BABAPPA: A Bash-Based Automated Pipeline for Codeml-Mediated Selection Analysis Integrating PRANK and IQ-TREE2

Krishnendu Sinha[1*]

[1]Department of Zoology, Jhargram Raj College, Jhargram-721507

***Corresponding author**

Dr. Krishnendu Sinha
Assistant Professor in Zoology
Jhargram Raj College
Jhargram-721507
dr.krishnendusinha@gmail.com

Abstract:

Detecting signatures of positive selection in protein-coding genes is fundamental to understanding adaptive evolution. However, conventional workflows using codeml (PAML) require extensive manual configuration (e.g. editing control files) and are limited by sequential execution, which is impractical for large-scale analyses. To address these challenges, BABAPPA (Bash-Based Automated Parallel Positive Selection Analysis), a fully automated pipeline has been developed that streamlines every step of codeml analysis. BABAPPA integrates codon-aware sequence filtering, alignment, phylogenetic reconstruction, branch designation, model fitting, likelihood-ratio testing, and multiple-testing correction into a single reproducible workflow. The BABAPPA has been benchmarked on three Brassicaceae orthologous gene sets and found that it produces identical positive-selection results (after FDR correction) to a standard codeml approach, while dramatically reducing computation time. Mean wall-clock time dropped from 2,579 s (standard workflow) to 1,430 s with BABAPPA, a 44.6% reduction in runtime which is equivalent to an 80.3% speedup (p = 8.227e-5). These gains are achieved without loss of sensitivity. The robust automation and efficiency of BABAPPA make it well-suited for large-scale genomic surveys of adaptive evolution.

Keywords: Positive Selection, Molecular Evolution, PAML(Codeml), PRANK, IQ-TREE2, Phylogenetics

1. Introduction

The detection of positive (adaptive) selection in protein-coding genes is a cornerstone of molecular evolution studies, as it reveals adaptive changes across lineages [1,2]. Maximum-likelihood models implemented in PAML's codeml (including site, branch, and branch-site models) remain a gold standard for such detection [3]. However, traditional codeml workflows are labor-intensive: researchers must manually edit configuration files, run multiple model comparisons sequentially, and collate likelihood-ratio tests (LRTs) across many genes and branches, applying corrections for multiple testing [4]. These manual steps are prone to error and severely limit throughput when analyzing large gene families or whole genomes. To overcome these bottlenecks, we developed BABAPPA, an automated pipeline that orchestrates the entire codeml analysis from raw sequences to statistically validated results. BABAPPA automates sequence quality control, codon-aware alignment, phylogenetic inference, branch annotation, model fitting, and LRT-based selection testing, all with parallel execution to maximize efficiency. It leverages standard bioinformatics tools, such as Biopython[5] for sequence handling, PRANK [6] for codon alignment, IQ-TREE2 [7] for phylogenetic inference, and GNU Parallel[8] for parallel processing alongside custom scripting. The result is a reproducible workflow that requires minimal manual intervention and delivers substantial speedups without compromising sensitivity (Figure 1). Here, the design of BABAPPA is described, and its performance is demonstrated against a standard codeml-based pipeline.

2. Materials and Methods

2.1. Validation datasets

To evaluate BABAPPA, three orthologous gene groups from Brassicaceae species were selected. These gene sets were obtained from the PlantTFDB[9], with each set comprising five putative orthologs from three closely related species. Transcription factor families representative of typical codeml use cases were included among the selected genes. Gene identifiers data for each orthogroup are provided in Supplementary File S1.

2.2. *Pipeline implementation*

BABAPPA is implemented as a suite of modular Bash and Python scripts coordinated by a master driver (`babappa.sh`). The pipeline runs end-to-end, from raw coding sequences to final selection results (Figure 1). It integrates the following key steps:

2.2.1 Sequence Quality Control

`seqQC.py` has been developed to apply stringent codon-level filters as per the standard protocol [2,10]. Input sequences must begin with an ATG start codon, end with a valid stop codon (TAA, TAG, or TGA), be longer than 300 base pairs, contain only the nucleotides A, T, G, and C, and have no in-frame stop codons internally. Sequences whose lengths are extreme outliers (outside 1.5× interquartile range) are also discarded. Only high-quality sequences pass to the next stage.

2.2.2 Phylogeny-Aware Alignment

Remaining sequences for each gene are aligned in codon mode using PRANK[6] (via `script0.sh`). PRANK is chosen for its phylogeny-aware approach to insertion/deletion placement, improving alignment accuracy for selection analysis.

2.2.3 Phylogenetic Reconstruction

For each codon alignment, BABAPPA infers a maximum-likelihood phylogeny using IQ-TREE2 [7] under an appropriate codon substitution model. By default, we perform 1,000 ultrafast bootstrap replicates [11] and approximate likelihood-ratio tests (aLRT) to assess branch support. This step is executed by `script1.sh`, which collects the resulting tree files.

2.2.4 Foreground Branch Designation

A custom Python script (`4GroundBranchGenerator.py`) processes each inferred tree. It iteratively marks each branch in turn as the “foreground” branch (by appending a #1 tag) for subsequent branch-site tests [2]. The output is a set of Newick tree files, each with a different branch annotated as foreground.

2.2.5 Codeml Analyses

The codeml models were then fitted to each gene and branch scenario. Control files (`codeml.ctl`) for site, branch, and branch-site models are generated dynamically from a template using embedded `sed` commands, eliminating manual edits. For each gene, `script2.sh` runs branch and branch-site models (alternative and null) in parallel across branches using GNU Parallel[8], while `script3.sh` and `script4.sh` run site models (including M0, M1a, M2a, M3, M7, M8) in parallel. A Python wrapper (`run_codeml.py`) ensures that the correct model versions (alternative vs. null) are run for branch-site comparisons. Throughout, all executions leverage multi-core processing to maximize speed.

2.2.6 Result Aggregation and Statistical Testing

After model fitting, custom scripts (`script5.sh`, `script6.sh`) parse codeml output files to extract log-likelihood (*lnL*) values and parameter counts for each model, writing per-gene summaries in CSV format. A final Python script (`lrt_bh_correction.py`) computes likelihood-ratio statistics ($2\Delta lnL$) between paired models, calculates *p*-values from χ^2 distributions, and applies the Benjamini–Hochberg false-discovery rate (FDR) correction[12] across tests. Raw and adjusted *p*-values, along with significance flags (FDR < 0.05), are output to summary Excel reports in accordance with `script7.sh` and `script8.sh`. All pipeline steps and custom scripts are documented and configurable; source code is available on GitHub.

2.3. Benchmarking and statistical analysis

BABAPPA was compared to a conventional codeml workflow in which sequence alignments, phylogenies, and codeml runs were performed sequentially with without any parallelization. Both pipelines were executed on a consumer laptop (Intel Core i7 13th Gen CPU, 16 GB RAM, 512 GB SSD) running Ubuntu Linux via Windows Subsystem for Linux. For each of the three orthogroups, both BABAPPA and the standard workflow were run three times to account for run-to-run variability. Wall-clock times were recorded for each complete analysis (in seconds). The distributions of runtimes were compared using a two-sided Wilcoxon rank-sum test (Mann–Whitney U, $\alpha = 0.05$) to assess the statistical significance of speed differences. All tests and data analyses were conducted in R.

3. Results

3.1. Concordance of positive-selection calls

BABAPPA's automation did not compromise the ability to detect positive selection. Across all three Brassicaceae orthogroups, BABAPPA identified exactly the same set of genes under significant branch-site selection (after FDR correction, threshold < 0.05) as the standard manual workflow. In other words, every gene flagged for positive selection by BABAPPA was also flagged by the conventional approach. This perfect concordance (see Supplementary Files S2–S7 for per-gene results) confirms that the automated pipeline preserves the sensitivity and specificity of codeml's likelihood-ratio tests despite the optimization of workflow execution.

3.2. Computational efficiency

In contrast, BABAPPA dramatically reduced computation time. The mean wall-clock time across the three orthogroups was 1,430.2 seconds for BABAPPA, versus 2,578.8 seconds for the standard sequential workflow. This represents a 44.6% lean reduction in runtime (i.e., (excluding variable manual-setup times). Equivalently, BABAPPA achieved an 80.3% increase in processing speed compared to the baseline. The Wilcoxon test on the recorded runtimes yielded $p = 8.227e-5$, indicating that the efficiency gains are highly significant. Figure 2 illustrates the faster runtimes achieved by BABAPPA. These speedups are primarily due to parallel execution of independent tasks (especially codeml model fits) and streamlined file handling.

4. Discussion

BABAPPA represents a unified solution to the technical bottlenecks of codeml-based selection analysis. By automating sequence quality control, codon alignment, tree inference, model execution, and statistical summarization, the pipeline greatly reduces manual effort and the potential for human error. The nearly 80% faster processing on standard hardware highlights the benefits of parallelized execution: tasks such as independent model fits and bootstrapped tree building are distributed across CPU cores, greatly shortening wall-clock time. The built-in aggregation of results and statistical tests (with FDR correction) further streamlines the end-to-end workflow, saving researchers from tedious data wrangling. Crucially, the automation does not trade off accuracy for speed. Our validation confirms that

BABAPPA produces identical positive-selection calls to the conventional approach (FDR<0.05), meaning that users can trust the pipeline's results for publication-quality studies. Because all steps, from filtering through output tables, are scripted, analyses are fully reproducible. Moreover, the modular design allows easy customization; for example, users may adjust filtering thresholds, alignment parameters, or substitute alternative phylogenetic models as needed. BABAPPA is freely available as open-source software. The pipeline code and documentation are hosted on GitHub at https://github.com/sinhakrishnendu/babappa, along with sample data and detailed setup instructions. A graphical user interface is currently under development to further lower the barrier to use for researchers less comfortable with the command line. In summary, BABAPPA fills a critical gap by providing an efficient, reproducible, and user-friendly platform for positive-selection analysis. As genomic data sets continue to expand, automated pipelines like BABAPPA will be essential for enabling large-scale evolutionary inferences even in "dry-lab" settings without extensive bioinformatics resources. It is anticipate that BABAPPA will be a valuable tool for the evolutionary biology community, facilitating rigorous selection scans across genes and genomes.

5. Conclusion

The detection of positive selection is a key step in understanding molecular adaptation. Traditional codeml workflows pose technical challenges that limit scalability. Here the BABAPPA has been introduced, an end-to-end automated pipeline that integrates codon-aware filtering, alignment, phylogenetic inference, codeml model testing, and FDR-corrected LRTs. In validation tests on Brassicaceae gene families, BABAPPA matched the detection performance of the standard pipeline while reducing computation time by almost half (an 80% speedup). These results demonstrate that BABAPPA delivers both accuracy and efficiency. Its modular, open-source design makes it readily adaptable, and its automation makes large-scale selection analyses practical. By simplifying and accelerating codeml-based studies, BABAPPA enables broader application of positive-selection analyses in evolutionary research.

6. Funding details

This research received no funding.

7. Conflicts of interest

The author report that there are no competing interests to declare.

8. Data availability statement

The data are provided within the manuscript or supplementary information files. Also, the GitHub repository at https://github.com/sinhakrishnendu/babappa.git contains all the supplementary materials in the form of raw and processed data, as well as all necessary scripts to replicate the analyses presented in this manuscript.

9. Acknowledgments

I am deeply grateful to my little son, whose boundless love for butterflies inspired the name "BABAPPA." In his delightful vocabulary, "BABAPPA" is synonymous with butterfly, and it was his creativity that gave the pipeline its name. I also thank Avijit Roy (Department of Zoology, Jhargram Raj College) for assistance in collecting the ortholog sequence data. I also acknowledge the developers of the open-source tools and databases used by BABAPPA, whose work made this pipeline possible.

10. Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the I used ChatGPT in order to refine this manuscript's and code's comments and readability. After using this tool/service, the I reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

11. References

1. Sinha, K., Jana, S., Pramanik, P. & Bera, B. Selection on synonymous codon usage in soybean (Glycine max) WRKY genes. *Scientific Reports 2024 14:1* **14**, 1–18 (2024).
2. Sinha, K. Episodic Positive Selection Signatures in Arabidopsis CONSTANS-like Genes COL3 and COL5 Indicating Adaptive Evolution in Red-Light Signaling Pathways. *bioRxiv* 2025.04.03.646976 (2025) doi:10.1101/2025.04.03.646976.
3. Yang, Z. PAML 4: phylogenetic analysis by maximum likelihood. *Mol Biol Evol* **24**, 1586–1591 (2007).
4. Álvarez-Carretero, S., Kapli, P. & Yang, Z. Beginner's Guide on the Use of PAML to Detect Positive Selection. *Mol Biol Evol* **40**, (2023).

5. Cock, P. J. A. *et al.* Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**, 1422–1423 (2009).

6. Löytynoja, A. Phylogeny-aware alignment with PRANK. *Methods Mol Biol* **1079**, 155–170 (2014).

7. Minh, B. Q. *et al.* IQ-TREE 2: New Models and Efficient Methods for Phylogenetic Inference in the Genomic Era. *Mol Biol Evol* **37**, 1530–1534 (2020).

8. Tange, O. GNU Parallel 2018. doi:10.5281/ZENODO.1146014.

9. Tian, F., Yang, D. C., Meng, Y. Q., Jin, J. & Gao, G. PlantRegMap: charting functional regulatory maps in plants. *Nucleic Acids Res* **48**, D1104–D1113 (2020).

10. Sinha, K., Jana, S., Pramanik, P. & Bera, B. Selection on synonymous codon usage in soybean (Glycine max) WRKY genes. *Scientific Reports 2024 14:1* **14**, 1–18 (2024).

11. Hoang, D. T., Chernomor, O., Von Haeseler, A., Minh, B. Q. & Vinh, L. S. UFBoot2: Improving the Ultrafast Bootstrap Approximation. *Mol Biol Evol* **35**, 518–522 (2018).

12. Benjamini, Y. & Hochberg, Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)* **57**, 289–300 (1995).

12. Tables

12.1. Table1

Details of scripts and their function with input and output format.

Script / File	Function	Input	Output
`seqQC.py`	Performs biological QC on coding sequences (start/stop codons, length, outliers).	• Raw FASTA (`.fasta`) or text (`.txt`) file of coding sequences	• QC-passed FASTA file • QC log file (`.log.txt`)
`script0.sh`	Orchestrates QC (via `seqQC.py`) and codon□aware MSA with PRANK in parallel.	• Raw sequence files (`*.txt`, `*.fasta`)	• `QCseq/` folder of QC-filtered FASTA • `msa/` folder of PRANK alignments (`*_msa*`)
`script1.sh`	Runs IQ-TREE 2 codon phylogenetic inference (bootstrap & ALRT), collects and labels trees.	• `msa/*.fas` MSA files	• `iqtreeoutput/<gene>/` IQ-TREE logs & treefiles • `treefiles/` consolidated treefiles • `foregroundbranch/` per-gene Newicks
`4GroundBranchGenerator.py`	Generates one Newick with a single foreground branch tag per gene for branch-site tests.	• One Newick treefile (`.treefile`) • Output directory	• Per-gene `.treefile` with `#1` marking as foreground
`run_codeml.py`	Writes codeml control files for site/branch/branch-site, and runs codeml accordingly.	• A folder containing `aligned.fasta` & a Newick treefile • Destination output folder	• Subfolders (`site/`, `branch/`, `branch-site/`, `branch-site-null/`) each with `results.txt` & `codeml.ctl`
`script2.sh`	Parallelizes branch & branch-site codeml runs across species & treefiles.	• `msa/` (aligned FASTA) • `foregroundbranch/` treefiles • Template `codeml.ctl`	• `codemloutput/<species>/<tree>_{B,BS,BS_NULL}/` folders with codeml outputs
`script3.sh`	Parallelizes M0 (site□null) codeml runs across species.	• `msa/` (aligned FASTA) • `treefiles/` • Template `codeml.ctl`	• `codemloutput/<species>/M0/results` folder with codeml output
`script4.sh`	Automates codeml site□model (Models 0,1,2,3,7,8) analyses in parallel.	• `msa/` (aligned FASTA) • `treefiles/` • Template `codeml.ctl`	• `sitemodel/<species>/` folders containing `output.txt` and model-specific logs
`script5.sh`	Extracts lnL & np from branch & branch-site codeml outputs into CSV.	• `codemloutput/<species>/../output.txt` files	• `codemlanalysis/<species>/lnL_np_values.csv`

script6.sh	Extracts lnL & np from site-model codeml outputs into CSV.	• sitemodel/<species>/output.txt	• sitemodelanalysis/<species>/lnL_np_values.csv
lrt_bh_correction.py	Computes LRT statistics and applies Benjamini–Hochberg correction (branch models).	• CSVs: codemlanalysis/<species>/*.csv	• Excel reports LRT_results_<…>.xlsx (raw p, FDR p, significance)
lrt_bh_correction.sitemodel.py	Computes LRT & BH correction for site-model comparisons.	• CSV: site-model results (*.csv with Model, lnL, np)	• lrt_results.csv summarizing LRT, df, p, BH-adjusted p
script7.sh	Wraps lrt_bh_correction.py to batch process species and move Excel outputs.	• codemlanalysis/<species>/*.csv • lrt_bh_correction.py	• BHanalysis/<species>/LRT_results_<…>.xlsx
script8.sh	Automates BH correction for site models across species (sitemodel workflow).	• sitemodelanalysis/<species>/*.csv • corresponding Python/Bash BH script	• SiteModelBH/<species>/ Excel or CSV outputs with FDR-adjusted results

12.2. Table 2

Complete Installation and Setup Instructions for dependencies to run BABAPPA. This table provides a detailed, step-by-step guide for installing the Windows Subsystem for Linux (WSL), setting up an Ubuntu distribution, installing all necessary dependencies, and running the BABAPPA pipeline. Each command is specifically labeled to indicate whether it should be executed in Windows Powershell (for system setup) or in the WSL Ubuntu Bash Terminal (for Linux environment preparation and pipeline execution).

Step	Purpose	Command (Windows Powershell as Admin)	Notes
Install WSL	Enable WSL and Virtual Machine Platform features	wsl --install	(Windows will automatically install WSL2 and Ubuntu by default); Windows Powershell (Admin)
(Optional)	List available distros	wsl --list --online	To see other available distributions; Windows Powershell (Admin)
(Optional)	Install Ubuntu manually if needed	wsl --install -d Ubuntu	Replace Ubuntu with Ubuntu-22.04 etc. if you want a specific version; Windows Powershell (Admin)
Check WSL Installation	Confirm WSL installed correctly	wsl --status	Should show Default Version: 2; Windows Powershell (Admin)
Launch WSL (Ubuntu)	Open installed Ubuntu distro	Search **Ubuntu** from Start menu and open it	First time setup will ask for username and password; WSL (Ubuntu Bash)
Update Ubuntu packages	Update package list inside Ubuntu	sudo apt update && sudo apt upgrade	Always recommended immediately after first install; WSL (Ubuntu Bash)
Install Core Tools	PRANK, IQ-TREE, PAML, GNU Parallel	sudo apt install prank iqtree paml parallel	Main external software needed; WSL (Ubuntu Bash)
Install Python and Pip	Set up Python3 and Pip3	sudo apt install python3 python3-pip	Required for BABAPPA Python scripts; WSL (Ubuntu Bash)
Install Python Packages	Biopython, SciPy, StatsModels, Pandas, openpyxl	pip3 install biopython scipy statsmodels pandas openpyxl	Statistical testing and reporting packages; WSL (Ubuntu Bash); WSL (Ubuntu Bash)
Clone or Copy BABAPPA	Place BABAPPA scripts inside WSL filesystem	Copy manually or use git clone	git clone <repo>
Run BABAPPA Pipeline	Execute BABAPPA	./babappa.sh	Main command to run everything; WSL (Ubuntu Bash)

13. Figure legends

13.1. Figure 1

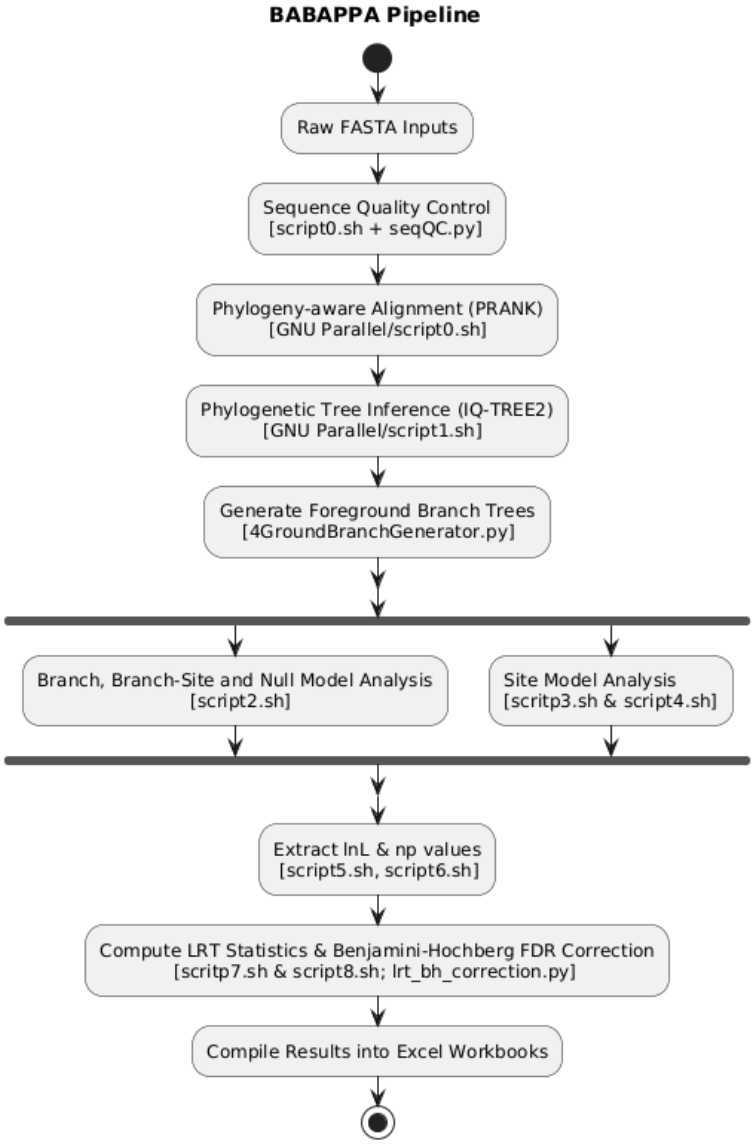
Schematic workflow of BABAPPA and comparative runtime performance against the standard codeml workflow.

13.2. Figure 2

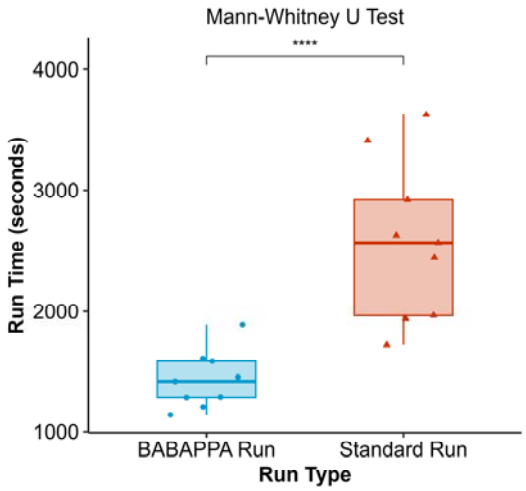
Benchmarking of BABAPPA run with standard run.

14. Figures

14.1. Figure 1



14.2. Figure 2


Mann-Whitney U Test
****
4000
3000
2000
1000
Run Time (seconds)
BABAPPA Run
Standard Run
Run Type