TAFFISH: A lightweight, modular, and containerized workflow framework for reproducible bioinformatics analyses

Kaiyuan Han[1], Ting Wang[1], Shi-Shi Yuan[1], Cai-Yi Ma[2], Wei Su[1], Kejun Deng[1], Hao Lv[1,*], Hao Lin[1,*]

[1] School of Life Science and Technology and Center for Informational Biology, University of Electronic Science and Technology of China, Chengdu 610054, China

[2] School of Computer Science and Technology, Aba Teachers College, Wenchuan, 623002, China

* To whom correspondence should be addressed.

Hao Lin: hlin@uestc.edu.cn

Hao Lv: hao.lyu@uestc.edu.cn

Abstract

Bioinformatics workflows often involve multiple tools and steps, with complex dependencies and environment-specific differences, making results difficult to reproduce across different systems. TAFFISH introduces a moderately engineered framework positioned between raw shell scripts and heavyweight workflow systems. It encapsulates each analysis tool, together with its fixed runtime environment and software version, into a modular, "Lego-like" component. It employs a lightweight, shell-native domain-specific language (DSL) to balance standardized environments with flexible workflow design. Each module in TAFFISH consists of a container image hosted on GitHub Packages and a corresponding script, and the system automatically pulls and runs the appropriate container when invoked through a unified interface. This design ensures a consistent interface, environment, and analysis results across Linux, macOS, and Windows (via WSL), significantly reducing cross-platform configuration overhead. We have developed a library of standardized modules for more than 60 widely used bioinformatics tools in the TAFFISH-Hub repository. As a demonstration, we applied a BLAST-based analysis of *Arabidopsis thaliana* P450 protein sequences and executed the same workflow on four different hardware/OS environments, obtaining identical results in all cases. This confirms that TAFFISH achieves high portability and reproducibility while preserving the flexibility of shell scripting. TAFFISH establishes a novel paradigm for constructing reusable, modular bioinformatics workflows, serving as an essential bridge between ad-hoc shell scripts and complex workflow frameworks, and enabling researchers to rapidly build and share reproducible analysis pipelines.

Keywords TAFFISH, bioinformatics workflow, reproducible, containerization

Introduction

With the rapid development of high-throughput sequencing and multi-omics technologies, the scale and complexity of bioinformatics analyses have expanded dramatically. Typical workflows often integrate multiple command-line tools and libraries across diverse operating systems and hardware platforms. However, this multi-tool, multi-dependency nature frequently leads to inconsistent results when the same workflow is executed in different environments, posing a significant challenge to reproducibility in scientific research[1,2]. Therefore, improving the portability and consistency of analyses while preserving flexibility has become an urgent problem for the bioinformatics community.

To address these challenges, researchers have proposed two main strategies. First, tool-level containerization projects such as Bioconda[5] and BioContainers[6,7] provide pre-built container images to standardize software environments, significantly reducing installation and dependency management overhead. While these approaches improve reproducibility at the single-tool level, they do not directly support the organization, reuse, or sharing of complete analysis workflows. Second, workflow management frameworks[8,9] such as Snakemake[10], Nextflow[11], and Galaxy[12–14] integrate containerization with job scheduling and dependency resolution to provide automated, large-scale workflow management[15,16]. These frameworks are robust and widely adopted in enterprise settings and large research teams. However, they typically require mastering a specialized domain-specific language (DSL)[17] or maintaining complex configuration files, which imposes a steep learning curve and long development cycle for individual researchers and small labs.[2]

Between these two extremes, there remains a critical need for an intermediate solution that combines the flexibility of raw shell scripting with the reproducibility and modularity of container-based environments. To meet this need, we developed TAFFISH (Tools And Flows Framework Intensify SHell), a lightweight and modular framework that packages each bioinformatics tool as an independent "Lego-like" module (a TAF-App) containing a fixed-version container image and an accompanying script. TAFFISH provides a unified invocation interface through a shell-native DSL, enabling users to quickly assemble, reuse, and share analysis workflows without concern for dependency conflicts or environment setup.

Leveraging the centralized module repository TAFFISH-Hub, researchers can achieve consistent results across different systems and platforms, significantly reducing development and deployment overhead.

Thus, TAFFISH is not intended to replace existing frameworks, but rather to fill the gap between ad-hoc scripting and heavyweight workflow systems. It offers a more direct, reproducible, and portable analysis paradigm for individual developers and small teams (Figure 1).

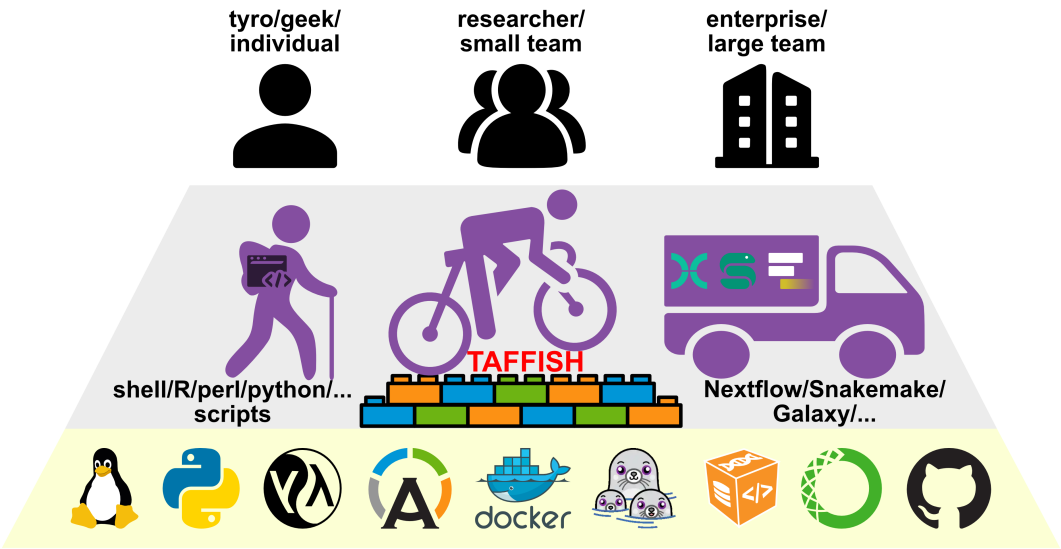

Figure 1. Ecological positioning of TAFFISH in bioinformatics workflow management.

The diagram highlights three layers: infrastructure (programming languages and container technologies), tools (scripts, TAFFISH, industrial systems), and users (beginners, research teams, enterprises). The horizontal axis represents the transition from flexibility to industrialization. TAFFISH occupies the middle niche, bridging simple scripting and industrial-scale systems to meet the needs of research teams.

Methods

In TAFFISH, each bioinformatics tool is encapsulated as an independent standardized component, termed a TAF-App. Each TAF-App contains a Docker[18] container image (with a fixed software version) and a corresponding .taf script. The container image ensures a consistent runtime environment, while the .taf script defines how to invoke the tool within that standardized environment. Through this dual encapsulation of the tool and its environment,

users obtain an identical usage experience and identical analysis results across different operating systems and hardware platforms.

To lower the learning curve, TAFFISH provides a lightweight, shell-based DSL called TAFFISH-Language[19]. The syntax of this DSL closely mirrors standard shell scripting, with only minimal enhancements. For example, a <container:...> tag explicitly binds a specific container image version and automatically loads that image. This design means users require almost no additional learning: they can call any packaged tool directly within a .taf file. For instance, in a BLAST workflow, the following snippet demonstrates how to build a database and perform a sequence alignment, and all container downloading, mounting, and execution are handled automatically by the system:

```
+FLOW:blastp-with-db
ARGS
  <dbout>
    ::*WORKDIR*::my-blast-db
RUN
  <container:taf-blast-v2.16.0:ghcr.io/taffish-org/blast:2.16.0>
    makeblastdb -in ::dbin:: -dbtype prot -title my-blast-db -out ::dbout::
    blastp -query ::in:: --db ::dbout::
```

In the snippet above, the RUN section is identical to standard shell commands, with containerization introduced only via the <container:...> tag. The ARGS block enables automatic recognition of command-line parameters, which can then be referenced in the script using placeholders like ::xxx::. For advanced users, TAFFISH-Language also supports complex parameter passing, default values, and other advanced features. The full syntax is documented in the official manual.

To facilitate cross-system tool reuse, TAFFISH provides a unified module repository called TAFFISH-Hub, hosted on GitHub Packages, which centrally stores all container images and .taf scripts. Users can manage TAF-App modules via the taf command-line tool, including installation, updates, removal, and searching, thereby avoiding the complexity of manually downloading images or handling dependencies. For example, running taf install BLAST will automatically fetch the appropriate BLAST container image and configure its script in the background, allowing the user to invoke the BLAST module through the standard interface without any manual setup.

At workflow runtime, the TAFFISH compiler parses the .taf file and translates the user-defined analysis pipeline into standardized shell commands, automatically inserting container launch and path-mapping details. The execution engine then pulls the specified container image and runs each task, generating output files, console output, and run logs. This process is transparent to the user and requires no manual environment configuration. This way, TAFFISH integrates workflow construction, tool encapsulation, container invocation, and result management into a single unified system, allowing analyses to be carried out in a highly controlled and reproducible environment.

In summary, the TAFFISH framework consists of four core components: standardized module packaging, a lightweight shell-based DSL, unified module management, and a containerized execution engine. These components work together to enable "develop once, reproduce anywhere" across different platforms, significantly reducing development and deployment overhead. The system architecture and workflow are illustrated in Figure 2, where the module repository, compiler, and execution engine together form the core technical framework of TAFFISH.

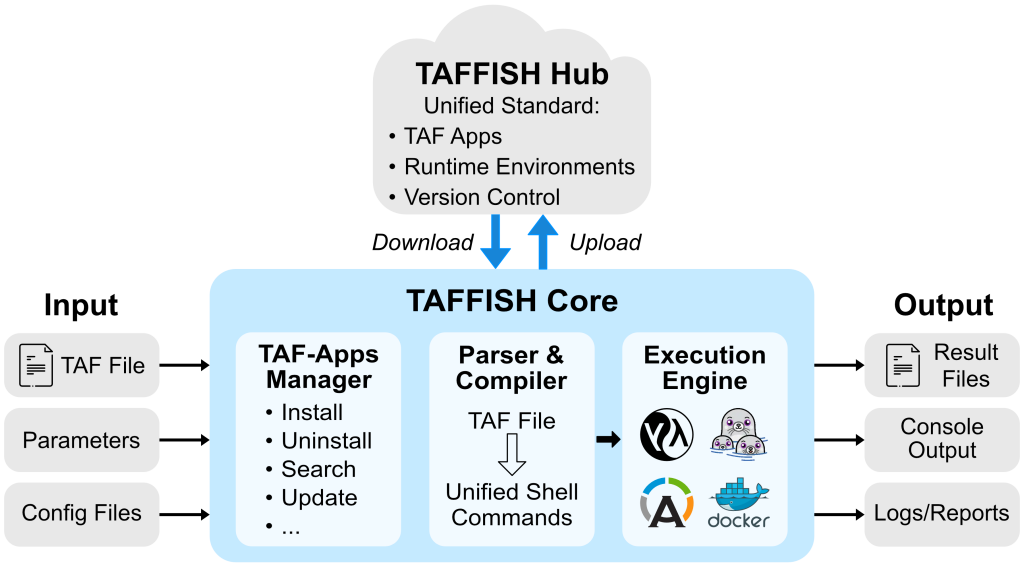

Figure 2. System architecture and workflow of TAFFISH.

TAFFISH leverages the TAFFISH-Hub to provide standardized workflow modules, runtime environments, and version control, enabling cross-platform portability and reproducibility. Users define workflows via .taf files, parameters, and configuration files. The core components include a module manager (for installing, updating, and searching modules), a parser & compiler (converting .taf files into unified shell commands), and an execution engine (running tasks in containerized environments). Workflow outputs include result files, console output, and log files.

Results

To evaluate TAFFISH's usability and extensibility in real-world bioinformatics tasks, we have built a standardized module library on TAFFISH-Hub containing over 60 mainstream bioinformatics tools as of this submission. This library covers significant steps of multi-omics analysis, including raw data quality control (FastQC[20], fastp[21]), alignment and quantification (Bowtie2[22], BWA[23], STAR[24], Kallisto[25], Salmon[26]), variant calling (GATK[27], FreeBayes[28]), 3D genomics (Juicer[29], HapHiC[30]), gene family and sequence analysis (BLAST[31], HMMER[32], CD-HIT[33,34], MMseqs2[35,36]), and structural biology and molecular simulation (PyMOL[37], AutoDock Vina[38], GROMACS[39,40], Rosetta[41]). Users can invoke any packaged tool via the unified interface through this modular ecosystem, enabling reproducible analyses across platforms.

To further demonstrate TAFFISH's portability and reproducibility, we constructed a BLAST-based workflow to identify members of a plant P450 protein family (Figure 3). First, we collected 271 known P450 protein sequences from *A. thaliana* and used TAFFISH's taf-blast module (invoking makeblastdb) to construct a P450 protein database[42–44]. We then took 22,245 full-length protein sequences from *Marchantia polymorpha* as a query set and performed blastp searches against the database to retrieve candidate P450 proteins[45]. The matched *M. polymorpha* protein IDs were extracted and deduplicated using standard shell commands (cut, sort, uniq), yielding 159 candidate P450 proteins.

We executed this analysis workflow on four independent hardware/OS environments: (1) Debian 12 Linux (AMD64), (2) Ubuntu 22.04.1 Linux (AMD64), (3) macOS (ARM64, Apple M2 chip), and (4) Windows via WSL (Debian 12, AMD64). On all platforms, without any dependency installation or environment adjustments, running the same .taf workflow got the exact same alignment results and produced an identical set of 159 candidate protein IDs. This demonstrates that through containerization and modular management, TAFFISH successfully eliminates the impact of hardware architecture and operating system differences on analysis outcomes, achieving consistent cross-platform results.

As shown in Figure 3, the workflow starts from known *A. thaliana* P450 protein sequences and the complete *M. polymorpha* proteome. It then uses TAFFISH's taf-blast module to build the database, perform sequence alignment, and output a list of candidate P450 protein IDs. Notably,

TAFFISH automatically manages all container images and dependencies during this process. As a result, users do not need to install or configure any additional software, which significantly lowers the technical barrier for developing and reproducing workflows across platforms.

Compared to a traditional pure-shell approach, TAFFISH significantly reduces development and maintenance overhead while preserving the flexibility of shell scripting. In the case above, the workflow code was about 90% shorter than an equivalent shell script, and manual environment configuration steps were almost eliminated. More importantly, users do not have to worry about OS discrepancies, dependency conflicts, or inconsistent tool versions, giving TAFFISH distinct advantages for multi-platform analyses, collaborative research, and workflow sharing.

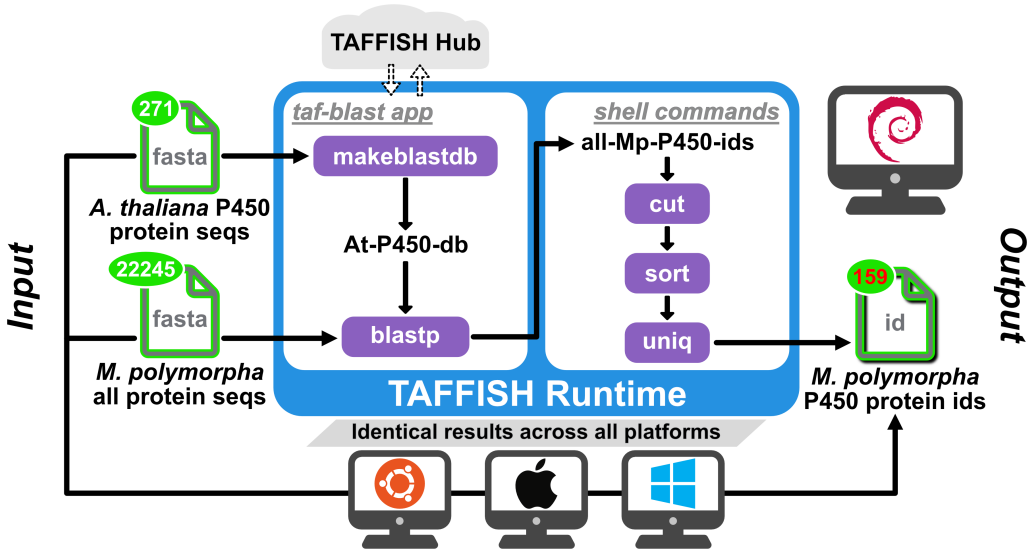

Figure 3. Cross-platform P450 protein identification workflow based on taf-blast.
Two hundred seventy-one known P450 protein sequences from *Arabidopsis thaliana* and 22,245 complete protein sequences from *Marchantia polymorpha* were used as input FASTA files. Within the TAFFISH runtime, the taf-blast app was used to construct a P450 protein database (via makeblastdb), followed by blastp searches against the *M. polymorpha* proteome to identify candidate P450 proteins. Subsequently, simple shell commands (cut, sort, uniq) were employed to extract and deduplicate the matched MP protein IDs, yielding a final set of 159 *M. polymorpha* P450 protein IDs. This workflow achieved seamless portability and produced identical results across Linux (Debian 12 and Ubuntu 22.04.1), macOS, and Windows (WSL with Debian 12), demonstrating the high reproducibility and cross-platform compatibility of TAFFISH.

Discussion

TAFFISH represents a moderately engineered analysis paradigm between traditional shell scripts and heavyweight workflow frameworks. By binding each bioinformatics tool with its runtime environment and encapsulating it as a standardized module (and invoking modules under a unified DSL), TAFFISH retains the high flexibility of shell scripting while resolving cross-platform portability and result consistency issues. Compared to conventional scripting that relies on users to manually configure environments, TAFFISH offers a more reliable way to execute workflows. Meanwhile, compared to powerful frameworks like Snakemake[10] or Nextflow[11] that have steep learning curves, TAFFISH's lightweight design lowers the entry barrier, making it suitable for the everyday workflow development needs of individual researchers and small teams.

TAFFISH's core advantage lies in its distinct methodological positioning relative to existing workflow management tools. Projects like BioContainers[6,7] and nf-core[46] address tool-level reproducibility by providing standardized container images, but they do not tackle workflow organization and automation. Frameworks such as Snakemake[10], Nextflow[11], and Galaxy[12–14] offer powerful features for task scheduling, dependency resolution, and distributed computing. Still, they require significant effort to learn and often to write complex rule files or maintain a specialized DSL. TAFFISH does not aim to replace these frameworks but rather complements the ecosystem between them. It encapsulates tools as reusable "Lego-like" modules and provides a unified invocation method via a shell-native DSL. As a result, users can compose analysis workflows as easily as stacking building blocks. This positioning lowers the technical barrier to workflow construction and offers a new solution for cross-platform collaboration and tool reuse.

In the broader context of reproducibility research, the significance of TAFFISH extends beyond technical implementation and influences the bioinformatics tool ecosystem. The modular design enables analysis tools to be packaged and shared independently, promoting reuse and reducing redundant redevelopment. The unified container images and scripting interface provide a solid foundation for cross-team collaboration, allowing researchers in different labs on different platforms to reproduce each other's workflows directly. Additionally, TAFFISH's

lightweight extensions to shell syntax at the DSL level make workflow files more interoperable across different workflow systems, raising the possibility of compatibility with mainstream standards like Common Workflow Language (CWL)[47] and Workflow Description Language (WDL)[48] in the future. These features position TAFFISH as a bridge between the tool-level and workflow-level ecosystems, helping to advance the standardization and modularization of bioinformatics analysis.

Nevertheless, TAFFISH has certain limitations. First, the number of modules currently available in TAFFISH-Hub is limited. The tool ecosystem needs to be continuously expanded to cover more analysis scenarios. Second, while containerization improves reproducibility, it introduces some storage overhead. Although runtime performance is nearly comparable to native execution, resource management should still be considered for large-scale deployments. In addition, TAFFISH is presently command-line oriented, which poses a usability hurdle for researchers without a computational background. To address these issues, we plan to implement two significant improvements in future work. First, we will develop a graphical user interface (GUI) to help users construct workflows through visual drag-and-drop and parameter configuration. Second, we aim to integrate AI assistants to automatically generate .taf files and recommend parameter settings, further reducing the complexity of workflow design. We will also work to improve compatibility with existing workflow standards (such as CWL[47], WDL[48], and nf-core[46]) to enable better interoperability across different platforms and ecosystems.

In summary, TAFFISH provides a lightweight, reproducible analysis solution for individuals and small teams, balancing flexibility and standardization. It does not replace existing workflow systems, but instead fills the gap between raw scripts and heavyweight frameworks. Through modular encapsulation, a unified interface, and containerized execution, TAFFISH offers new approaches to cross-platform consistency, workflow reusability, and collaborative research, and opens a new methodological paradigm for developing bioinformatics workflows.

Data and Software Availability

The TAFFISH software (executable), documentation, and container images are all publicly available. The executable can be obtained from the project's GitHub repository. User documentation is provided on the official website (https://taffish.com). All TAF-App modules and core container images are hosted on GitHub (https://github.com/taffish-org). The example data and .taf scripts mentioned in this paper are also available on GitHub (see the repository at taffish-org/examples, under the folder blast-Mp-AtP450). TAFFISH is currently available only for academic and non-commercial use, and a related patent is under application.

Reference

1. Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature* **533**, 452–454 (2016).
2. Cohen-Boulakia, S. *et al.* Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems* **75**, 284–298 (2017).
3. Markowetz, F. Five selfish reasons to work reproducibly. *Genome Biol* **16**, 274 (2015).
4. Grüning, B. *et al.* Practical Computational Reproducibility in the Life Sciences. *Cell Syst* **6**, 631–635 (2018).
5. Grüning, B. *et al.* Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat Methods* **15**, 475–476 (2018).
6. da Veiga Leprevost, F. *et al.* BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* **33**, 2580–2582 (2017).
7. Bai, J. *et al.* BioContainers Registry: searching bioinformatics and proteomics tools, packages, and containers. *J Proteome Res* **20**, 2056–2061 (2021).
8. Reiter, T. *et al.* Streamlining data-intensive biology with workflow systems. *Gigascience* **10**, giaa140 (2021).
9. Djaffardjy, M. *et al.* Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. *Comput Struct Biotechnol J* **21**, 2075–2085 (2023).
10. Köster, J. & Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* **28**, 2520–2522 (2012).
11. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. *Nat Biotechnol* **35**, 316–319 (2017).
12. Giardine, B. *et al.* Galaxy: A platform for interactive large-scale genome analysis. *Genome Res* **15**, 1451–1455 (2005).
13. Goecks, J., Nekrutenko, A., Taylor, J., & The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* **11**, R86 (2010).
14. Blankenberg, D. *et al.* Galaxy, a web-based genome analysis tool for experimentalists. *Curr Protoc Mol Biol* **0 19**, Unit-19.1021 (2010).
15. Leipzig, J. A review of bioinformatic pipeline frameworks. *Brief Bioinform* **18**, 530–536 (2017).
16. Wratten, L., Wilm, A. & Göke, J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nat Methods* **18**, 1161–1168 (2021).
17. Fowler, M. *Domain-Specific Languages*. (Pearson Education, 2010).
18. Boettiger, C. An introduction to Docker for reproducible research. *SIGOPS Oper. Syst. Rev.* **49**, 71–79 (2015).
19. Ahmed, A. E. *et al.* Design considerations for workflow management systems use in production genomics research and the clinic. *Sci Rep* **11**, 21680 (2021).
20. Simon Andrews *et al.* FastQC: A Quality Control tool for High Throughput Sequence Data. Babraham Bioinformatics (2010).
21. Chen, S., Zhou, Y., Chen, Y. & Gu, J. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* **34**, i884–i890 (2018).
22. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat Methods*

9, 357–359 (2012).
23. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).
24. Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013).
25. Bray, N. L., Pimentel, H., Melsted, P. & Pachter, L. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol* **34**, 525–527 (2016).
26. Patro, R., Duggal, G., Love, M. I., Irizarry, R. A. & Kingsford, C. Salmon: fast and bias-aware quantification of transcript expression using dual-phase inference. *Nat Methods* **14**, 417–419 (2017).
27. McKenna, A. *et al.* The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).
28. Garrison, E. & Marth, G. Haplotype-based variant detection from short-read sequencing. Preprint at https://doi.org/10.48550/arXiv.1207.3907 (2012).
29. Durand, N. C. *et al.* Juicer provides a one-click system for analyzing loop-resolution Hi-C experiments. *Cell Syst* **3**, 95–98 (2016).
30. Zeng, X. *et al.* Chromosome-level scaffolding of haplotype-resolved assemblies using Hi-C data without reference genomes. *Nat. Plants* **10**, 1184–1200 (2024).
31. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *Journal of Molecular Biology* **215**, 403–410 (1990).
32. Potter, S. C. *et al.* HMMER web server: 2018 update. *Nucleic Acids Res* **46**, W200–W204 (2018).
33. Li, W. & Godzik, A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* **22**, 1658–1659 (2006).
34. Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152 (2012).
35. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol* **35**, 1026–1028 (2017).
36. Mirdita, M., Steinegger, M. & Söding, J. MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics* **35**, 2856–2858 (2019).
37. PyMOL. Schrodinger, LLC.
38. Trott, O. & Olson, A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry* **31**, 455–461 (2010).
39. Berendsen, H. J. C., van der Spoel, D. & van Drunen, R. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications* **91**, 43–56 (1995).
40. Abraham, M. J. *et al.* GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **1–2**, 19–25 (2015).
41. Leaver-Fay, A. *et al.* Rosetta3: An Object-Oriented Software Suite for the Simulation and Design of Macromolecules. *Methods Enzymol* **487**, 545–574 (2011).
42. Paquette, S. M., Jensen, K. & Bak, S. A web-based resource for the Arabidopsis P450, cytochromes b5, NADPH-cytochrome P450 reductases, and family 1 glycosyltransferases (http://www.P450.kvl.dk). *Phytochemistry* **70**, 1940–1947 (2009).

43. Jensen, K. & Møller, B. L. Plant NADPH-cytochrome P450 oxidoreductases. *Phytochemistry* **71**, 132–141 (2010).
44. Lamesch, P. *et al.* The Arabidopsis Information Resource (TAIR): improved gene annotation and new tools. *Nucleic Acids Res* **40**, D1202–D1210 (2012).
45. Bolser, D., Staines, D. M., Pritchard, E. & Kersey, P. Ensembl Plants: Integrating Tools for Visualizing, Mining, and Analyzing Plant Genomics Data. *Methods Mol Biol* **1374**, 115–140 (2016).
46. Ewels, P. A. *et al.* The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol* **38**, 276–278 (2020).
47. Crusoe, M. R. *et al.* Methods included: standardizing computational reuse and portability with the Common Workflow Language. *Commun. ACM* **65**, 54–63 (2022).
48. Auwera, G. V. der. Deciphering a mystery workflow written in WDL. *Terra* https://terra.bio/deciphering-a-mystery-workflow-written-in-wdl/ (2022).