Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

YR-MPE: a modular desktop platform for agent-operable and reproducible phylogenetic workflows

Zhi-Jie Xu
Donghua University https://orcid.org/0009-0003-3855-0921
Yi-Yang Jia
Shanghai Jiao Tong University https://orcid.org/0009-0001-9556-9501
Zhi-Han Gu
Donghua University
Ming-Rui Shang
Donghua University
Xin-Bo Cheng
Donghua University
Shuo Han
Donghua University
Yang Jin
Donghua University
Yi-Tong Lei
Donghua University
Rui Zhang
Donghua University https://orcid.org/0000-0002-0451-2375
Yunlong Zhang
zhyl@dhu.edu.cn
Donghua University https://orcid.org/0009-0006-5971-4157

Method Article

Keywords: phylogenetics, reproducible workflows, graphical user interface, scientific agents, provenance, Windows

Posted Date: November 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11771753/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

YR-MPE: a modular desktop platform for agent-operable and reproducible phylogenetic workflows

Zhijie Xu[1], Yiyang Jia[3], Zhihan Gu[1], Mingrui Shang[1], Xinbo Cheng[1], Shuo Han[1], Yang Jin[1], Yitong Lei[1], Rui Zhang[2], Yunlong Zhang[1*]

[1]Department of Biological Sciences and Technology, College of Biological Science and Medical Engineering, Donghua University, Shanghai, 201620, China.

[2]Department of Cardio-Pulmonary Circulation, Shanghai Pulmonary Hospital, Tongji University School of Medicine, Shanghai, 200433, China.

[3]Shanghai Center of Systems Biomedicine, Shanghai Jiao Tong University, Shanghai, 200240, China.

*Corresponding author(s). E-mail(s): zhyl@dhu.edu.cn;

Abstract

Phylogenomic analyses combine many tools, file formats and execution modes. Desktop graphical user interfaces make individual steps accessible but hide their state from scripts; command-line pipelines make steps scriptable but leave parameter discovery, provenance and recovery to the user; and emerging LLM agents can call command-line tools but cannot inspect a GUI session or its artifacts. We present YR-MPE, a modular desktop platform whose YPhylo Workbench unifies these modes through a typed workspace, a plugin contract and a semantic action layer. Every operation in the action registry is exposed as a machine-callable action; controls without a CLI equivalent are declared GUI-only. The workspace records every imported plugin result as a run with its inputs, parameters, tool identity and output artifacts, the latter with SHA-256 hashes; each compiled workflow step becomes its own run. A placeholder compiler turns JSONL command sequences into reproducible workflow DAGs, while a skill layer maps

each declared control to the corresponding command-line option; plugins with tool-specific contracts expose tool-specific actions. YR-MPE runs natively on Windows and bundles alignment, model selection, maximum-likelihood, Bayesian and species-tree inference, molecular dating, selection and visualization tools. We describe the interface and its provenance/workflow substrate, and report three representative recipes: an organelle-genome workflow, a single-locus model-to-tree analysis, and an agent-driven run. YR-MPE is released under GPL-3.0 at https://github.com/Gipsy-The-Sheller/YR-MPE.

Keywords: phylogenetics, reproducible workflows, graphical user interface, scientific agents, provenance, Windows

1 Introduction

A modern phylogenetic study is rarely a single analysis [1]. A typical project starts with annotated genomes, extracts coding sequences, groups orthologues [2], aligns each locus, trims ambiguous columns [3], selects a substitution model, infers gene trees [4, 5], and finally reconstructs a species tree or a dated tree [6–9]. A project may also include molecular dating, site-heterogeneous mixture models [10], or species-tree inference under gene-tree discordance [11–17]. Each step may use a different program, file format, and parameter vocabulary. The resulting burden is not only compute time but coordination: locating inputs, remembering model choices, and reconstructing what was done when results are revisited months later.

Desktop graphical user interfaces (GUIs) have made individual steps accessible; MEGA, PhyloSuite, raxmlGUI and BEAST 2 are familiar examples [11, 12, 18–22]. Recent hybrid tools such as eGPS-oneBuilder combine a GUI with a CLI for tree construction and multi-tree interpretation [23]. Workflow systems such as Galaxy, Nextflow and Snakemake address batch execution and reproducibility [24–26]. These tools cover parts of the workflow, but their interfaces remain separate: a GUI session is not a script, a command-line pipeline does not know what a GUI has selected, and a workflow engine does not expose the parameter semantics of an interactive plugin. The gap is especially visible for an external agent: a command-line tool is callable, but callable is not necessarily operable, and a GUI contains the necessary state but only as widgets and pixels. Current AI-for-science systems often expose scientific tools as APIs or services [27–29], and typically do not inherit the control semantics or provenance of a desktop application. The question is therefore how to expose one semantic operation layer to human, agent and workflow clients.

We present YR-MPE, a modular desktop platform designed around that question. Its YPhylo Workbench combines a typed, provenance-backed workspace; a semantic action layer shared by GUI and external agents; and reproducible JSONL workflows that can be compiled into a previewable graph. Plugins declare their controls and result types, so a new tool can be added without changing the host. The distribution includes

native Windows builds of the analysis programs it drives, avoiding a separate Linux environment or container for the same tools. The plugin suite covers alignment and trimming, model selection, maximum-likelihood, parsimony and distance inference, Bayesian inference, species-tree and dating analyses, and downstream diagnostics. Our aim is to let a GUI user, a scripted workflow and an external agent perform the same analysis through the same interface, without duplicating the semantics or losing the provenance of the result.

2 Results

2.1 Design overview

YR-MPE separates the platform into a plugin layer, a typed workspace and a workbench. Plugins own a tool dialog and the logic that turns user choices into a command; the workspace owns data items, selection state and provenance; the YPhylo Workbench owns the menu, the workspace view and the side panels. The separation keeps each layer small: a plugin owns its dialog and command, the workspace owns selection and provenance, and the workbench routes typed results without per-tool code. Figure 1 summarizes a default analysis path and its JSONL representation. Figure 2 groups the same operations into the eight menus of the workbench and shows the direction in which a typical analysis moves across them; the arrows show the recommended data-flow order. The same installation includes native Windows builds of the bundled programs, so the GUI, the command-line tools and the agent bridge refer to one runtime rather than three separately managed environments. The resulting interface differs from existing systems in the properties summarized in Table 1.

Property	YR-MPE	MEGA 12	PhyloSuite 2	Galaxy	raxmlGUI 2
Desktop GUI	✓	✓	✓	✗	✓
Native Windows binaries	✓	✓	✓	✗	✓
Batch execution	✓	✓[a]	✓	✓	✗
Run definition + lineage	✓	✗	✗	✓	✗
Agent-operable semantic actions	✓	✗	✗	✗	✗
Reproducible workflow DAG	✓	✗	✗	✓	✗
Species-tree inference	✓	✗	✓	✗	✗
Molecular dating	✓	✓	✓	✓	✗
Plugin extension surface	✓	✗	✗	✓	✗

Table 1 Interface-level comparison with representative desktop and workflow systems. A check in the "run definition + lineage" row means the released interface stores each run's tool and parameters together with its input and output items in one reloadable workspace record; the reproducible workflow plan is a separate artifact. [a]MEGA-CC is the batch interface distributed with MEGA; Galaxy is web-based.

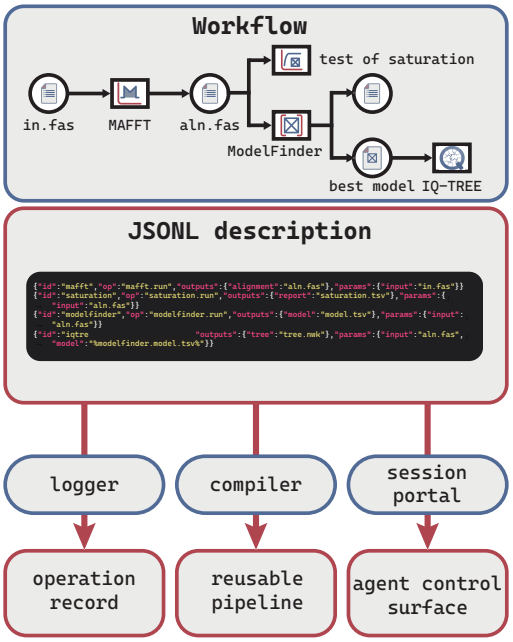


Fig. 1 Workflow and JSONL representation of a default analysis. Top: an input alignment is imported into MAFFT; the resulting alignment branches to substitution-saturation testing and ModelFinder, whose best model is passed to IQ-TREE 3. Bottom: the same steps as a short JSONL description. The three roles of the description are the logger (operation record), the compiler (reusable pipeline) and the session portal (agent control surface).

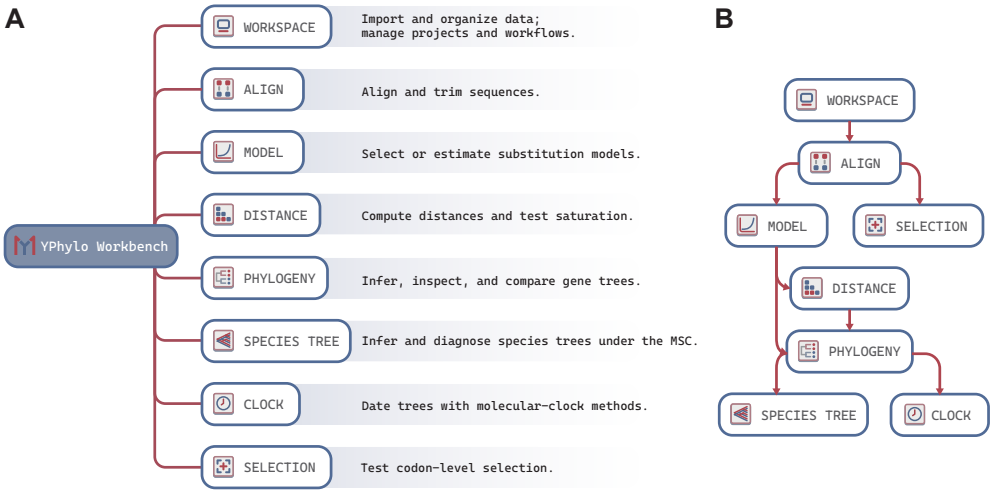


Fig. 2 Menu structure and workflow direction in YPhylo Workbench. (A) The eight menu groups, each with a one-line summary of the operations it contains. (B) Recommended direction of data flow across the groups: a data set enters through Workspace and proceeds through alignment, model selection or selection testing, distance calculation and phylogeny, with species-tree and clock analyses as terminal branches. The arrows indicate the recommended data-flow order; any tool can run once its typed inputs are available.

2.2 A typed workspace and a common plugin contract

Every plugin inherits a common lifecycle: import data, configure parameters, run a hidden subprocess or an in-process analysis, report progress, and emit a structured result. The result is a `ResultSpec` containing one or more datasets; each dataset contains typed items such as an alignment, model, distance matrix, phylogeny, tree set or chain. The host validates the spec and materializes the items without importing plugin-specific dataset code. This contract is what makes the GUI, the workspace and the agent interface agree on what an "alignment" or a "tree" is.

The workspace is a persistent project store rather than an in-memory session. It records datasets, items, runs, streams and imports. Every imported plugin result creates a run record that links its input items, parameters and output items, and retained artifacts are hashed. Direct file imports are recorded as imports rather than runs. The Inspector presents the same provenance graph as a human-readable view, and reopening the project restores the analysis state without manual reconstruction (Supplementary Note S4).

This design also gives the GUI a precise selection model. An item can be active (green), part of a multi-object selection (blue), or queued in the Selected basket. The sidebar recommendations are derived from the types in the basket: a tree set or at least two trees enables ASTRAL-IV, wASTRAL and RF comparison; one tree plus one tree set enables TreeRecs and RF; a single tree enables LSD2; raw sequences enable the four aligners; an alignment enables trimming, model selection, ML/BI/MP inference, distance calculation, selection tests and CASTER-site; a distance matrix enables DecentTree; and a chain enables MiniTracer. A model item modifies an alignment-based operation and participates when an alignment is also selected. The full mapping is listed in Supplementary Table S3 and the recommendation logic is described in Supplementary Note S8.

2.3 Provenance-backed dataset formation

The most demanding workflow in the release starts from annotated genomes. The One-Step OrthoFinder plugin accepts GenBank format (`.gb`, `.gbk` or `.gbff`), from which it extracts coding sequences, or protein FASTA in proteome mode. It runs OrthoFinder, identifies single-copy orthogroups, and imports the resulting loci as a dataset. The dataset can then be sent to MAFFT or MACSE2, trimmed with TrimAl or GBlocks, passed through ModelFinder or ModelTest-NG, and used for IQ-TREE or RAxML-NG inference. The important property is that the handoff between steps is a first-class workspace operation. Each output item points to the run that produced it, and each run points to the input items and the command that consumed them.

The three dataset-formation routes serve different users. The One-Step OrthoFinder plugin is the automated, agent-facing route: it is designed for batches of genomes and for workflows in which an agent can discover, configure and run the same steps as a person. SeqMatrix and SeqDBG are the human-facing routes. SeqMatrix targets non-genomic, fragmentary Sanger data: sequence files are shown in a file tree and

dragged into partition columns, so a dataset is assembled visually and exported without copying, pasting or renaming files by hand. SeqDBG reconciles divergent gene annotations and exports the reconciled dataset; it also shows annotation adjacency and gene rearrangement.

We use this workflow as the primary case study because it exercises the full handoff chain: multiple input files, a third-party binary with its own runtime, one-to-many result expansion, and a long chain of format conversions. In the task03 case study, ten annotated organelle GenBank files yielded 140 coding sequences, 14 single-copy orthogroups with one sequence from each sample, and a 10-taxon 11,109-bp concatenation; both the GUI and agent routes matched the hidden reference tree (RF 0; Table 2). The run retained the OrthoFinder output, per-orthogroup CDS files, the IQ-TREE report, log, model and split files, so the workspace could be reopened with the same artifact graph.

2.4 An agent-operable interface

The agent interface is a view of the same semantic actions as the GUI. A local, token-protected file bridge handles commands, responses and progress events as pure text (in `.jsonl` format). A GUI timer polls the bridge on the Qt main thread, so plugin actions always run in the same thread as the widgets they manipulate (Supplementary Note S5).

Before invoking anything, an agent can inspect four state surfaces:

- `workbench.menu` returns the toolbar menu tree and a flat list of actions with stable IDs, paths and enabled/checked state;
- `workbench.workspace` returns datasets, items, selection states, runs, streams and imports;
- `workbench.sidebar` returns the Selected basket, recommended operations and the current Inspector operations;
- `workbench.skills` and `workbench.help_doc` return the skill catalog and a live document for one plugin.

The live help document is the key to parameter-level parity. Executable-backed plugins declare `control_bindings()` for each control that has a stable command-line option; a tool with a richer semantic API, such as OrthoFinder, exposes tool-specific actions instead. For example, the ModelTest-NG binding for sequence type maps `DNA` to `-d nt`, and the RAxML-NG binding for bootstrap maps a replicate count to `--bs-trees`. Skills are stored as Markdown plus a manifest under `YR-MPE/skills/`; the live view adds current values, resolved values and a command preview. This makes the documentation executable enough to drive a tool and explicit enough to audit.

Once a plugin window is open, a generic action surface is available through `workbench.invoke_plugin`: `plugin.inspect_controls`, `plugin.modify_options`, `plugin.run`, `plugin.stop`, `plugin.inspect_console` and `plugin.export_to_workspace`. Tool-specific APIs, such as the OrthoFinder actions,

extend the generic surface. The agent then can start an analysis through semantic actions alone.

2.5 Reproducible workflow specifications and preview

A reusable workflow needs a path-independent representation. The workflow compiler accepts a JSONL specification with an optional header:

```
{"schema":"yrmpe.workflow.v1","kind":"workflow",
 "name":"demo","inputs":{"raw":"raw.fas"}}
{"id":"mafft","op":"mafft.run",
 "outputs":{"alignment":"work/aligned.fas"},
 "params":{"input":"raw.fas","output":"work/aligned.fas"}}
{"id":"raxml","op":"raxmlng.run",
 "outputs":{"tree":"work/tree.nwk"},
 "params":{"msa":"work/aligned.fas","output":"work/tree.nwk"}}
```

Before graph construction, the compiler drops read-only discovery actions and steps whose recorded response is a failure; these are useful in the agent transcript but are not workflow nodes. It then collects output declarations, assigns each output a symbolic placeholder such as `%mafft.alignment.fas%`, rewrites matching inputs, and validates the resulting DAG for duplicate producers, unknown placeholders and cycles. The same graph can be rendered as a workflow preview: data items are circles, tools are squares, and orthogonal arrows show the artifact flow. Clicking a tool node opens its plugin in workflow mode. Path inputs are locked to placeholders and the Run button becomes *Close & Apply*, so editing a step returns a configuration object instead of launching a second analysis inside the editor. The preview is therefore both a visual explanation and an editing surface for the same plan.

2.6 Tool coverage and representative recipes

The release covers four groups; Table 1 compares the interface with four representative desktop and workflow systems. Alignment and trimming include MAFFT [30], MUSCLE5 [31], Clustal Omega [32], MACSE2 [33], TrimAl [34] and GBlocks [35]. Model selection includes ModelFinder [36], ModelTest-NG [37] and model-parameter estimation. Tree inference includes IQ-TREE 3 [38], RAxML-NG [39], MrBayes [40], PhyloBayes-MPI [14], MPBoot [41] and DecentTree [42]. Species-tree methods include the supertree/quartet approaches ASTRAL-IV and wASTRAL in the ASTER package [15, 43], direct site-based inference with CASTER-site [16], and full-likelihood Bayesian inference with BEST, implemented through MrBayes [17]. For small datasets, BEST provides a full-likelihood Bayesian route through MrBayes. Dating and reconciliation are covered by LSD2 [44] and TreeRecs [45]. Visualization and diagnostics include IcyTree [46], MiniTracer and the sequence/dataset views. MiniTracer is a Python implementation of Tracer [47] with a redesigned interface layout to better guide MCMC diagnostics.

A practical distinction is native x86_64 Windows support. Several tools that are commonly distributed as UNIX-only binaries, most notably DecentTree and MPBoot, are provided here as native Windows builds. We place particular emphasis on MrBayes-MPI-BEAGLE: its Windows build couples MS-MPI with BEAGLE 3 [40, 48], so a Bayesian analysis can use MPI parallelism and BEAGLE acceleration on a standard Windows desktop without a Linux virtual machine or container. PhyloBayes-MPI is distributed in the same way.

The released skill manifest documents 41 plugins in eight menu groups (Supplementary Note S6 and Table S1). The set is broader than the examples above: it includes ASTRAL-IV/wASTRAL, CASTER-site, TreeRecs and BEST/MrBayes MSC for species trees, together with gCF/sCF concordance [49], site-level sCF/CASTER diagnostics and MSC branch diagnostics; PD-Guide, TreeMerger, IcyTree, MiniTracer and RF comparison for tree handling and diagnostics; SeqMatrix and SeqDBG for fragmentary and annotated data; and BaCoCa C-value saturation testing [50] alongside the codon Z-test of Nei and Gojobori [51] and the batch dN/dS tools of Yang and Nielsen [52]. The saturation plugin reports the partition-level C-value [53] alongside Ti/Tv versus p-distance scatter plots, so saturation is inspected directly rather than judged against a fixed exclusion threshold. Each documented plugin has a Markdown skill.

The gene-tree MSC workflow takes a fixed species tree and a set of gene trees. It scores every internal split of the fixed tree with IQ-TREE 3 `--gcf`, which reports gCF, gDF1, gDF2, gDFP and the number of decisive gene trees, and with ASTER [15]. ASTRAL-IV returns qCF, qDF1 and qDF2 for the same splits together with a coalescent-unit length, and wASTRAL returns a weighted coalescent-unit length [43]. The merged table keeps one row per split and adds the gCF-derived coalescent length $-\ln[(3/2)(1-\mathrm{gCF})]$ as `x_gcf`. The Table, Overview, Simplex and Bipartition info tabs are built from the merged rows; in the Simplex tab, gCF points are coloured by gDFP and ASTRAL-IV qCF points by their coalescent-unit length.

The site-level diagnostic plugin is a separate workflow from the gene-tree MSC diagnostics. It scores the same fixed species tree with IQ-TREE 3 likelihood sCF [54] and CASTER-site [16], then reports sCF/sDF1/sDF2 beside CASTER q1–q3; concordance factors describe topological variation rather than branch support [55]. Agreement among inference methods is a model-specific robustness check, not independent evidence; concordance factors instead quantify gene-tree and site-level variation under the multispecies coalescent. Every concordance table is paired with a bipartition membership matrix (taxa as rows, splits as columns), an overview plot and a ternary simplex view; the simplex construction follows Allman et al. [56], with related R tooling in MSCquartets [57]. The matrix keeps full taxon labels visible and shades the smaller side of each split, while the simplex shows the three-way concordance composition and encodes gDFP as colour for gCF, and a branch's length estimation (in coalescent unit) as colour for CASTER q or ASTRAL qCF values. The sampling error of the gCF-derived coalescent length is derived in Supplementary Note S10.

Three recipes exercise the design. The first is the organelle workflow above: GenBank → OrthoFinder → single-copy orthogroups → MAFFT/MACSE2 → TrimAl →

ModelTest-NG → RAxML-NG. The second is a single-locus analysis: alignment, ModelFinder, IQ-TREE 3 and IcyTree, with the model item passed automatically to the tree plugin. The third is agent-driven: the agent opens a tool through the menu tree, reads its live control map, sets the same parameters a GUI user would set, runs the tool, checks the console and exports the typed result. The three recipes share the same workspace and provenance rules. Figure 3 shows representative interface states drawn from these recipes, from dataset management and matrix assembly through tree display, concordance diagnostics and chain inspection to reconciliation, saturation and selection testing.

2.7 Open-task case studies

A fixed recipe tests execution. The open tasks test whether the interface helps when the method is not prescribed. We released four tasks with hidden references: a 50-locus species tree under incomplete lineage sorting, a three-amplicon ribosomal data set with indels and a missing partition, a 10-sample organelle data set with linked CDS features and shuffled gene order, and a 30-locus matrix with missing taxa and discordant loci. Each task supplied raw inputs and a time limit (60–120 min); the choice of gene trees, alignment, reference and method was left to the participant. Each run required a Newick tree and a short validation record, and was scored by taxon-set match, unrooted Robinson–Foulds (RF) distance and, for the missing-data task, recall of the hidden discordant loci. The task inputs, prompts and scoring script are retained in the distribution (Supplementary Note S7); the reference trees are withheld from the participants.

The author (GUI), GitHub Copilot (local bridge; using GPT 5.6-Luna) and Claude Code (DeepSeek backend) completed all four tasks with valid Newick trees and exact taxon sets. Copilot matched the reference on tasks 01 and 03 (RF 0) and was two splits from it on task 04 (RF 2; normalized 0.20). On task 02 Copilot produced a valid tree 16 splits from the reference (normalized 0.94); it aligned the reverse-complemented amplicons without direction adjustment. The author's run used `--adjustdirection` and recovered RF 0. Claude Code matched the reference on tasks 01–03 (RF 0) and was two splits from it on task 04 (RF 2; normalized 0.20); its task04 QC named all three discordant loci (3/3).

Claude Code's task02 run, after the skill update, inferred a consistent orientation assignment across each partition, reverse-complemented the minority records, and then aligned the corrected amplicons. A no-skill control with the same model and task recovered the same topology, so the case series does not isolate a skill-specific improvement.

On task 04 the three available trees were two splits from the reference; the author's and Claude Code's QC named all three discordant loci (3/3), and Copilot's named none.

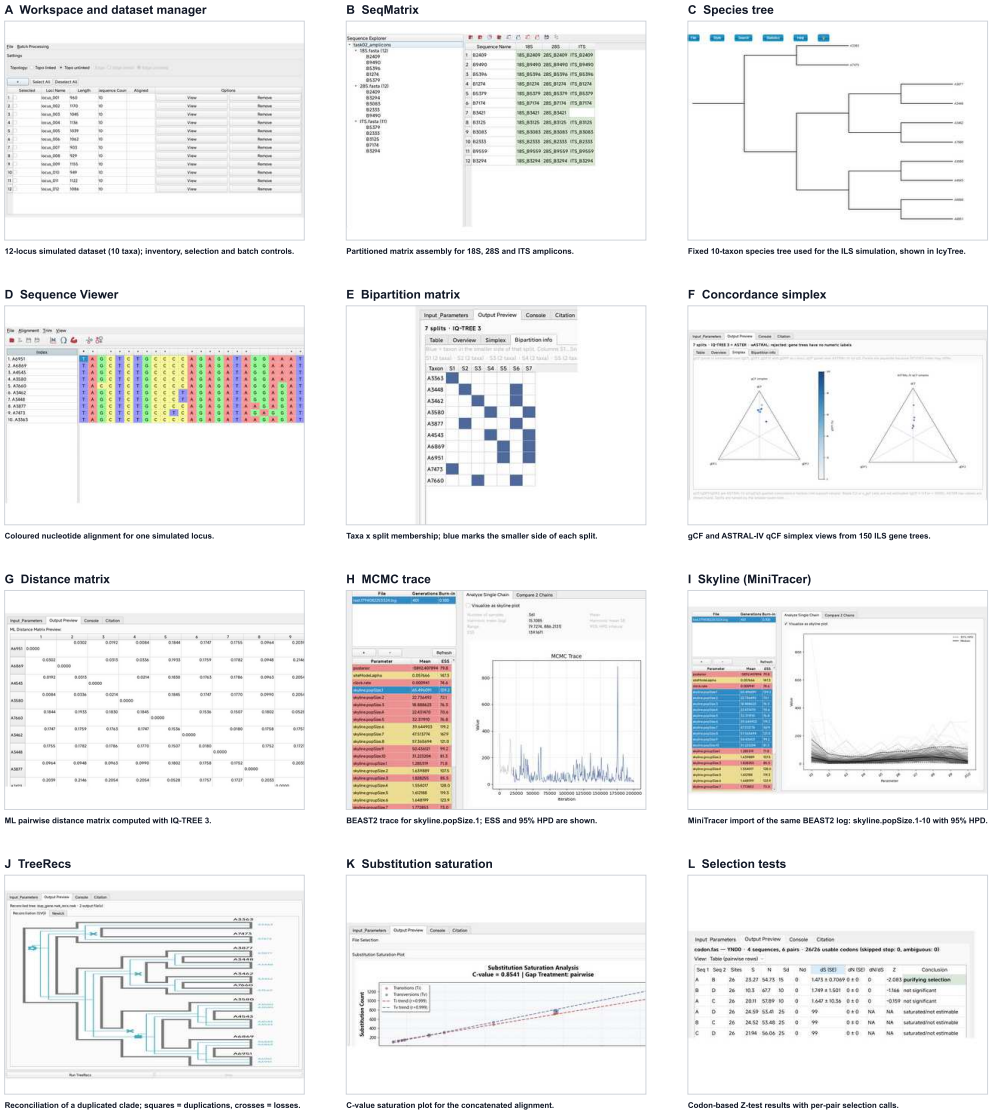


Fig. 3 Interface coverage of YR-MPE. (A) Dataset manager for a 12-locus, 10-taxon SimPhy/AliSim data set. (B) SeqMatrix assembly of 18S, 28S and ITS amplicons. (C) Fixed 10-taxon species tree used for the ILS simulation, displayed in IcyTree. (D) Sequence Viewer with nucleotide colouring. (E) Bipartition membership matrix; blue marks the smaller side of each split. (F) gCF and ASTRAL-IV qCF simplex views based on 150 msprime gene trees simulated under ILS. (G) ML pairwise distance matrix computed with IQ-TREE 3. (H) BEAST2 MCMC trace for skyline.popSize.1 from a 200,000-generation run. (I) MiniTracer import of the same BEAST2 log, showing the ten skyline.popSize parameters with 95% HPD. (J) TreeRecs reconciliation of a duplicated clade; squares mark duplications and crosses mark losses. (K) C-value substitution-saturation plot for the concatenated alignment. (L) Codon-based Z-test selection results.

		Unrooted RF			QC		
Task	Challenge	GUI user	Copilot	Claude Code	GUI user	Copilot	Claude Code
01	50 loci, ILS	0	0	0	–	–	–
02	ribosomal amplicons, indels	0	16 (0.94)	0	–	–	–
03	linked organelle contigs	0	0	0	–	–	–
04	missing taxa, discordant loci	2 (0.20)	2 (0.20)	2 (0.20)	3/3	0/3	3/3

Table 2 Open-task case studies. Every run produced a parseable unrooted tree with the expected taxon set. RF is the unrooted Robinson–Foulds distance to a hidden reference tree; normalized RF is shown in parentheses when nonzero. QC is the number of hidden discordant loci named in the task04 QC report; a dash means that the task did not require this report. The GUI column is the author using YPhylo Workbench; the Copilot column is GitHub Copilot operating the same local bridge; the Claude Code column is the DeepSeek-backed agent using the full skill surface. One run per cell.

The series shows that the same semantic surface can be driven by a person and by an agent, and that an agent can recover the reference topology without a prescribed pipeline.

2.8 Evaluation evidence

The organelle case study tests the workflow handoff and provenance graph (RF 0 for both the GUI and agent routes). The four open tasks provide interface evidence for a person and two agents (Table 2; protocol in Supplementary Note S7). The two-agent plugin pilot tests extensibility (Supplementary Note S9). The matched MrBayes runs test compute efficiency (Supplementary Note S2), and the workbench memory footprint is reported in Supplementary Note S1.

A separate development pilot asks one agent to implement a One-Step SibeliaZ dataset-formation plugin from the template and another agent to discover, run and validate it. In the pilot, the authoring agent passed all 13 structural checks and 17 unit tests on the isolated workspace. The usage agent then discovered the installed plugin through the menu and skill surfaces, ran the real 12-accession chloroplast fixture with `k=19` and `m=100`, exported 273 LCB items covering 2,633 copies, and passed all nine usage checks; the run completed in 4.2 s (Supplementary Note S9). SibeliaZ exercises block parsing and typed dataset materialisation, a contract the existing tree plugins do not cover. The protocols, datasets and scoring rules are described in Methods; the open-task scores are in Table 2.

3 Discussion

The central design choice in YR-MPE is to treat the GUI, the command line and an external agent as three clients of the same semantic layer. This avoids a common failure mode of automation work: a new scriptable interface is built beside the GUI, then diverges from it. In our implementation, a menu action, an agent command and a workflow step all resolve to a registered operation with a parameter schema.

The difference is the transport and the level of detail returned to the caller, not the operation itself. This is why the same result contract can feed a workspace item, a preview node and an agent response.

The local-first design has practical consequences. First, the desktop session is the unit of trust: the bridge is a local file channel or a token-protected local session, and commands are dispatched through registered actions. Second, the GUI thread remains the owner of widget state; agent commands are queued and dispatched by the workbench rather than touching Qt objects from a worker thread. Third, the workspace is inspectable outside the application: a user can archive it, compare manifests, or hand it to a colleague without exporting a proprietary project file.

Windows-native packaging is a deliberate trade-off. It removes the need for a Linux environment or a container for tools that are often distributed only as source, and it makes the same binaries available to the GUI, the workflow runner and the agent bridge. The cost is maintenance: each bundled binary and its runtime libraries must be audited when a tool changes version.

The workflow representation extends provenance with reproducibility. A run log records what happened; a placeholder plan records what should happen next, independent of a machine-specific path. Replaying the plan creates new run records and artifact hashes, so the intended workflow and the artifacts that actually exist can be compared.

3.1 Relationship to AI-for-science systems

The AI4S literature has largely optimized the path from a natural-language request to a scientific tool call [28, 29]. That path is one part of a desktop analysis environment. A workbench already has a typed object graph, validated parameters and provenance; an agent should inherit those. Conversely, provenance systems such as SciForge [58] and AIPOCH Open-Science [59] start from the research record and make evidence or artifact history first-class. YR-MPE is complementary: it starts from the interactive analysis tools and makes the GUI-to-CLI contract the unit of provenance. The combination suggests a useful division of labor: a desktop workbench owns tool semantics and local validation; an agent owns planning and cross-tool orchestration; a workflow layer owns replay and portability.

3.2 Relationship to existing tools

YR-MPE complements the individual analysis programs by making their interfaces interoperable. It targets the desktop case in which a user wants to inspect intermediate results, change a model, and keep the analysis auditable, while an agent can perform the same operations. Table 1 covers interface and provenance properties rather than algorithms.

eGPS-oneBuilder couples a GUI and CLI for gene-family tree construction, multi-method comparison and optional structure-informed clustering [23]. Its structure-derived trees and single-cell cluster comparisons are phenetic summaries, not general species-tree inference. YR-MPE instead exposes a plugin-wide semantic and provenance layer across 41 phylogenetic tools and workflow clients.

4 Limitations

Three boundaries define the released scope. First, the Windows-native distribution carries a maintenance cost: each bundled binary and its runtime libraries must be audited at every version change, and tools that depend on Linux-only components or a non-portable MPI configuration stay outside the bundle. Second, the workflow specification is an editable, path-portable plan replayed within YR-MPE; it is not a CWL, Nextflow or Galaxy interchange format, and replay is user-reviewed: the engine resolves placeholders and records runs, with each step confirmed before execution. Third, tool coverage is incremental: the plugin contract supports adding programs, and MCMCtree, codeml and HyPhy are the next integrations.

5 Conclusion

YR-MPE is a modular desktop platform for phylogenetic analysis that makes the workspace, the analysis plugins and the external agent share one semantic interface. YR-MPE contributes four contracts that make GUI, CLI and agent operations interchangeable: typed results, recorded provenance, reproducible workflow placeholders, and GUI-to-CLI control bindings. The release covers a broad set of alignment, model-selection, tree-inference, species-tree, dating, selection and visualization tools on Windows. The release implements the interface, provenance and workflow contracts and illustrates them in the organelle, open-task, Bayesian and plugin-pilot evaluations reported here.

6 Methods

6.1 Implementation and deployment

YR-MPE is implemented in Python with PyQt5. The platform is organized as a plugin manager, a workspace manager, a result importer and a set of plugin windows. Each plugin inherits a common lifecycle and emits a validated `ResultSpec`; the host materializes the contained datasets without plugin-specific code. The object model and entry protocol are specified in Supplementary Note S3. Third-party programs are started through hidden subprocess helpers, so a GUI launched without a console does not create a console window for every tool. The Windows distribution bundles the analysis binaries and their runtime libraries; the source repository contains the plugin

code, the build recipes and the binary inventory. A plugin can be opened from the toolbar, from a workflow step node, or by an agent action.

Infrastructure for human–agent collaboration.

The engineering substrate directly supports the human–agent contract. The common four-tab plugin layout, file tags, result previews and typed import path give every operation a stable input/output surface that both a person and an agent can inspect. SeqMatrix and SeqDBG cover two complementary dataset entry styles, while the concatenator, sequence viewer and TreeMerger expose the corresponding state transitions; hidden subprocess helpers and a shared console report progress and errors without creating a terminal window for each tool.

The update path is equally part of the same design. Plugins are pure-source Python modules loaded by the host, so a tool window and its declared controls can be updated without recompiling the workbench. The Windows distribution ships a versioned inventory of the analysis binaries and runtime libraries, including Windows builds of DecentTree and MPBoot and MS-MPI/BEAGLE-enabled MrBayes and PhyloBayes-MPI; the GUI, command-line tools and agent bridge therefore operate on the same binaries. Because a new plugin can declare its own controls and result types without changing the host, the `control_bindings` and skill documents can be added or migrated one plugin at a time. These properties are the substrate that lets the semantic contract be enforced across the toolset.

The workspace store is a dependency-free record store rather than a database. Imports, runs and streams are explicit records, and the store can be read by a headless runner or an agent without Qt (Supplementary Note S4). Result import is the provenance boundary: when a plugin result enters the workspace, the host creates a run record with the method and tool identity, the tool version from the binary inventory, any declared parameters and command preview, the input and output item IDs, and SHA-256 hashes of the retained artifacts. A workflow result carries one run specification per step, so each tool invocation becomes a separate run in the same graph.

6.2 Semantic actions and agent bridge

Every user-facing operation is registered in a `UIActionRegistry` with an action ID, a description and a parameter schema. GUI controls and external clients call the same handler. The GUI session can be exposed through a local file bridge: a random token is generated for each session, commands, responses and events are exchanged as pure text, and a QTimer polls the command stream from the Qt main thread (Supplementary Note S5). A command is dispatched through the registry, which admits only registered operations. The bridge is local-only.

The discovery surface is part of the same registry. Menu actions are assigned stable IDs by their toolbar path; workspace state is serialized from the dataset selection manager and the workspace store; sidebar state includes the Selected basket, recommended

send-to operations and Inspector operations. Skill documents are loaded from a manifest under `YR-MPE/skills/`. A plugin's `help_doc()` combines the catalog entry with live control bindings: each row contains the GUI control, its current value, a resolved CLI value, the resulting argument list and a GUI-only note when no CLI equivalent exists. Automated tests check the bindings against each tool's `--help` output.

6.3 Workflow compilation and preview

A workflow is a sequence of steps with an optional header declaring external inputs. The compiler first removes transcript entries that do not change state: read-only discovery calls such as `workbench.menu`, `workbench.skills` and `plugin.inspect_*`, and steps whose recorded response is an error or a failure. Only executable steps enter the graph. It then performs two passes: it collects every declared output and assigns a placeholder of the form `%<step>.<role>.<extension>%`, and it rewrites matching parameter values, derives dependencies and validates the graph. Path comparisons use normalized path strings so Windows separators and case differences do not create duplicate producers. The output is a reproducible DAG with explicit input and artifact slots; a companion preview renders the DAG as data circles and tool squares. A tool node can be opened in a configuration mode in which path inputs are locked to placeholders and the Run button applies the configuration instead of launching the tool. The execution engine consumes the same plan, resolves placeholders to run-specific paths and records the resulting runs.

6.4 Evaluation methods

The workflow case study starts from annotated organelle genomes and follows the path described in Results: dataset formation with OrthoFinder, extraction of single-copy coding sequences, alignment, trimming, model selection and tree inference. We report the number of files and GUI actions or agent commands, output validity (taxon set and Newick parsing), the unrooted RF distance to the reference tree and whether the workspace reopens with the same artifact graph. The organelle case reports the handoff and provenance behaviour.

The four open tasks and their scoring rules are described in Supplementary Note S7. In brief, each task supplied raw inputs and a time limit; the choice of gene trees, alignment, reference and method was left to the participant. Each run was scored by taxon-set match, unrooted RF distance and, for the missing-data task, recall of the hidden discordant loci. Each participant ran each task once, except for the task02 skill ablation reported in Supplementary Note S7: the same Claude Code model repeated task02 with the Markdown skill documents disabled.

The MrBayes acceleration protocol is described in Supplementary Note S2. Stock and MPI+BEAGLE builds used identical inputs, trees and priors; each task used two runs with four Metropolis-coupled chains, 200,000 generations and a fixed burn-in, and the two builds were executed under the same conditions. We compared consensus trees by RF distance and report wall-clock time and likelihood behaviour.

Simulated data for the methods examples were generated with SimPhy and AliSim [60, 61]; the 150 coalescent gene trees used for the ILS concordance example in Fig. 3 were generated with msprime 1.3.4 [62]. The ITS2 case uses public sequences. The real 12-accession chloroplast fixture used by the plugin pilot is described in Supplementary Note S9. Workbench memory measurements are described in Supplementary Note S1.

The plugin pilot is a two-agent development test. One agent receives the plugin template, the control-binding specification and the skill-documentation contract, and is asked to implement a One-Step SibeliaZ dataset-formation plugin: run SibeliaZ on FASTA inputs, parse the locally collinear blocks, extract every block copy and export one typed dataset item per block. A second agent, without the first agent's transcript, receives a normal usage task and must discover the plugin through the menu/skill surface, configure it, run it and validate the typed result. We record whether the plugin loads, GUI/CLI parity, end-to-end task success, agent turns and human interventions. The task exercises block parsing and typed dataset materialisation, separating extension cost from tree inference.

6.5 Availability

YR-MPE is distributed under GPL-3.0. Source code, plugin templates, the workflow compiler, the skill manifest and the evaluation scripts are available at https://github.com/Gipsy-The-Sheller/YR-MPE. The source archive and release artifacts are archived at Zenodo (DOI: https://doi.org/10.5281/zenodo.21966720); binary releases and the user documentation are distributed through the release page at the same repository.

Supplementary information. The supplementary manual contains button-level tutorials, the complete tool inventory, the workflow JSONL examples and the benchmark protocols.

Acknowledgements. The authors thank the developers of the open-source programs integrated into YR-MPE for making their tools available.

Declarations

Funding

This work was funded by the National Natural Science Foundation of China (No. 32470883) and the Medicine-Engineering Interdisciplinary Project of Shanghai Xuhui District Stomatological Hospital (No. SHXYFYG202304).

Conflict of interest

The authors declare no conflict of interest.

Ethics approval and consent to participate

Not applicable. This study did not involve human participants, human tissue or animal experimentation.

Consent for publication

Not applicable.

Data availability

The source code, plugin templates, workflow compiler, skill manifest and evaluation scripts are available at https://github.com/Gipsy-The-Sheller/YR-MPE. The source archive and release artifacts are archived at Zenodo (DOI: https://doi.org/10.5281/zenodo.21966720). No new primary data were generated; the empirical inputs are public or are described in the supplementary information.

Code availability

See Data availability.

Author contributions

Zhijie Xu: Writing—original draft; conceptualization; software; methodology; formal analysis; visualization; validation; data curation. **Yiyang Jia**: Conceptualization; writing—review and editing; software; validation; formal analysis; visualization. **Zhihan Gu**: Validation; data curation; formal analysis. **Mingrui Shang**: Validation; formal analysis. **Xinbo Cheng**: Validation; formal analysis. **Shuo Han**: Validation; visualization. **Yang Jin**: Validation; formal analysis. **Yitong Lei**: Validation; formal analysis. **Rui Zhang**: Conceptualization; methodology; supervision; writing—original draft; writing—review and editing; funding acquisition; resources; project administration. **Yunlong Zhang**: Conceptualization; methodology; supervision; writing—original draft; writing—review and editing; funding acquisition; resources; project administration. All authors have read the final manuscript and approved it for publication.

References

[1] Kapli P, Yang Z, Telford MJ. Phylogenetic tree building in the genomic age. Nature Reviews Genetics. 2020;21(7):428–444. https://doi.org/10.1038/s41576-020-0233-0.

[2] Fernández R, Gabaldón T, Dessimoz C. Orthology: Definitions, Prediction, and Impact on Species Phylogeny Inference. In: Scornavacca C, Delsuc F, Galtier N,

editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 2.4:1–2.4:14. Available from: https://hal.inria.fr/PGE.

[3] Ranwez V, Chantret N. Strengths and Limits of Multiple Sequence Alignment and Filtering Methods. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 2.2:1–2.2:36. Available from: https://hal.inria.fr/PGE.

[4] Pupko T, Mayrose I. A Gentle Introduction to Probabilistic Evolutionary Models. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 1.1:1–1.1:21. Available from: https://hal.inria.fr/PGE.

[5] Stamatakis A, Kozlov AM. Efficient Maximum Likelihood Tree Building Methods. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 1.2:1–1.2:18. Available from: https://hal.inria.fr/PGE.

[6] Schrempf D, Szöllősi G. The Sources of Phylogenetic Conflicts. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 3.1:1–3.1:23. Available from: https://hal.inria.fr/PGE.

[7] Rannala B, Edwards SV, Leaché A, Yang Z. The Multi-species Coalescent Model and Species Tree Inference. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 3.3:1–3.3:21. Available from: https://hal.inria.fr/PGE.

[8] Bryant D, Hahn MW. The Concatenation Question. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 3.4:1–3.4:23. Available from: https://hal.inria.fr/PGE.

[9] Pett W, Heath TA. Inferring the Timescale of Phylogenetic Trees from Fossil Data. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 5.1:1–5.1:18. Available from: https://hal.inria.fr/PGE.

[10] Lartillot N. PhyloBayes: Bayesian Phylogenetics Using Site-heterogeneous Models. In: Scornavacca C, Delsuc F, Galtier N, editors. Phylogenetics in the Genomic Era. HAL; 2020. p. 1.5:1–1.5:16. Available from: https://hal.inria.fr/PGE.

[11] Zhao D, Ye T, Gao F, Jakovlić I, La Q, Tong Y, et al. PhyloSuite v2: The development of an all-in-one, efficient and visualization-oriented suite for molecular dating analysis and other advanced features. iMeta. 2025;4(6):e70095. https://doi.org/10.1002/imt2.70095.

[12] Bouckaert R, Heled J, Kühnert D, Vaughan T, Wu CH, Xie D, et al. BEAST 2: a software platform for Bayesian evolutionary analysis. PLoS Computational Biology. 2014;10(4):e1003537. https://doi.org/10.1371/journal.pcbi.1003537.

[13] Lartillot N, Philippe H. A Bayesian Mixture Model for Across-Site Heterogeneities in the Amino-Acid Replacement Process. Molecular Biology and Evolution. 2004;21(6):1095. https://doi.org/10.1093/molbev/msh112.

[14] Lartillot N, Rodrigue N, Stubbs D, Richer J. PhyloBayes MPI: Phylogenetic Reconstruction with Infinite Mixtures of Profiles in a Parallel Environment. Systematic Biology. 2013;62(4):611–615. https://doi.org/10.1093/sysbio/syt022.

[15] Zhang C, Nielsen R, Mirarab S. ASTER: A Package for Large-Scale Phylogenomic Reconstructions. Molecular Biology and Evolution. 2025;42(8):msaf172. https://doi.org/10.1093/molbev/msaf172.

[16] Zhang C, Nielsen R, Mirarab S. CASTER: Direct species tree inference from whole-genome alignments. Science. 2025;387(6737):eadk9688. https://doi.org/10.1126/science.adk9688.

[17] Liu L, Pearl DK. Species trees from gene trees: reconstructing Bayesian posterior distributions of a species phylogeny using estimated gene tree distributions. Systematic Biology. 2007;56(3):504–514. https://doi.org/10.1080/10635150701429982.

[18] Kumar S, Stecher G, Suleski M, Sanderford M, Sharma S, Tamura K. MEGA12: Molecular Evolutionary Genetic Analysis Version 12 for Adaptive and Green Computing. Molecular Biology and Evolution. 2024;41(12):msae263. https://doi.org/10.1093/molbev/msae263.

[19] Tamura K, Stecher G, Kumar S. MEGA11: Molecular Evolutionary Genetics Analysis Version 11. Molecular Biology and Evolution. 2021;38(7):3022–3027. https://doi.org/10.1093/molbev/msab120.

[20] Zhang D, Gao F, Jakovlić I, Zou H, Zhang J, Li WX, et al. PhyloSuite: An integrated and scalable desktop platform for streamlined molecular sequence data management and evolutionary phylogenetics studies. Molecular Ecology Resources. 2020;20(1):348–355. https://doi.org/10.1111/1755-0998.13096.

[21] Edler D, Klein J, Antonelli A, Silvestro D. raxmlGUI 2.0: A graphical interface and toolkit for phylogenetic analyses using RAxML. Methods in Ecology and Evolution. 2021;12(2):373–377. https://doi.org/10.1111/2041-210X.13512.

[22] Stamatakis A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics. 2014;30(9):1312–1313. https://doi.org/10.1093/bioinformatics/btu033.

[23] Yu D, Zhang W, Li H, Li L. eGPS-oneBuilder: an integrated GUI-CLI workflow for sequence-based phylogenetic tree construction, structure-informed clustering, and interactive tree interpretation. Genomics Communications. 2026;3(1):0015. https://doi.org/10.48130/gcomm-0026-0015.

[24] The Galaxy C. The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. Nucleic Acids Research. 2024;52(W1):W83–W94. https://doi.org/10.1093/nar/gkae410.

[25] Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nature Biotechnology. 2017;35(4):316–319. https://doi.org/10.1038/nbt.3820.

[26] Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, et al. Sustainable data analysis with Snakemake. F1000Research. 2021;10:33. https://doi.org/10.12688/f1000research.29032.2.

[27] Boiko DA, MacKnight R, Kline B, Gomes G. Autonomous chemical research with large language models. Nature. 2023;624(7992):570–578. https://doi.org/10.1038/s41586-023-06792-0.

[28] Gao S, Zhu R, Sui P, Kong Z, Aldogom S, Huang Y, et al.: Democratizing AI scientists using ToolUniverse. ArXiv:2509.23426 (2025).

[29] Gao S, Fang A, Huang Y, Giunchiglia V, Noori A, Schwarz JR, et al. Empowering biomedical discovery with AI agents. Cell. 2024;187(22):6125–6151. https://doi.org/10.1016/j.cell.2024.09.022.

[30] Katoh K, Standley DM. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. Molecular Biology and Evolution. 2013;30(4):772–780. https://doi.org/10.1093/molbev/mst010.

[31] Edgar RC. Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. Nature Communications. 2022;13(1):1–9. https://doi.org/10.1038/s41467-022-34630-w.

[32] Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular Systems Biology. 2011;7:539. https://doi.org/10.1038/msb.2011.75.

[33] Ranwez V, Douzery EJP, Cambon C, Chantret N, Delsuc F. MACSE v2: Toolkit for the Alignment of Coding Sequences Accounting for Frameshifts and Stop Codons. Molecular Biology and Evolution. 2018;35(10):2582–2584. https://doi.org/10.1093/molbev/msy159.

[34] Capella-Gutiérrez S, Silla-Martínez JM, Gabaldón T. trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses. Bioinformatics. 2009;25(15):1972–1973. https://doi.org/10.1093/bioinformatics/btp348.

[35] Talavera G, Castresana J. Improvement of Phylogenies after Removing Divergent and Ambiguously Aligned Blocks from Protein Sequence Alignments. Systematic

Biology. 2007;56(4):564–577. https://doi.org/10.1080/10635150701472164.

[36] Kalyaanamoorthy S, Minh BQ, Wong TKF, von Haeseler A, Jermiin LS. ModelFinder: fast model selection for accurate phylogenetic estimates. Nature Methods. 2017;14(6):587–589. https://doi.org/10.1038/nmeth.4285.

[37] Darriba D, Posada D, Kozlov AM, Stamatakis A, Morel B, Flouri T. ModelTest-NG: a new and scalable tool for the selection of DNA and protein evolutionary models. Molecular Biology and Evolution. 2020;37(1):291–294. https://doi.org/10.1093/molbev/msz189.

[38] Wong TKF, Ly-Trong N, Ren H, Baños H, Roger AJ, Susko E, et al. IQ-TREE 3: Phylogenomic Inference Software using Complex Evolutionary Models; 2025. Available from: https://ecoevorxiv.org/repository/view/8916/.

[39] Kozlov AM, Darriba D, Flouri T, Morel B, Stamatakis A. RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. Bioinformatics. 2019;35(21):4453–4455. https://doi.org/10.1093/bioinformatics/btz305.

[40] Ronquist F, Huelsenbeck JP. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics. 2003;19(12):1572–1574. https://doi.org/10.1093/bioinformatics/btg180.

[41] Hoang DT, Vinh LS, Flouri T, Stamatakis A, von Haeseler A, Minh BQ. MPBoot: fast phylogenetic maximum parsimony tree inference and bootstrap approximation. BMC Evolutionary Biology. 2018;18(1):11. https://doi.org/10.1186/s12862-018-1131-3.

[42] Wang W, Barbetti J, Wong T, Thornlow B, Corbett-Detig R, Turakhia Y, et al. DecentTree: scalable Neighbour-Joining for the genomic era. Bioinformatics. 2023;39(9):btad536. https://doi.org/10.1093/bioinformatics/btad536.

[43] Zhang C, Mirarab S. Weighting by Gene Tree Uncertainty Improves Accuracy of Quartet-based Species Trees. Molecular Biology and Evolution. 2022;39(12):msac215. https://doi.org/10.1093/molbev/msac215.

[44] To TH, Jung M, Lycett S, Gascuel O. Fast Dating Using Least-Squares Criteria and Algorithms. Systematic Biology. 2016;65(1):82–97. https://doi.org/10.1093/sysbio/syv068.

[45] Penel S, Menet H, Tricou H, Daubin V, Tannier E. Treerecs: an integrated phylogenetic tool, from sequences to reconciliations. Bioinformatics. 2020;36(18):4822–4824. https://doi.org/10.1093/bioinformatics/btaa615.

[46] Vaughan TG. IcyTree: rapid browser-based visualization for phylogenetic trees and networks. Bioinformatics. 2017;33(15):2392–2394. https://doi.org/10.1093/

bioinformatics/btx155.

[47] Rambaut A, Drummond AJ, Xie D, Baele G, Suchard MA. Posterior Summarization in Bayesian Phylogenetics Using Tracer 1.7. Systematic Biology. 2018;67(5):901–904. https://doi.org/10.1093/sysbio/syy032.

[48] Ayres DL, Cummings MP, Baele G, Darling AE, Lewis PO, Swofford DL, et al. BEAGLE 3: Improved Performance, Scaling, and Usability for a High-Performance Computing Library for Statistical Phylogenetics. Systematic Biology. 2019;68(6):1052–1061. https://doi.org/10.1093/sysbio/syz020.

[49] Minh BQ, Hahn MW, Lanfear R. New Methods to Calculate Concordance Factors for Phylogenomic Datasets. Molecular Biology and Evolution. 2020;37(9):2727–2733. https://doi.org/10.1093/molbev/msaa106.

[50] Kück P, Struck TH. BaCoCa – A heuristic software tool for the parallel assessment of sequence biases in hundreds of gene and taxon partitions. Molecular Phylogenetics and Evolution. 2014;70:94–98. https://doi.org/10.1016/j.ympev.2013.09.011.

[51] Nei M, Gojobori T. Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. Molecular Biology and Evolution. 1986;3(5):418–426. https://doi.org/10.1093/oxfordjournals.molbev.a040410.

[52] Yang Z, Nielsen R. Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. Molecular Biology and Evolution. 2000;17(1):32–43. https://doi.org/10.1093/oxfordjournals.molbev.a026236.

[53] Struck TH, Nesnidal MP, Purschke G, Halanych KM. Detecting possibly saturated positions in 18S and 28S sequences and their influence on phylogenetic reconstruction of Annelida (Lophotrochozoa). Molecular Phylogenetics and Evolution. 2008;48(2):628–645. https://doi.org/10.1016/j.ympev.2008.05.015.

[54] Mo YK, Lanfear R, Hahn MW, Minh BQ. Updated Site Concordance Factors Minimize Effects of Homoplasy and Taxon Sampling. Bioinformatics. 2023;39(1):btac741. https://doi.org/10.1093/bioinformatics/btac741.

[55] Lanfear R, Hahn MW. The Meaning and Measure of Concordance Factors in Phylogenomics. Molecular Biology and Evolution. 2024;41(11):msae214. https://doi.org/10.1093/molbev/msae214.

[56] Allman ES, Mitchell JD, Rhodes JA. Gene Tree Discord, Simplex Plots, and Statistical Tests under the Coalescent. Systematic Biology. 2022;71(4):929–942. https://doi.org/10.1093/sysbio/syab008.

[57] Rhodes JA, Baños H, Mitchell JD, Allman ES. MSCquartets 1.0: Quartet Methods for Species Trees and Networks under the Multispecies Coalescent Model in R.

Bioinformatics. 2021;37(12):1766–1768. https://doi.org/10.1093/bioinformatics/btaa868.

[58] Gao Z, et al.: SciForge: An AI-Native, Multimodal Workbench for Scientific Discovery. ArXiv:2607.16038 (2026).

[59] Ouyang W, et al.: AIPOCH Open-Science: An Open-Source AI Research Workbench with Reproducible .science Packages. ArXiv:2610.17110 (2026).

[60] Mallo D, de Oliveira Martins L, Posada D. SimPhy: Phylogenomic Simulation of Gene, Locus, and Species Trees. Systematic Biology. 2016;65(2):334–344. https://doi.org/10.1093/sysbio/syv082.

[61] Ly-Trong N, Naser-Khdour S, Lanfear R, Minh BQ. AliSim: A Fast and Versatile Phylogenetic Sequence Simulator for the Genomic Era. Molecular Biology and Evolution. 2022;39(5):msac092. https://doi.org/10.1093/molbev/msac092.

[62] Baumdicker F, Bisschop G, Goldstein D, Gower G, Ragsdale AP, Tsambos G, et al. Efficient ancestry and mutation simulation with msprime 1.0. Genetics. 2022;220(3):iyab229. https://doi.org/10.1093/genetics/iyab229.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- YRMPE1.2.0supplementaryinformation.pdf
- YRMPE1.2.0toolreference.pdf
- YRMPE1.2.0usermanual.pdf
- YRMPE1.2.0casestudies.pdf