/exoseq                        9  dev        -             -              -
nf-core/mag                           9  dev        -             -              -
nf-core/vipr                          5  dev        -             -              -
nf-core/mhcquant                      4  1.2.6      1 months ago  -              -
nf-core/clinvap                       3  dev        -             -              -
nf-core/ddamsproteomics               3  dev        -             -              -
nf-core/hlatyping                     2  1.1.4      1 months ago  1 months ago   Yes
nf-core/nascent                       2  dev        -             -              -
nf-core/scrnaseq                      2  dev        -             -              -
nf-core/bacass                        1  dev        -             -              -
nf-core/bcellmagic                    1  dev        -             6 days ago     No
nf-core/epitopeprediction             1  dev        -             -              -
nf-core/guideseq                      1  dev        -             -              -
nf-core/hic                           1  dev        -             -              -
nf-core/neutronstar                   1  dev        -             -              -
nf-core/proteomicslfq                 1  dev        -             -              -
nf-core/smrnaseq                      1  dev        -             -              -
```

Figure 2: Example output from the nf-core framework tools, the 'list' command shows available analysis pipelines.

To help developers get started with new pipelines, *nf-core* provides a standardized template pipeline which adheres to all *nf-core* guidelines (Table 1). Using a command-line wizard, developers enter their pipeline name and metadata to get a bare-bones pipeline with a large number of "`TODO`" comment lines that can be aggregated by most code editors, each explaining what needs to be added or customized. The use of this template greatly lowers the learning curve when getting started with Nextflow and helps to homogenize the coding style of all *nf-core* pipelines.

	Guideline	Comment
❗	Be built using Nextflow	
❗	Have an MIT license	
❗	Software bundled using Docker / Singularity	
❗	Continuous integration testing	Tests run on Travis CI with minimal test datasets.
❗	The pipeline must not have any failures in the *nf-core* lint tests	Lint tests run on Travis CI with each update.
❗	Stable release tags	GitHub releases are used with an associated changelog and DOI using Zenodo.
❗	Common pipeline structure and usage	
❗	Run in a single command	Pipelines should not be split into multiple sub-pipelines.
❗	Excellent documentation	Pipeline specific documentation, in addition to documentation on the main *nf-core* website.
❗	A responsible contact person / GitHub username	
✓	Software bundled using bioconda	Software should be packaged for bioconda if not already available.
✓	Require as little input metadata as possible	Optional pipeline steps to convert file types and build reference indexes recommended.
✓	Optimized output file formats	Standard file formats where possible, including CRAM.
✓	Explicit support for running in cloud environments	
✓	Benchmarks from running on cloud environments such as AWS	

Table 1: nf-core guidelines. See https://nf-co.re/guidelines for the latest version.

❗ *denotes requirements,* ✓ *denotes recommendations.*

The *nf-core* pipeline template and guidelines change over time and new best practice insights need to be integrated into existing pipelines. We provide an automated synchronization mechanism that introduces relevant base template changes to existing pipelines as pull requests on Github every time a new version of the helper tools is released

(Fig S1). The pipeline maintainers can review the suggested changes and merge them into the pipeline source code, updating the pipeline with minimal manual effort.

All *nf-core* pipelines can be run on any computing infrastructure supported by Nextflow. This facilitates the possibility to run *nf-core* pipelines on any POSIX system as well as cloud computing infrastructures such as AWS Batch, Google Cloud Platform and Kubernetes. The opportunity to run pipelines locally during development for initial examination, and then to proceed seamlessly to large-scale computational resources in HPC or cloud settings, provides users and developers with great flexibility.

Discussion

The utilization of workflow management systems and container technologies alone do not ensure reproducible computational analysis. They can still lead to local platform dependencies which inherently limits portability and reproducibility if not implemented properly. *nf-core* provides both an architectural framework and a community platform to develop, share and discuss best-practice bioinformatics analysis pipelines.

By leveraging the full feature sets of Nextflow, (Bio-) Conda, Docker, and Singularity, the *nf-core* framework with its community pipelines directly tackle the reproducibility and portability concerns of researchers. This helps new users and developers alike and enables the sharing of fully reproducible best-practices implementations across institutions. All pipelines adhere to the same usage pattern enabling both developers and users to familiarize themselves with *nf-core* pipelines rapidly.

As the usage of workflow management tools spreads, an increasing number of tertiary tools are tying into the ecosystem. The *nf-core* analysis pipelines are at the forefront of this, with integration to the Dockstore platform with GA4GH support. We are in the process of building interactive command-line and graphical interfaces to further simplify launching *nf-core* pipelines.

We plan to extend the existing framework for automatic benchmarks using full-size test datasets. Additionally, we intend to provide more information for accurate cloud computing price estimates. Looking ahead, we hope to welcome more contributors and pipelines to the *nf-core* community to build on the solid foundation that has been established.

References

1. Sansone, S.-A. *et al.* FAIRsharing as a community approach to standards, repositories and policies. *Nat. Biotechnol.* **37**, 358 (2019).
2. Perkel, J. M. A toolkit for data transparency takes shape. *Nature* **560**, 513–515 (2018).
3. Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature* **533**, 452–454 (2016).
4. Mohr, C. *et al.* qPortal: A platform for data-driven biomedical research. *PLoS One* **13**, e0191603 (2018).
5. Henry, V. J., Bandrowski, A. E., Pepin, A.-S., Gonzalez, B. J. & Desfeux, A. OMICtools: an informative directory for multi-omic data analysis. *Database* **2014**, (2014).
6. Blankenberg, D. *et al.* Dissemination of scientific software with Galaxy ToolShed. *Genome Biol.* **15**, 403 (2014).
7. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
8. assemblerflow. assemblerflow/flowcraft. *GitHub* Available at: https://github.com/assemblerflow/flowcraft. (Accessed: 3rd April 2019)
9. Federico, A. *et al.* Pipeliner: A Nextflow-based framework for the definition of sequencing data processing pipelines. *bioRxiv* 476515 (2018). doi:10.1101/476515
10. Snakemake-Workflows. *GitHub* Available at: https://github.com/snakemake-workflows.

(Accessed: 3rd April 2019)

11. Köster, J. & Rahmann, S. Building and documenting workflows with python-based snakemake. in *German Conference on Bioinformatics 2012* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012).
12. Afgan, E. *et al.* The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res.* **46**, W537–W544 (2018).
13. Amstutz, P. *et al.* Common Workflow Language, v1.0. (2016). doi:10.6084/m9.figshare.3115156.v2
14. Grüning, B. *et al.* Practical Computational Reproducibility in the Life Sciences. *Cell Syst* **6**, 631–635 (2018).
15. da Veiga Leprevost, F. *et al.* BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* **33**, 2580–2582 (2017).
16. Möller, S. *et al.* Robust Cross-Platform Workflows: How Technical and Scientific Communities Collaborate to Develop, Test and Share Best Practices for Data Analysis. *Data Science and Engineering* **2**, 232–244 (2017).
17. Grüning, B. *et al.* Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods* **15**, 475 (2018).
18. O'Connor, B. D. *et al.* The Dockstore: enabling modular, community-focused sharing of Docker-based genomics tools and workflows. *F1000Res.* **6**, 52 (2017).
19. Peltzer, A. *et al.* EAGER: efficient ancient genome reconstruction. *Genome Biol.* **17**, 60 (2016).
20. Travis CI - Test and Deploy Your Code with Confidence. Available at: https://travis-ci.org/. (Accessed: 21st January 2019)

Acknowledgments

The authors would also like to thank the following people for their contributions: C Wang, S Paneerselvam, F Bonath, O Contreras-López, S Haglund, R Hammarén, D Yuen, J Wan, M Proks, A Jemt, RA Olsen, J Westholm (Science for Life Laboratory, Stockholm, Sweden); Chi Chuan (A*STAR Genome Institute of Singapore, Singapore, Singapore); M Hoeppner (Institut für Klinische Molekularbiologie, Kiel, Germany); V Malladi (University of Texas Southwestern Medical Center, Dallas, USA); A Duncan (Ontario Institute for Cancer Research, Canada); H Gourlé (Swedish University of Agricultural Sciences, Uppsala, Sweden); C Mohr, D Straub, T Koch, G Gabernet (Quantitative Biology Centre, Tübingen, Germany); Q Zhao (Sun Yat-sen University Cancer Center, Guangzhou, China); and E Floden (Centre for Genomic Regulation (CRG), Barcelona, Spain). AP, SF and SN acknowledge funding from Deutsche Forschungsgemeinschaft (core facilities initiative, KO-2313/6-1 and KO- 2313-2, Institutional Strategy of the University of Tübingen, ZUK 63). AP and SN acknowledge funding by the Sonderforschungsbereich SFB/TR 209 “Liver cancer” of the Deutsche Forschungsgemeinschaft (DFG).

Supplementary Materials

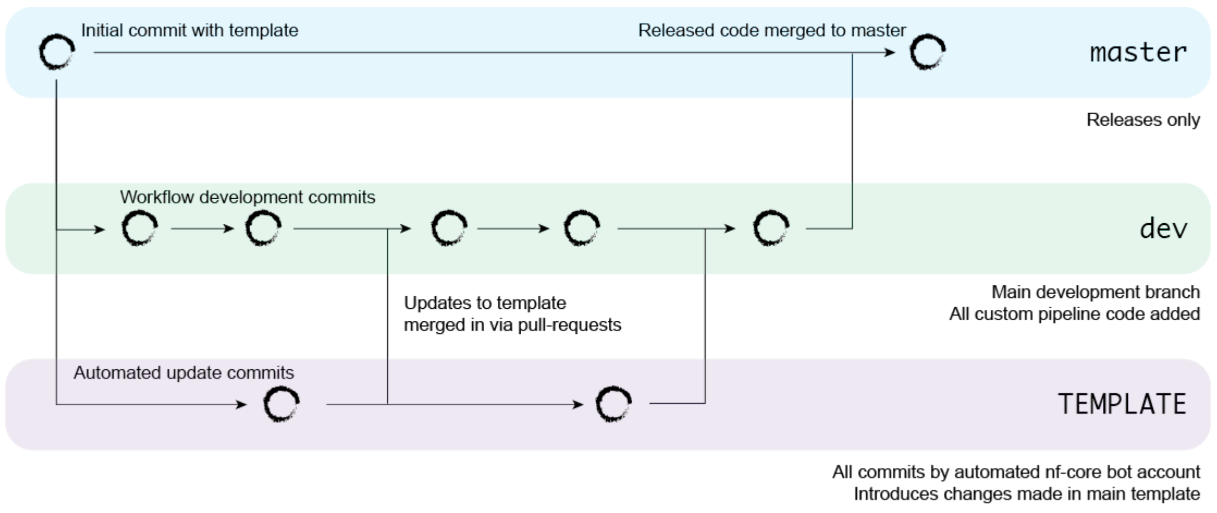

Initial commit with template
Released code merged to master
master
Releases only
Workflow development commits
dev
Updates to template
merged in via pull-requests
Main development branch
All custom pipeline code added
Automated update commits
TEMPLATE
All commits by automated nf-core bot account
Introduces changes made in main template

Figure S1: Template synchronization works using Git branching: the first commit for a pipeline is created immediately after using the template creation command and is branched into a special `TEMPLATE` branch. This remains untouched by the developer to keep the template commit history clean. Every time a new helper tool version gets released the continuous integration script automatically creates the pipeline again using the new version of the template. This is committed to the `template` branch and a pull-request to the pipeline's development branch is automatically created with a Github bot account. This pull-request will only contain the relevant diff produced by the updated template. By merging the pull-request the changes in the pipeline's implementation remain untouched but framework changes are integrated.