BABAPPA: An 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:

Detection of episodic positive selection is fundamental to understanding adaptive evolution, yet conventional codeml-based workflows require extensive manual configuration and suffer from limited parallelization. To overcome these limitations, we developed BABAPPA (BAsh-Based Automated Parallel Positive selection Analysis), an end-to-end, fully automated pipeline that integrates sequence quality control, codon-aware alignment, phylogenetic inference, foreground branch designation, codeml analyses (site, branch and branch-site), likelihood ratio testing with Benjamini-Hochberg correction, and result summarization. Validation on three Brassicaceae orthogroups showed that BABAPPA yields identical positive-selection calls to a standard codeml workflow while reducing mean wall-clock time from 2,578.78 s to 1,430.22 s (a 44.6 % reduction, corresponding to an 80.31 % increase in processing speed; p = 8.227e-5). The robust automation of BABAPPA, combined with its significant efficiency improvements, makes it exceptionally well-suited for large-scale genomic surveys and scientific publications.

Keywords: Positive Selection, Molecular Evolution, PAML/Codeml, PRANK, IQ-TREE2, Phylogenetics, Automation Pipeline, Graphical User Interface (GUI)

1. Introduction

Positive selection leaves characteristic signatures in protein-coding genes and drives adaptive change across diverse lineages[1,2]. Maximum-likelihood models implemented in PAML's codeml-particularly site, branch, and branch-site models-remain the gold standard for detecting these signatures[3]. However, traditional codeml workflows often involve manual editing of control files, sequential execution of multiple hypotheses across numerous branches, and laborious aggregation of likelihood ratio tests (LRTs) with multiple-testing corrections[4]. These steps not only introduce human error but also severely constrain throughput when analyzing large gene families or genome-wide datasets. To address these challenges, BABAPPA (BAsh-Based Automated Parallel Positive selection Analysis) an integrated pipeline has been developed that automates every stage of the positive-selection analysis while optimizing parallel execution. BABAPPA leverages Biopython[5] for sequence parsing, PRANK[6] for phylogeny-aware codon alignment, IQ-TREE 2[7] for phylogenetic reconstruction, GNU Parallel[8] for efficient resource utilization, and custom Python scripts for dynamic control-file generation and statistical testing. The result is a reproducible workflow (Figure 1) that requires minimal manual intervention and delivers substantial speedups without compromising sensitivity.

2. Materials and Methods

2.1. Validation datasets

Three orthologous gene groups from Brassicaceae species were selected to benchmark BABAPPA and obtained from PlantTFDB[9]. Each group comprises five putative orthologs randomly drawn from three closely related species. Detailed gene identifiers for each orthogroup are provided in Supplementary File S1.

2.2. Pipeline implementation

BABAPPA consists of a series of modular scripts controlled by a master Bash driver, `babappa.sh`, supported by the dependencies (Table 1 and 2). First, `seqQC.py` applies stringent codon-based quality filters: sequences must start with ATG, end with a valid stop codon, exceed 300 bp, contain only A/T/G/C nucleotides, lack internal in-frame stops, and pass interquartile range length outlier rejection[2,10]. Surviving sequences are aligned with PRANK[6] in codon mode within `script0.sh`. Next, `script1.sh` invokes IQ-TREE 2[7]

under the codon model with 1,000 ultrafast bootstrap[11] and aLRT replicates (defaults which can be customized) to infer maximum-likelihood phylogenies. Generated treefiles are collected and processed by `4GroundBranchGenerator.py`, which annotates each branch in turn as foreground for subsequent branch-site tests[2]. Codeml[3,4] analyses run in parallel via `script2.sh` (branch and branch-site models) and `script2.sh` and `script3.sh` (M0 and other site models, M1a, M2a, M3, M7 and M8)[4]. Control files are dynamically generated from a template (`codeml.ctl`) by embedded sed commands in each script, ensuring consistency and eliminating manual editing. A Python wrapper (`run_codeml.py`) orchestrates alternative and null hypothesis runs for branch-site tests. Result aggregation employs `script5.sh` and `script6.sh` to extract lnL and parameter counts from codeml output, writing per-gene CSV summaries. Finally, `lrt_bh_correction.py` computes LRT statistics and applies Benjamini–Hochberg correction using SciPy and StatsModels, outputting Excel reports with raw and adjusted p-values alongside significance flags.

2.3. Benchmarking and statistical analysis

Benchmarking was performed on a laptop equipped with an Intel Core i7 13th Gen 13620H processor, 16 GB RAM, 512 GB SSD, Windows 11 Home, and an NVIDIA GeForce RTX 4050 GPU (6 GB VRAM). For each of the three validation orthogroups, both BABAPPA and a standard codeml workflow (sequential execution, manual control-file editing, no optimized parallelism) were run three times. Clock-times were recorded in seconds. Distributions of runtimes were compared using a two-sided Wilcoxon rank-sum test (Mann–Whitney U) with significance threshold $\alpha = 0.05$. All statistical analyses were conducted in R.

3. Results

3.1. Concordance of positive-selection calls

Across all three orthogroups, BABAPPA and the standard workflow identified identical sets of genes with significant branch-site selection signals after FDR correction (FDR < 0.05), confirming that automation in BABAPPA does not compromise detection sensitivity (Supplementary File S2–S7).

3.2. Computational efficiency

BABAPPA's mean runtime was 1,430.22 s, compared to 2,578.78 s for the standard workflow, corresponding to a 44.6 % reduction in wall-clock time and an 80.31 % speedup in processing. The Wilcoxon test yielded $p = 8.227 \times 10^{-}$, demonstrating highly significant efficiency gains (Figure 2).

4. Discussion

BABAPPA streamlines positive-selection analysis by automating sequence quality control, alignment, phylogenetic inference, codeml hypothesis testing, and statistical validation in a single, reproducible pipeline. The nearly 80 % speedup observed on consumer-grade hardware underscores the benefits of optimized parallelization and consolidated file-handling. By eliminating manual control-file edits and providing built-in result aggregation, BABAPPA reduces both effort and potential for user error, making large-scale and genome-wide selection surveys practical for dry-lab researchers. A graphical user interface is under development to further lower the barrier for non-computational users. BABAPPA is freely available at https://github.com/sinhakrishnendu/babappa.git under an open-source license.

5. Conclusion

The detection of positive selection across genes is a critical step in understanding molecular adaptation and evolutionary processes. However, existing workflows often impose technical barriers due to their reliance on manual steps and lack of integration. BABAPPA offers a fully automated, reproducible, and computationally efficient solution to this challenge. By combining stringent quality control, codon-aware alignment, robust phylogenetic inference, streamlined codeml analyses, and automated statistical validation within a unified framework, BABAPPA greatly simplifies positive selection studies even for large and complex datasets. Validation across multiple orthogroups demonstrated that BABAPPA achieves the same detection sensitivity as conventional workflows, while substantially reducing computational time-achieving an 80.31% increase in processing speed. These results highlight BABAPPA's potential to enable broader applications of selection analysis in dry-lab research environments[1,2]. Given its modular design and open-source availability, BABAPPA stands as a valuable tool for evolutionary biologists seeking efficient, large-scale, and reproducible analyses of molecular adaptation.

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, PG 4th Semester (2023-2025 academic session) student at Jhargram Raj College, Department of Zoology, for collecting the ortholog data. I thank the global open-access community for their dedication to free science and for breaking down barriers to knowledge without which this study was not possible. I appreciate those who work to democratize information, driving progress and empowerment worldwide.

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. 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	• `msa/` (aligned FASTA)	• `codemloutput/<species>/M0/results`

	runs across species.	• treefiles/ • Template codeml.ctl	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

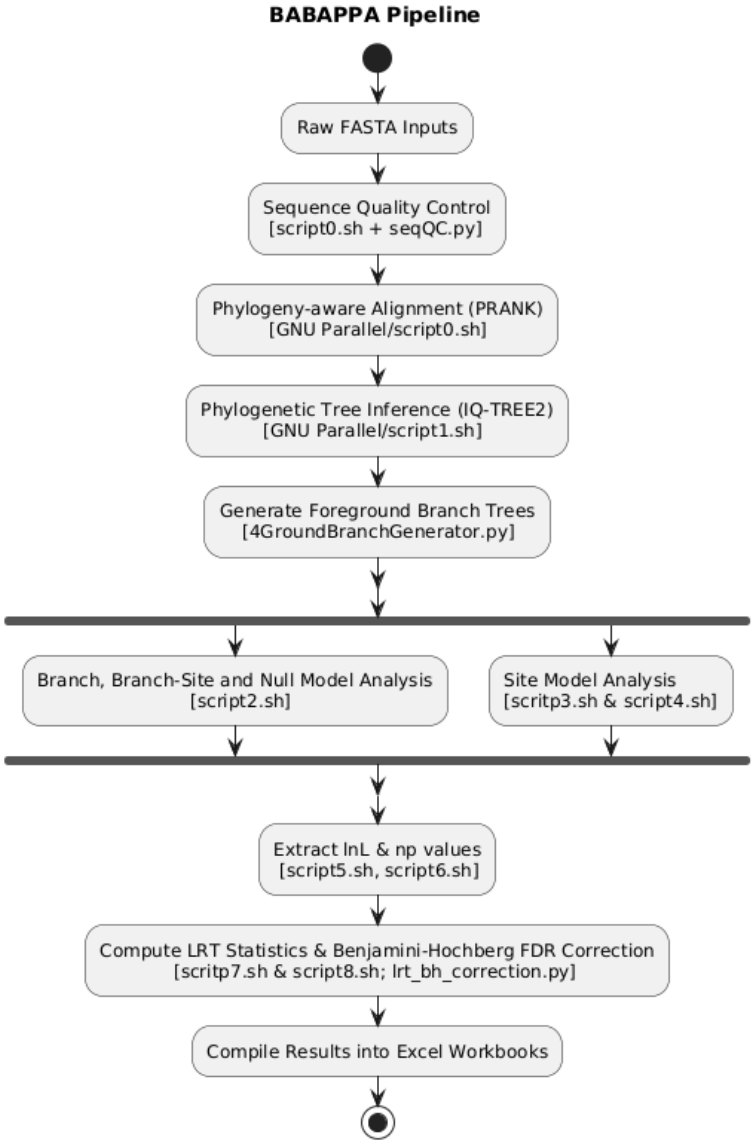
Schematic workflow of BABAPPA and comparative runtime performance against the standard codeml workflow.

13.2. Figure 2

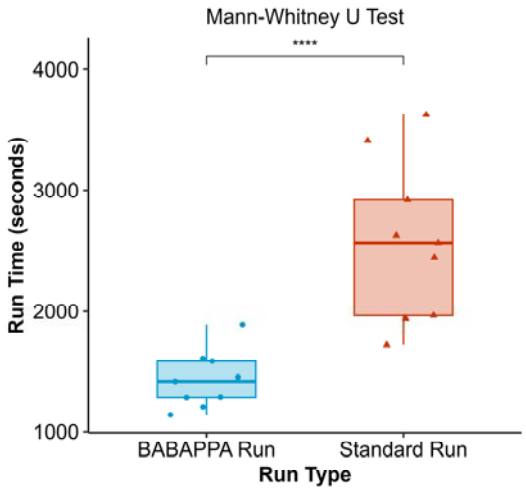
Benchmarking of BABAPPA run with standard run.

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