Supplementary material for "JaCK & Jill: exact marginal likelihoods for small phylogenetic trees"

Matthew D. Schwartz Scott V. Edwards Paul O. Lewis

phyloexact 0.2.3
October 1, 2026

PRELIMINARY

Contents

The phyloexact 0.2.3 manual **2**

S1 Installation, data, and a first session **2**
S1.1 Requirements and installation . . . 2
S1.2 Input files, formats and taxon order . . . 4
S1.3 The four kinds of answer . . . 5
S1.4 A first session . . . 7
S1.5 Running the validation suite . . . 9
S1.6 Version history, license, citation . . . 9
S1.7 Experimental interfaces in 0.2.x . . . 10

S2 Calls, options, results and refusals **10**
S2.1 `px.evidence` . . . 10
S2.2 `px.compare` . . . 12
S2.3 `px.compare_models` . . . 12
S2.4 `px.decidability` . . . 13
S2.5 `px.grade_model`, the ceiling-gap meter . . . 13
S2.6 `px.check_adequacy` and `px.prior_predictive` . . . 14
S2.7 Tree-free model comparison . . . 14
S2.8 `px.quintet.adjudicate` . . . 15
S2.9 Keyword options . . . 15
S2.10Returned objects . . . 15
S2.11Refusals . . . 15

S3 Models, priors, errors, costs and benchmark **17**
S3.1 What is computed . . . 17
S3.2 Models by kind of answer . . . 19
S3.3 Per-model notes . . . 19
S3.4 The estimate and its error statement . . . 23
S3.5 Cost planning . . . 25
S3.6 Adding a model . . . 27

S3.7 Benchmark design and full tables . . . 28
S3.8 K2P, incomplete-gamma and version notes . . . 32
S3.9 Source of the MrBayes stepping-stone offset . . . 35
S3.10 Data behind the example analyses . . . 36

S4 Certificates and genome-scale tools **36**
S4.1 Purpose of a certificate . . . 36
S4.2 Fields . . . 37
S4.3 `px.verify_certificate` . . . 40
S4.4 Integrity of the reference files . . . 41
S4.5 The window-scan tool . . . 41
S4.6 Quartet weight files . . . 44
S4.7 Exact fit ceilings at alignment scale . . . 44

The phyloexact 0.2.3 manual

This supplement is the user manual for phyloexact 0.2.3, the package described in the main text. Section S1 covers installation, input data, the four kinds of answer, and a complete first session. Section S2 is the reference for every call, its options, the objects returned, and the refusals a user can meet. Section S3 lists the models, priors, error statements and costs, explains how to add a model, and gives the full benchmark design and tables behind Section 5 of the main text. Section S4 documents the certificate that accompanies every number, the routine `px.verify_certificate`, and the command-line tools for genome window scans and quartet weight files. Every command and every printed value in this supplement was produced by the 0.2.3 source tree on one Linux server, the benchmark server of Section S3.7; wall-clock times name the thread count. The repository pages `GUIDE.md`, `MODELS.md` and `SPEC.md` remain the full reference for details this manual abridges, and they are versioned with the code.

S1 Installation, data, and a first session

S1.1 Requirements and installation

phyloexact 0.2.3 is a pure-Python package that runs from its source tree. It needs Python 3.10 or later and four libraries: `numpy` (≥1.22), `scipy` (≥1.8), `mpmath` (≥1.2) and `sympy` (≥1.10). Three optional components extend what the package can return. `python-flint`, the Python binding of the Arb ball-arithmetic library (Johansson, 2017), is required for the proven-interval answers; without it an interval request is refused with a message naming the missing library. `numba` compiles the likelihood kernels of the estimator and is optional; the pure-`numpy` path is the automatic fallback and returns identical numbers. At 2.5×10^6 evaluations on tRNA-Gln the 0.2.3 estimator took 31 s per run on the fallback path and one loaded thread (Section S3.7). An earlier release measured its compiled kernel at 21.4 s on one thread and 4.9 s on sixteen, against 29.4 s for the fallback. `torch` with a CUDA device enables the graphics-processor path for exact Jukes–Cantor values at large variable-site counts (Section S3.5). We develop and measure the package on Linux and have not run it on Windows. On macOS the interval engines need the `fork` process start method, which is not the platform default there; the refusal message names the one-line fix, `multiprocessing.set_start_method('fork', force=True)`.

The source tree holds the package directory `phyloexact/`, the reference bundle `pins/` of fixed data (exact values, engines, measurement files, datasets) that the package reads and tests itself

PRELIMINARY

against, the manual pages, and `examples/`. The bundle is bound to the code by content hash, and the tree is therefore installed in place:

```
$ pip install numpy scipy mpmath sympy
$ pip install python-flint          # optional: proven intervals
$ cd jackandjill/                    # the unpacked source tree
$ pip install -e .                   # editable install; or: export PYTHONPATH=$PWD
```

A non-editable build (`pip install .` without `-e`, `pip wheel`, `python -m build`) is refused with `NonEditableBuildRefused`, because the reference bundle lies outside the package directory and 0.2.3 does not yet package it as wheel data. A package directory copied on its own refuses at `import phyloexact` with `BundleLayoutError`. A virtual environment created inside the unpacked directory is a supported layout. Files under `pins/` should not be edited, because each one is opened through a table of content hashes and an edited file stops the self-test by design (Section S4.4).

A two-minute installation test. The first two commands below print the version and the hardware probe that phyloexact runs at import; the third computes one exact marginal likelihood on a distributed alignment and tests it against the data.

```
$ python -c "import phyloexact as px; print(px.__version__)"
0.2.3
$ python -c "import phyloexact as px; print(px.capabilities())"
{'python': '3.12.3', 'platform': 'Linux-6.17.0-1022-gcp-x86_64-with-glibc2.39',
 'cores_logical': 96, 'cgroup_quota_cores': None, 'flint': True, 'flint_error': None,
 'torch': False, 'cuda': False, 'gpu_mem_gb': None, 'numpy': True, 'scipy': True,
 'numba': False, 'cores': 96, 'start_method': 'fork', 'fork': True}
$ python - <<'EOF'
import phyloexact as px
ds  = px.Dataset.from_fasta("pins/parity/datasets/hominid-mt-trnaphe.fasta")
rep = px.evidence(ds, topology="12|34", model="JC69", register="R1")
print(rep.register, rep.logZ, px.verify_certificate(rep.certificate, dataset=ds)["ok"])
EOF
R1 -161.93798030991047 True          # about 17 s on one thread
```

Table S1 lists the fields of the probe and what each one changes. The probe is copied into every result as `capability_snapshot` for information only and is read by nothing.

Table S1: Fields returned by `px.capabilities()`, the import-time hardware and software probe, and what each enables. A request that needs a missing component is refused with a message naming the component.

field	effect
`python`, `platform`	copied into certificates; no routing effect
`cores`, `cores_logical`, `cgroup_quota_cores`	default process count for interval runs and the validation suite (cgroup-aware)
`flint`, `flint_error`	`python-flint` importable: proven intervals available
`numba`	compiled estimator kernels; the `numpy` fallback gives identical values
`torch`, `cuda`, `gpu_mem_gb`	graphics-processor exact path for large m
`start_method`, `fork`	the interval engines need a fork-started worker pool

Threads and processes. The exact engines and the estimator run single-threaded by default, which leaves process-level parallelism to the user. Setting the environment variable

PRELIMINARY

`CE_FASTLIK_THREADS=k` lets the estimator's compiled kernel use k threads; the parallel kernel is numerically identical to the serial one. Interval computations take `procs=` on the call, and the window-scan tool takes `--procs` (Section S4.5). We recommend `OMP_NUM_THREADS=1` when many calls run side by side, which is how every timing in this supplement was taken unless a thread count is stated.

Environment variables. `PHYLOEXACT_PIN_ROOT` points the package at a reference bundle outside the tree; `PHYLOEXACT_FASTBNB_SO` selects a locally compiled five-taxon kernel (Section S2.8); `PHYLOEXACT_KEEP_R2_LOGS=1` keeps the interval engines' progress logs; `CE_NO_NUMBA=1` disables the compiled kernels.

S1.2 Input files, formats and taxon order

One reader. Every alignment phyloexact reads passes through one FASTA parser; there is no NEXUS or PHYLIP reader in 0.2.3, and such files must be converted to FASTA first. It accepts UTF-8 with or without a byte-order mark, drops whitespace inside sequence lines, and upper-cases the rows. It refuses, with a message naming the problem, a bare `>` header, text before the first header, a name that occurs twice, an empty file, and sequences of unequal length. In nucleotide rows `U` is read as `T`, so an RNA-spelled alignment and its DNA spelling are the same dataset at every entry point.

Three constructors. `px.Dataset.from_fasta(path, taxa=[...], alphabet=...)` reads a file; `taxa=` selects and orders a subset of the sequences and is applied before the equal-length test, and a ragged sequence outside the subset therefore does not abort the read. A missing taxon raises `MissingTaxaError`, which names the sequences the file does hold. `px.Dataset.from_rows({name: sequence}, taxa=[...])` takes rows already in memory. `px.Dataset.from_counts({pattern: count}, taxa=[...])` takes a site-pattern count table whose keys are four plain A/C/G/T letters. Counts may be integers, integral floats, or the decimal strings a CSV reader yields, and two spellings of one pattern (`'ACGU'`, `'ACGT'`) are summed. A key carrying a gap or ambiguity code is refused at this constructor; drop such columns first. Every `taxa=` and `subset=` argument is an ordered sequence (list, tuple, array); a `set` is refused because its iteration order would make the same call name different trees on different runs.

Taxon order names the topologies. For four taxa in the order (a, b, c, d) the three unrooted topologies are written `'12|34'` ($ab|cd$), `'13|24'` ($ac|bd$) and `'14|23'` ($ad|bc$). A Newick string over the taxon names is the same request: `'((a,b),(c,d));'`, `'((a,b),c,d);'` and `'12|34'` give the same number and the same output. Branch lengths in a Newick string, a leaf that is not a taxon, a repeated or omitted leaf, and an unresolved tree are refused. The topology argument names a shape; the marginal likelihood integrates the branch lengths. Fewer than four taxa are refused at every entry point, since three taxa have one unrooted tree.

Missing data. At every four-taxon entry point a column carrying a gap or an IUPAC ambiguity code in any row is dropped before counting, and the result states the usable-column count `N_used` and `dropped_columns`. The five-taxon maximum-likelihood module (Section S2.8) instead sums the tip's partial likelihood over the states an ambiguity code admits and keeps the column. The two conventions answer different questions, and both are stated on their outputs. Throughout this manual N is the number of usable columns. m is the number of those columns that vary within a

PRELIMINARY

sister pair of the topology at hand; for `12|34`, the columns where taxa 1 and 2 differ or taxa 3 and 4 differ. The cost of an exact value is governed by m (Section S3.5); `ds.stats(topology)` reports (n_{taxa}, N, m).

Alphabets and SR4 recoding. A dataset carries an `alphabet`, `'nucleotide'` or `'protein'`, detected from the letters or declared by the caller (`alphabet='protein'`). State it for a peptide that happens to use only nucleotide letters, and for a nucleotide file with a stray non-letter character. Amino-acid rows have no nucleotide site patterns, and every nucleotide model refuses them (`px.AlphabetError`). The model label `'JC-SR4'` recodes them on entry into the four SR4 classes of Susko and Roger (2007) (AGNPST, CHWY, DEKQR, FILMV) and treats the recoded rows as a four-state alignment under Jukes–Cantor. The recoder is public, `px.recode_sr4(ds)`, for entry points that serve no recoded label. In protein rows `U` stays selenocysteine and is not a usable SR4 site.

Row order and the dataset hash. No number depends on the order of rows in the file, because the exact answers depend on the multiset of site patterns only. The estimator runs in one canonical frame (taxa sorted by name), and the same labeled quartet in any row order gives the same estimate at the same seed. Every result carries `dataset_sha256`, a hash of the normalized rows written one per line in sorted name order; it is independent of row order and, for protein rows, includes the alphabet.

JSON conventions. Every output document is RFC 8259 JSON. Exact rationals are decimal strings (`"num"`/`"den"` or `"p/q"`), and an infinite float view is the string `"inf"` or `"-inf"`. An integer above 2^{53} in magnitude is written as a decimal string, which keeps a float-only reader from rounding it, and integers of any length are written in sub-quadratic time.

S1.3 The four kinds of answer

Every number phyloexact returns is one of four kinds: an exact value, an interval with proven endpoints, an estimate with a measured error, or a proof that two topologies have equal marginal likelihoods. The package's keyword for the kind of answer is "register", with codes `"R1"` (exact), `"R2"` (proven interval), `"R3"` (estimate) and `"R0"` (tie proof). Below, the codes appear only where a user types or reads them (Table S2).

Exact values. Under the Jukes–Cantor (Jukes and Cantor, 1969) and Kimura (Kimura, 1980, 1981) models the site likelihood is a polynomial in the transformed branch lengths $x_e = e^{-4t_e/3}$, and under the default prior each x_e is uniform on $(0, 1)$. The marginal likelihood is then a ratio of two integers, which phyloexact computes in integer arithmetic and returns as such; Section S3.3 summarizes how, and the companion paper gives the mathematics (Schwartz et al., 2026a).

Proven intervals. For HKY85 (Hasegawa et al., 1985) and GTR+Γ_4 (Tavaré, 1986; Yang, 1994) the site likelihood is not such a polynomial, and 0.2.3 has no exact engine for them (Section S3.3). Instead, phyloexact returns an interval whose endpoints are exact rationals assembled from ball arithmetic (Johansson, 2017) and are guaranteed to bracket Z. The label reports what the computation closed within its budget, `"R2"` when the lower endpoint is positive and `"R2-one-sided"` (a proven upper bound only) when it is not.

PRELIMINARY

Table S2: The four kinds of answer. "Models" abbreviates Table S8; sizes are for four taxa unless stated; N usable columns, m variable columns in the sense of Section S1.2.

code	kind	what `rep.value` holds	available for
`"R1"`	exact value	a Python `Fraction`, the marginal likelihood Z as a ratio of two integers; `rep.logZ` is the nearest double-precision value to $\ln Z$	four taxa only: JC69 and JC-SR4 to $N \leq 150$; K2P to $N \leq 35$, $m \leq 11$; K3ST to $N \leq 5$; JC+R at micro scale
`"R2"`	proven interval (enclosure)	the float view of $\ln Z$; `rep.interval` $= (\ln Z_{\text{lo}}, \ln Z_{\text{hi}})$, outward-rounded views of exact rational endpoints	four taxa: JC69, JC-SR4, JC+R (always two-sided); HKY85 and GTR+Γ_4 at a fitted or user-fixed parameter point; HKY85 with κ free
`"R3"`	estimate with measured error	a float $\ln Z$; `rep.error_nats` = the measured error figure, or `None` outside a measured size class	JC69 at any taxon count $n \geq 4$ (a Newick topology); JC-SR4, K2P, HKY85, GTR+Γ_4, user plug-ins (four taxa)
`"R0"`	tie proof	no value: the stabilizer group of the site-pattern counts, the orbits of topologies it forces to tie, and the permutation that proves each tie	any leaf-exchangeable model; $4 \leq n \leq 32$ taxa

Estimates with a measured error. The estimate is a randomized quasi-Monte Carlo importance-sampling value (Owen, 1997; Dick et al., 2013). Its error figure is empirical and is not a standard error. The figure is three times the largest deviation from exact values measured on a swept class of alignments at the same budget, combined with the spread of four internal replicates as max(class tolerance, $3 \times$ spread). The default budget of 10^5 evaluations carries a figure of 0.014 in natural-log units (log units below; the program prints them as nats) for JC69 quartets inside the swept class. Outside the swept class `rep.error_nats` is `None` and the result's first note states the cause (Section S3.4).

Tie proofs. When a relabeling of the taxa leaves the site-pattern counts unchanged and maps one topology onto another, the two marginal likelihoods are equal under every model and prior in the stated class. In that case phyloexact returns the proof instead of two numbers. The proof is stated for one of three nested model classes. Class `raw` is any model with exchangeable leaves, GTR and mixtures included; `folded_k2p` comprises the models symmetric under base relabelings that preserve the purine/pyrimidine split, namely K2P and JC69. Class `folded` comprises the base-symmetric models JC69, JC-SR4 and JC+R.

Asking for a kind. Every call that computes a marginal likelihood takes `register=`. The default `"auto"` returns the exact value when the alignment lies inside the measured size limits of Table S11. Otherwise it returns the estimate, labeled as such, with a note naming N, m and the measured run time of the exact computation it declined. `"auto"` does not start an interval computation on its own; request one with `register="R2"` or `target="enclosure"`. A forced kind (`"R1"`, `"R2"`, `"R0"`) is either delivered or refused with an exception naming the reason and the nearest request that would succeed (Section S2.11). `target="exact"` under `"auto"` runs the exact computation

PRELIMINARY

at any m inside its hard limits. `target=` accepts exactly `'enclosure'`, `'rigorous'`, `'certified'` (all three meaning the interval) and `'exact'`, and a contradictory pair such as `register="R3", target="exact"` is a `ValueError`. A quantity read off several marginal likelihoods (a Bayes factor, a topology posterior, a model-comparison row) carries the weakest kind among its parts. Exact values and intervals are four-taxon computations in 0.2.3. For five or more taxa phyloexact 0.2.3 offers only the JC69 estimate (with no error figure, because the size classes with a measured error cover only quartets) and tie proofs for up to 32 taxa. A request for an exact five-taxon value is refused. The separate call `px.quintet.adjudicate` ranks the fifteen trees on five taxa by JC69 maximum likelihood with proven bounds but computes no marginal likelihood (Section S2.8).

S1.4 A first session

The session below uses two alignments distributed with the package. The hominid mitochondrial tRNA-Phe quartet (human, chimpanzee, gorilla, orangutan) has $N = 69$ after 4 gapped columns are dropped and $m = 10$ on every topology. The tRNA-Gln quartet of the main text (mouse, rat, chicken, *Xenopus*) has $N = 68$ and $m = 25$, 32, 33 on the three topologies. Printed values are abridged with `...`; run times are one thread on the benchmark server of Section S3.7 under full load.

```
>>> import phyloexact as px
>>> ds = px.Dataset.from_fasta("pins/parity/datasets/hominid-mt-trnaphe.fasta")
>>> ds, ds.stats()
(Dataset(sequences, n_taxa=4, N=69, dropped=4), DatasetStats(n_taxa=4, N=69, m=10, ...))
>>> ds.taxa
['human', 'chimp', 'gorilla', 'orangutan']

>>> # the tie proof first: no arithmetic (0.09 s)
>>> r0 = px.decidability(ds)
>>> r0["stabilizer"]["folded"]["forced_ties"], r0["stabilizer"]["folded"]["generators"]
([['12|34', '14|23']], [[2, 1, 0, 3]])   # equal evidence under every base-symmetric model
>>> px.compare(ds, "12|34", "14|23", model="JC69")["verdict"]
'forced-tie'                          # log_bf 0 exactly; the permutation is returned

>>> # evidence summed over the three topologies; 'auto' delivers exact values here
>>> rep = px.evidence(ds, topology="all", model="JC69", register="auto")   # 48 s
>>> rep.register, rep.winner
('R1', '13|24')
>>> {T: r.logZ for T, r in rep.per_topology.items()}
{'12|34': -161.93798030991047, '13|24': -161.9366608915647, '14|23': -161.93798030991047}
>>> rep.posterior                              # exact Fractions; float views shown
{'12|34': 0.33318..., '13|24': 0.33362..., '14|23': 0.33318...}

>>> cmp = px.compare(ds, "13|24", "12|34", model="JC69")
>>> cmp["verdict"], cmp["register"], cmp["log_bf"]
('computed', 'R1', 0.0013194183457577633)   # exact log Bayes factor; cmp["bf_exact"] is the
    rational

>>> v = px.verify_certificate(rep.certificate, dataset=ds)
>>> v["ok"], v["n_performed"], v["n_skipped_in_legs"]
(True, 27, 3)
```

The shallow hominid locus has no topological signal under Jukes–Cantor, since two topologies tie by symmetry and the third leads by an exact log Bayes factor of 0.0013 log units. The three named skips (`n_skipped_in_legs`) are the full exact recomputations of the three per-topology values, which `px.verify_certificate` runs unasked only up to $N = 40$ (Section S4.3).

PRELIMINARY

The tRNA-Gln quartet is deeper, and its three topologies straddle the default size limit for exact values ($m \leq 25$ at $N \leq 68$; Table S11).

```
>>> gln = px.Dataset.from_fasta("pins/phylo_gkz/realdata/trnaq_quartet.fasta")
>>> [gln.stats(T).m for T in ("12|34", "13|24", "14|23")]
[25, 32, 33]
>>> [px.route("JC69", gln.stats(T), "auto", px.capabilities()).register for T in ("12|34",
    "13|24", "14|23")]
['R1', 'R3', 'R3']                    # what 'auto' would deliver per topology, no compute

>>> ex = px.evidence(gln, "12|34", "JC69", register="R1")              # 356 s, one thread
>>> ex.logZ, ex.interval, ex.error_nats
(-261.40306415034735, (-261.4030641503474, -261.40306415034735), 0.0)
>>> len(str(ex.value.numerator)), len(str(ex.value.denominator))
(188, 302)

>>> est = px.evidence(gln, "12|34", "JC69", register="R3", budget=100000, seed=0)   # 6.2 s
>>> est.logZ - ex.logZ, est.error_nats
(-8.83e-05, None)
>>> est.certificate["notes"][0]
'R3 estimate (float Monte Carlo), UNGRADED (regime): ... N = 68 usable sites exceeds the record's
    largest
 swept case N = 32; m = 25 ... exceeds ... m = 9 ...; replicate spread 0.000282035 nats over 4
    internal
 replicates; no error figure is claimed for this number'

>>> allT = px.evidence(gln, topology="all", model="JC69", register="R3", budget=100000, seed=0)
>>> allT.posterior                                                  # 13 s
{'12|34': 0.9999993409513044, '13|24': 4.84e-07, '14|23': 1.75e-07}

>>> k2p = px.evidence(gln, "12|34", "K2P", register="R3", budget=100000, seed=0)    # 5.9 s
>>> k2p.logZ, k2p.certificate["replicates"]["spread_nats"]
(-259.6388999380458, 0.0148547)
>>> px.compare_models(gln, "12|34", models=("K2P", "JC69"), register="R3").log_bf
3.791578082126705                     # K2P over JC69, proper-prior log Bayes factor, no error
    figure

>>> px.evidence(gln, "12|34", "K2P", register="R1")
RegisterUnavailable: K2P ... N=68 > ... cutoff 40 ... [unlock: truncate to a shorter locus, ...
 (model='K2P-gate'), or register='R3' ...]

>>> hky = px.evidence(gln, "12|34", "HKY85", register="R2", minutes=3, procs=1)     # 184 s
>>> hky.register, hky.interval, hky.certificate["pinned_point"]["kappa"]
('R2-one-sided', (-inf, -230.79193341269783), '413/100')

>>> px.verify_certificate(ex.certificate, dataset=gln)["ok"]
True                                  # 18 tests performed, 1 named skip: value not recomputed
    (N=68 > 40)
>>> px.verify_certificate(ex.certificate, dataset=gln, strict=True)["ok"]
False                                 # strict counts the named skip as a failure; recompute=True
    redoes it
```

The routing decision is available without computing anything (`px.route`), and its notes name the measured run time behind it. An estimate outside the measured size class prints its replicate spread and no error figure, even when, as here, it lies within 10^{-4} log units of the exact value. The interval label reports what the computation closed. Here three minutes on one process gave a proven upper bound for HKY85 at $\kappa = 413/100$; a two-sided interval on this quartet is a long multi-process run (Table S12). The last two lines are the expected outcome above 40 sites, where the value is

PRELIMINARY

not recomputed unasked; `strict=True` counts that named skip as a failure and `recompute=True` removes it (Section S4.3).

S1.5 Running the validation suite

One command, run from any writable working directory, executes the whole suite and writes the report `VALIDATION_REPORT.json` there:

```
$ python -m phyloexact.validate                          # every test; nonzero exit on any failure
$ python -m phyloexact.validate --report /path/VALIDATION_REPORT.json
$ python -m phyloexact.validate --gates V2,V9            # a named subset
$ python -m phyloexact.validate --parity                 # also recompute the distributed exact
                                                         #   reference values (m up to 33)
$ python -m phyloexact.validate --parity --full          # every distributed value (m up to 59;
    hours)
$ python -m phyloexact.validate --tree-sha               # hash of the delivered files
```

The suite has 38 numbered tests. They compare every exact engine with an independent evaluator on fixed inputs, and confirm that every interval contains the exact value and every estimate reproduces it within the printed error wherever an exact value exists. They also re-run the symmetry, recoding, adequacy, tree-free, scan, weight-file, ceiling and five-taxon tools on fixtures with known answers. Further tests plant deliberately corrupted certificates and reference files, each of which must be rejected by name, and guard the manual pages against drift from the code. `GATES.md` in the repository lists all 38 one line each and is regenerated from the code by `--write-gates-md`. The suite needs no graphics processor and no external data, and a missing optional component is reported as a named skip that does not fail the run. Run times depend on the host and are printed in the report. For orientation, an earlier release's default suite (23 tests) ran in 63 minutes on the benchmark server of Section S3.7; 0.2.3 quotes no time of its own. `px.validate(gates="V1,V24", report=PATH)` is the same suite from Python.

S1.6 Version history, license, citation

- 0.1.0: exact Jukes–Cantor and Kimura values and JC69 intervals; the first estimator (JC69, K2P, HKY85, GTR+Γ_4, GM); tie proofs for four and five taxa; `px.verify_certificate`. Also the ceiling-gap meter, the exact posterior-predictive check, tree-free model comparison, and the window-scan, ceiling and weight-file tools. The comparison programs' rows and the timing sweep of Section S3.7 date from this version's benchmark campaign; the phyloexact accuracy rows there were re-run under 0.2.3.
- 0.2.0: the source release that accompanied the first preprint; JC+R and K3ST exact at micro scale; tie proofs to 32 taxa; `topology="all"`; the plug-in examples. Also proven intervals at fitted HKY85 and GTR+Γ_4 points, for JC+R, and for HKY85 with κ free, and interval posterior-predictive checks.
- 0.2.1 (not released): model comparisons and model grades report the proper-prior log Bayes factor as their headline; results of these two kinds from 0.2.0 must be re-emitted.
- 0.2.2: the estimator changed to a multi-start mixture with four internal replicates whose spread every result carries; earlier estimates cannot be reproduced by this estimator and are refused as a version gap.
- 0.2.3: error figures read off a ten-seed sweep against exact values; measured size limits for the default routing; the general-Markov estimator withheld; a warranted core and an experimental

PRELIMINARY

set of interfaces (Section S1.7). Also a coverage table for `px.verify_certificate` and the source-tree install tests.

The code is under the MIT license and the manual pages under CC BY 4.0. Please cite the main paper for the package and Schwartz et al. (2026a) for the mathematics of the exact values; the `suite_version` field of any result (`phyloexact-0.2.3`) names the version that produced it.

S1.7 Experimental interfaces in 0.2.x

Release 0.2.x has a warranted core and an experimental set of interfaces. Experimental means available and tested, with every result labeled by its kind, but the interface may change in release 0.2.4, and verification of certificates across package versions is promised only for the core. The experimental set is:

- the tree-free comparison calls `px.pail_evidence`, `px.pail_row` and `px.pail_aggregate` (Section S2.7);
- `prior=` with a non-default prior, and `px.PlugIn` user models (Sections S3.1 and S3.6);
- the genome-scan command-line tools and functions `python -m phyloexact.hill.scan`, `.weights`, `.verify_cert` and `px.hill.sup_ceiling_partitioned` (Sections S4.5 to S4.7);
- the options `cap=`, `total_budget=` and `nstarts=` of `px.quintet.adjudicate` (Section S2.8);
- the estimate form of `px.check_adequacy`, and `px.prior_predictive` (Section S2.6);
- `dataset=` given to `px.verify_certificate` as a path, a mapping or a superset (a `Dataset` object is core; Section S4.3);
- certificates written by unreleased development builds (those of release 0.2.0 are core).

All other calls, kinds of answer and certificates are the core. Where this manual describes an experimental interface, the text marks it "(experimental in 0.2.x, Section S1.7)". Known issues and the 0.2.4 backlog are listed in `ISSUES.md` at the repository root.

S2 Calls, options, results and refusals

Every entry point is a plain function on the module `px` (`import phyloexact as px`), and Table S3 lists them with their inputs and outputs. The subsections give each call's signature, options, returned object and one executed example; Tables S4 to S6 collect the keyword options, the attributes of the returned objects, and the refusals.

S2.1 `px.evidence`

```
px.evidence(ds, topology="12|34", model="JC69", register="auto", target=None, *,
            prior=None, budget=100000, seed=0, minutes=5.0, procs=None, prec=128,
            mode=None, subset=None, topology_prior=None, box_path="v3", verbose=False,
            n_kappa=None, pinned_point=None)  ->  EvidenceReport | TopologySumReport
```

`ds` is a `Dataset`; `subset=[four names]` selects a quartet from a larger one. `topology` is a key, a Newick string, or `"all"`. `model` is a label of Table S8 or a `px.PlugIn`. `register` and `target` choose the kind of answer (Section S1.3). The remaining keywords apply to one kind each (Table S4).

PRELIMINARY

Table S3: Entry points of phyloexact 0.2.3. "Kinds" are the answer kinds of Table S2 the call can return. Run times are single-thread examples from this supplement's sessions on the named alignment; they scale with m as in Table S11.

call	input	returns	kinds; example run time
`px.Dataset.from_fasta / from_rows / from_counts`	FASTA path, {name: row}, or {pattern: count}	a `Dataset` (`.N`, `.taxa`, `.stats()`)	n/a
`px.evidence(ds, topology, model, register)`	one topology, or `"all"`	`EvidenceReport` or `TopologySumReport` with certificate	all four; 18 s exact (tRNA-Phe), 6 s estimate
`px.compare(ds, A, B, model)`	two topologies (keys or Newick), $4 \le n \le 32$	dict: tie proof, or two marginal likelihoods and their log Bayes factor	all four; 0.004 s tie, 31 s exact
`px.compare_models(ds, topology, models)`	one topology, two model labels	`ModelComparisonReport`: Bayes factor (exact `Fraction` when both exact)	exact / interval / estimate; 139 s exact ($N = 16$), 12 s estimate
`px.decidability(ds, label=, topology=)`	any dataset, $4 \le n \le 32$	tie-proof certificate (stabilizer groups, orbits, forced ties)	tie proof; 0.09 s
`px.grade_model(ds, rep)`	a dataset and an evidence report on it	model-grade certificate: ceiling gap in log units per site	kind of `rep`; $<$ 0.01 s
`px.check_adequacy(ds, topology, model, register)`	four taxa	adequacy certificate: per-class posterior-predictive table and discrepancy D	exact / interval (JC69) / estimate
`px.prior_predictive(model, topology)`	no data	the $N = 0$ predictive over pattern classes	exact
`px.pail_evidence`, `px.pail_row`, `px.pail_aggregate`	four taxa; one or more models	topology-averaged marginal likelihood per model; between-model differences; a corpus summary	exact / estimate; 28 s
`px.hill.sup_ceiling_from_counts` and kin	any alignment or pattern-count table	exact upper bound on every independent-sites model's fit	exact; 2.3 s at 2,000 taxa $\times 10^5$ sites
`px.quintet.adjudicate(fasta, name_map=)`	five sequences	maximum-likelihood ranking of the fifteen trees on five taxa with proven bounds (not a marginal likelihood)	its own kind (Section S2.8)
`px.verify_certificate(cert, dataset=)`	a certificate and (optionally) the data	dict: `ok`, per-test rows, skip counts	0.06 s to minutes (Section S4.3)
`px.PlugIn(loglik_fn, d, name=)`	a vectorized log-likelihood on $[0,1]^d$	a model object accepted by every call that computes a marginal likelihood	estimate (Section S3.6)
`px.BetaPrior(a, b)`	two shapes in $[1, 10^6]$	a prior object for `prior=`	estimate, no error figure (experimental in 0.2.x, Section S1.7)
`px.route(model, ds.stats(T), register, px.capabilities())`	no compute	the routing decision and its notes, or the refusal	
`px.capabilities()`, `px.validate()`, `px.recode_sr4(ds)`, `px.canonical_topology(ds, t)`, `px.FOLD_LABELS`		probe; validation report; recoded `Dataset`; canonical key; the 15 fold-class labels	

PRELIMINARY

One topology. The call returns an `EvidenceReport`. For an exact value `rep.value` is the `Fraction` Z and `rep.logZ` the nearest double-precision value to $\ln Z$. `rep.interval` is then a pair of doubles at most two units in the last place wide that encloses $\ln Z$, and `rep.error_nats` is `0.0`. For an interval `rep.interval` holds outward-rounded float views of the exact endpoints, which satisfy $\text{lo} \le \ln Z_{\text{lo}} \le \ln Z \le \ln Z_{\text{hi}} \le \text{hi}$. `rep.width` and `rep.error_nats` give its width, and `rep.one_sided` flags a proven upper bound only. For an estimate `rep.value` is the float $\ln Z$ and `rep.error_nats` the measured error figure or `None`. When `"auto"` stops at an estimate for a model that also offers an interval, `rep.meta["r2_available"]` names that option and its cost.

All three topologies. `topology="all"` returns a `TopologySumReport` with the total $Z_{\text{tot}} = \sum_T p(T)\, Z_T$ over the three quartet topologies and the posterior $P(T \mid \text{data}) = p(T) Z_T / Z_{\text{tot}}$. The topology prior $p(T)$ is uniform unless `topology_prior={topology: weight}` is given; integers, `Fraction`s and `"a/b"` strings are exact, and a float is read as the decimal typed. `rep.posterior` holds exact `Fraction`s when all three values are exact, floats for estimates, and per-topology pairs $(P_{\text{lo}}, P_{\text{hi}})$ of proven bounds for intervals. `rep.winner` is the posterior mode, or, for intervals, the topology whose lower bound exceeds every rival's upper bound (else `None`). `rep.register` is the weakest kind among the three per-topology results, `rep.per_topology` maps each key to its `EvidenceReport`, and its certificate is a compound of kind `topology-sum` embedding the three per-topology ones. With `topology="all"` five-taxon data are refused and referred to `px.quintet`, and larger taxon sets are refused because no engine in the package enumerates their topologies. A single Newick topology on five or more taxa is served by the JC69 estimate (Section S2.2).

S2.2 `px.compare`

```
px.compare(ds, topoA, topoB, model="JC69", register="auto", **kw)  ->  dict
```

The call first computes the stabilizer of the site-pattern counts and asks whether a relabeling of the taxa carries `topoA` to `topoB` within the symmetry class the model admits (Section S1.3). If so it returns the tie with no arithmetic, for example `{'verdict': 'forced-tie', 'register': 'R0', 'log_bf': 0, 'tie_register': 'folded', 'witness': [2, 1, 0, 3], ...}` on the tRNA-Phe quartet for `12|34` against `14|23`. Otherwise it computes both marginal likelihoods at whatever kind the model delivers and returns `verdict 'computed'`. The dict then carries `log_bf`, the exact ratio `bf_exact` when both marginal likelihoods are exact, `error_nats` (the two error figures summed), `label`, `leg_registers`, the two `EvidenceReports` `A` and `B`, and the tie proof computed first. Topologies are quartet keys, quintet keys, or Newick strings over the taxa for $4 \le n \le 32$; two spellings of the same unrooted tree are refused as one topology given twice. On five or more taxa an untied pair is computed under JC69 as an estimate, by pruning on the unrooted tree the Newick names. Only this JC69 estimate serves any taxon count; other models refuse beyond four taxa. `register="R0"` forced on a pair the stabilizer does not tie is refused, since a tie proof carries no marginal likelihood values.

S2.3 `px.compare_models`

```
px.compare_models(ds, topology, models=("JC69", "K2P"), register="auto", target=None, *,
                  allow_mixed_registers=False, subset=None, **kw)  ->  ModelComparisonReport
```

The Bayes factor of `models[0]` over `models[1]` on one dataset and one topology. Under `"auto"` both marginal likelihoods are computed at the best kind both models can deliver. For example, K2P against JC69 on a 68-column quartet runs both as estimates, and the report notes that JC69 alone

PRELIMINARY

could have been exact. A forced `register` applies to both models; `target="enclosure"` makes the interval the only shared kind. Mixing kinds is refused unless `allow_mixed_registers=True`, in which case the comparison carries the weaker kind. The headline `log_bf` is the proper-prior log Bayes factor, in which each marginal likelihood is divided by its prior's total mass before the ratio is taken. That matters only for the Kimura models' default integration domains (mass $(2/3)^5 = 32/243$ for K2P; Section S3.1). When both marginal likelihoods are exact, `rep.bf` is the exact `Fraction` and `error_nats` is 0. On the 16-column prefix of tRNA-Gln ($N = 16$, $m = 9$) the exact log Bayes factor of K2P over JC69 is 0.6921 log units, an 80-digit over 80-digit rational, in 139 s. On the full 68 columns the same call at `register="R3"` gives 3.7916 log units for K2P with `error_nats None`, both marginal likelihoods being outside the measured size class. Two spellings of one model (`("JC69", "jc")`) are refused; the same string twice is the explicit self-comparison control and returns a log Bayes factor of 0 within `error_nats`. The certificate is a compound of kind `model-comparison` whose `comparison` block lists both models, their kinds, their prior masses, the exact ratio when it exists, and how the error figure was formed.

S2.4 `px.decidability`

```
px.decidability(ds, label="", topology=None, model=None)  ->  dict   # the tie-proof certificate
```

The tie proof for a dataset of $4 \le n \le 32$ taxa, computed without reference to any model's likelihood. For each of the three symmetry classes it carries the stabilizer group of the site-pattern counts as generators and order, the taxon classes that confined the search, and the orbits of topologies. Every orbit is listed for $n \le 7$ (over 3, 15, 105, 945 topologies), and for $n \le 5$ the forced pairs among the package's quartet and quintet keys are listed too. For $n \ge 8$ the full topology set is refused and `topology=` (a Newick string) returns that topology's orbit, refused past 20,000 members with a proven lower bound on the orbit size stated instead. The search does not enumerate the symmetric group, and a search past its node budget (400,000 nodes) is refused without a partial answer. `label=` is a caller's name for the window or run, carried in the output and read by nothing; `model='JC-SR4'` makes the statement about SR4-recoded amino-acid rows. `px.verify_certificate` takes 0.003 s here; on a highly symmetric taxon set (many identical rows) it can cost far more than the proof did, because it rebuilds the group from the generators.

S2.5 `px.grade_model`, the ceiling-gap meter

```
px.grade_model(ds, rep, convention="ambient", alpha=1)  ->  dict   # kind 'model-grade'
```

Given a dataset and an evidence report on it, the meter returns the gap, in log units per site, between the model's proper-prior marginal likelihood and the best fit any independent-sites model could reach on this alignment. That best fit is the unconstrained multinomial likelihood $\prod_k (n_k/N)^{n_k}$ over the observed site-pattern counts n_k (Goldman, 1993), an upper bound on every such model's marginal likelihood under every prior, computed here as an exact rational. The headline field is `meter["gap_sup_nats_per_site_HEADLINE"]`; two Dirichlet-smoothed references (`ambient`, over all 4^n patterns, and `observed`) are quoted beside it as declared references whose sign is part of the result. On the tRNA-Gln quartet under exact JC69 the headline gap is 0.9066 log units per site (61.65 log units over $N = 68$; ambient reference -0.993, observed 0.540). The gap carries the weaker kind of its two inputs, and two models' headlines on one dataset differ by exactly their proper-prior log Bayes factor per site. Gaps at different alignment sizes are not comparable, because the finite-size headroom of the multinomial bound grows with the number

PRELIMINARY

of distinct patterns; compare models within an alignment. Above 43,150 sites the exact references would exceed the digit ceiling for exact integers (Section S4.3) and the call refuses, naming `px.hill.sup_ceiling_from_counts` as the tool at that scale (Section S4.7).

S2.6 `px.check_adequacy` and `px.prior_predictive`

```
px.check_adequacy(data, topology="12|34", model="JC69", register="auto", budget=None, seed=None,
                  taxa=None, progress=None, workers=None, cache_dir=None, prec=128, procs=None)
    ->  dict
px.prior_predictive(model="JC69", topology="12|34")  ->  dict
```

An exact posterior-predictive check (Bollback, 2002) on the model's own sufficient statistics. For each pattern class k with observed count vector n, the predictive probability of one new site falling in class k is $p_{\mathrm{pred}}(k \mid n) = w_k\, Z(n + e_k)/Z(n)$, a ratio of two marginal likelihoods. Here e_k adds one site to class k and w_k is the class multiplicity, and the code confirms $\sum_k p_{\mathrm{pred}}(k \mid n) = 1$ as an exact identity. The report tabulates observed frequencies against predictives and returns the discrepancy $D = \mathrm{KL}(\text{observed} \,\|\, \text{predictive})$ in log units per site. JC69 uses the 15 base-symmetric pattern classes (`px.FOLD_LABELS`) and K2P the 36 classes symmetric under purine/pyrimidine-preserving relabelings; K2P is served exactly only, inside its exact size limits. `register="R2"` (JC69 quartets, $N \le 149$, needs `python-flint`) returns the same table with proven bounds from sixteen intervals, with restart files under `cache_dir=`. `budget=`, `seed=` and `workers=` are refused there, and `progress=` takes an optional callback. `register="R3"` (experimental in 0.2.x, Section S1.7) tabulates raw site patterns for HKY85, GTR+Γ_4 and JC69 with a labeled Monte Carlo standard error and an effective sample size; below 5 effective samples the table is flagged and `strict=True` refuses it. The call is four-taxon only and names `taxa=[four names]` as the remedy on larger datasets; K3ST and JC+R are refused on every kind. `px.prior_predictive` (experimental in 0.2.x, Section S1.7) is the $N = 0$ case, the model's prior distribution over pattern classes for one site.

S2.7 Tree-free model comparison

```
px.pail_evidence(ds, model="JC69", register="auto", *, subset=None, **kw)   ->  PailReport
px.pail_row(ds, models=("JC69", "K2P"), register="auto", *, allow_mixed_registers=False,
            subset=None, **kw)                                              ->  PailRowReport
px.pail_aggregate(rows, labels=None)                                        ->  dict
```

These calls (experimental in 0.2.x, Section S1.7) compare substitution models on a quartet without choosing a tree. `pail_evidence` returns one model's marginal likelihood averaged over the three topologies with weight 1/3, $\bar{Z} = (Z_{12|34} + Z_{13|24} + Z_{14|23})/3$, exact as a `Fraction` when all three are exact; `ln_Zbar_normalized` divides by the model's prior mass. `pail_row` computes $\bar{Z}$ for each model in `models` and every pairwise difference $\Delta_{A-B} = \ln(\bar{Z}_A/\mathrm{vol}_A) - \ln(\bar{Z}_B/\mathrm{vol}_B)$; its flat `.row` names `model_order`, each model's $\ln \bar{Z}$, kind, winning topology and three-way-tie flag, and the differences. Keyword arguments (`budget`, `seed`, `minutes`, `mode`, ...) are forwarded to `px.evidence` for every component marginal likelihood. Under `"auto"` each model runs at its own best kind, and a row that would place an estimate beside an exact value is refused with the two requests that would succeed. These are `register="R3"` for both models, or `allow_mixed_registers=True`, which labels every difference with the weaker kind of its pair. On tRNA-Gln at `register="R3"` the row gives $\ln \bar{Z}$ (normalized) $= -258.710$ for K2P and -262.502 for JC69, $\Delta_{\mathrm{K2P-JC}} = 3.7916$ log units, winning topology `12|34` under both, in 28 s; `px.verify_certificate` passes all 60 tests on the row. The difference equals the fixed-topology Bayes factor of Section S2.3 to four figures

PRELIMINARY

because `12|34` carries all but 7×10^{-7} of the posterior mass. `pail_aggregate(rows)` sums saved rows over a corpus, reading the first and last model of `model_order` as the pair to report; rows saved as JSON with sorted keys aggregate the same. The eighteen-locus demonstration of the main text is distributed as `pins/pail_demo/`, and the validation suite recomputes its exact values.

S2.8 `px.quintet.adjudicate`

```
px.quintet.adjudicate("five_taxa.fasta")                          # records named H, OT, CO, RN, OG
px.quintet.adjudicate("my_five.fasta", name_map={"H": "taxonA", "OT": "taxonB", "CO": "taxonC",
                                                 "RN": "taxonD", "OG": "taxonE"},
                      cap=200000, total_budget=1500000, nstarts=..., seed=...)
```

The five-taxon module ranks the 15 rooted five-lineage shapes by maximum likelihood under JC69 with an exact bound on the rank, using interval branch-and-bound with outward-rounded floating point. It returns a ranking and not a marginal likelihood; this is a kind of answer of its own, weaker than an exact value, and its report says so. The five sequences must be named by the seat labels or mapped with `name_map=`. The default pass caps box evaluations at 25,000 per rival and 100,000 per call, with one retry at 75,000 and 300,000 when the pass abstains with two to four surviving rivals. `cap=` and `total_budget=` (experimental in 0.2.x, Section S1.7) run a single larger pass, and the report's `box_evals_total` says what was spent. An abstention is itself a result; it means a rival on status `BUDGET` ran out of evaluations, and a long or strongly resolved locus can need a larger single pass. This module keeps gapped and ambiguous columns (Section S1.2), and its report states both column counts. The compiled kernel `fastbnb.so` is built for x86-64 with AVX2; elsewhere a slower reference path runs, and `build_fastbnb.sh OUT.so` with `PHYLOEXACT_FASTBNB_SO` pointing at the result restores the compiled path without replacing the distributed file.

S2.9 Keyword options

Three rules govern how the options of Table S4 interact. An option that selects an engine feature (`mode`, `prior`, `box_path`, `n_kappa`, `pinned_point`) is refused on a request for a single marginal likelihood that cannot use it, which means a misplaced option raises an error instead of being ignored. On a compound request the option applies to every component that can use it, and the notes of the other components say it was not applied. Integer-valued options (`budget`, `seed`, `cap`) accept Python and `numpy` integers and refuse floats, booleans and strings.

S2.10 Returned objects

Two conventions hold across all the objects of Table S5. Every float view of an exact quantity is the correctly rounded nearest double (`logZ`, `log_bf` on exact inputs, `ln_Zbar`) or a directed outward rounding (`interval`, posterior bounds). The exact object is always reachable (`value`, `bf_exact`, the `num/den` strings of the JSON). Every error figure is either `0.0` (exact), an interval width, a measured estimate figure, or `None` with the cause in the first note. A compound's figure is formed from its components (summed for a Bayes factor, the largest for a topology sum) and is `None` as soon as one component has none.

S2.11 Refusals

A request that cannot be met as asked raises an exception whose message names the obstacle (a theorem, a size limit, or a measured cost) and the nearest request that would succeed.

PRELIMINARY

Table S4: Keyword options of `px.evidence` and of the compound calls that forward to it. A keyword the requested computation cannot use is refused with a message naming it. On a compound request it applies to every component that can use it, and the notes of the other components say it was not applied.

keyword	values	default	meaning and scope
`register`	`"auto"`, `"R1"`, `"R2"`, `"R3"`, `"R0"`	`"auto"`	kind of answer requested; a forced kind is delivered or refused
`target`	`'exact'`; `'enclosure'` = `'rigorous'` = `'certified'`	`None`	what `"auto"` should reach for; contradictory with a forced kind → `ValueError`
`budget`	integer, 1,500 $\leq b \leq$ 10^7 (mode floors 1,100–1,500)	100,000	all likelihood evaluations of the estimate: a pilot search for the modes (about a fifth) plus the importance stage over its four replicates (`n_evals` in the certificate equals it); the stated error is indexed by it
`seed`	integer	0	seed of the estimate; replicate r of a run seeded s uses scrambling seed $4s + r$
`mode`	`'v1'`, `'v2'`, `'survey'`	`'v1'`	estimator proposal family: product-Beta, Beta-marginal copula, or the window engine (JC69 only)
`prior`	`px.BetaPrior(a,b)`	`None`	product-Beta prior on the cube coordinates; JC69 and K2P estimates at `mode='v2'` only; no error figure (experimental in 0.2.x, Section S1.7)
`minutes`, `procs`	float; int	5.0; probe's core count	wall budget and worker processes of an HKY85 / GTR+Γ_4 interval; the label reports what closed
`prec`	bits	128	ball-arithmetic precision of the JC69 / JC+R interval
`pinned_point`	`{"kappa": "1", "pi": [...]}`; `{"s": [6], "pi": [4], "alpha": ...}`	`None` (fit)	user-fixed rational point for the HKY85 / GTR+Γ_4 interval instead of the fitted one
`box_path`	`"v3"`, `"v1"`	`"v3"`	subdivision engine of the fixed-point interval; `"v1"` is the alternative without `fork` and at repeated-eigenvalue points
`n_kappa`	int ≤ 1024	16	initial number of κ boxes of the free-κ HKY85 interval
`topology_prior`	{topology: weight}	uniform	with `topology="all"` only
`subset`	ordered list of four names	`None`	quartet selected from a larger dataset; a `set` is refused
`verbose`	bool	`False`	leave the interval engines' progress on stdout
`allow_mixed_registers`	bool	`False`	`compare_models`, `pail_row`: permit kinds to differ across models, labeling the weaker
`convention`, `alpha`	`'ambient'`/`'observed'`; Dirichlet α	`'ambient'`, 1	`grade_model`: the smoothed reference declared beside the headline
`cache_dir`, `workers`	path; int	`None`	`check_adequacy`: checkpoint directory; worker processes
`label`	string	`""`	`decidability`: a window or run name carried in the certificate

PRELIMINARY

Table S5: Attributes of the returned report objects. Every report also carries `.certificate`, the JSON object of Section S4.

object	attributes
`EvidenceReport`	`value` (`Fraction` or float $\ln Z$), `register`, `logZ`, `interval`, `width`, `one_sided`, `error_nats`, `meta` (`r2_available`), `certificate`
`TopologySumReport`	`value` (exact Z_{tot} or float $\ln Z_{\text{tot}}$), `ln_Z_total`, `register`, `posterior`, `winner`, `interval`, `error_nats`, `per_topology` {key: `EvidenceReport`}, `certificate` (kind `topology-sum`)
`px.compare` dict	`verdict` (`'forced-tie'` or `'computed'`), `register`, `log_bf`, `bf_exact`, `error_nats`, `label`, `leg_registers`, `A`, `B`, `r0_certificate`; on a tie: `tie_register`, `witness`, `witness_statement`, `statement`, `topologies`, `certificate`
`ModelComparisonReport`	`bf` (`Fraction` when both exact, else `None`), `log_bf`, `register`, `error_nats`, `A`, `B` (`EvidenceReports`), `certificate` (kind `model-comparison`)
`PailReport`	`value`, `ln_Zbar`, `ln_Zbar_normalized`, `register`, `winner`, `forced_tie_all3`, `per_topology`, `error_nats`, `certificate` (kind `pail-marginal`)
`PailRowReport`	`register`, `deltas` {`'A_minus_B'`: {`ln`, `register`, `mixed`, `exact`, `error_nats`}}, `per_model` {label: `PailReport`}, `row` (flat), `certificate` (kind `pail-row`)
`grade_model`, `check_adequacy`, `decidability`, `prior_predictive`	the certificate dict itself (kinds `model-grade`, `adequacy-check`, `R0`, `prior-predictive`)
`verify_certificate` dict	`ok`, `register`, `checks` [{`check`, `status`, `kind`, `info`, `emit`}], `n_performed`, `n_skipped`, `n_skipped_informational`, `n_skipped_in_legs`, `strict`
`RouteDecision`	`register`, `engine`, `model`, `notes`

`RegisterUnavailable` carries the attributes `reason`, `unlock` and `capability` and a `to_dict()` method, so a pipeline can log the refusal as data. In practice a user meets the refusals of Table S6, which gives message excerpts from this supplement's sessions.

S3 Models, priors, errors, costs and benchmark

S3.1 What is computed

Let an alignment have usable columns $c_1, \ldots, c_N$ on n taxa, let T be a topology with edge set $E(T)$, and let M be a substitution model with free parameters θ (none for JC69). phyloexact computes the marginal likelihood

$$Z_T(M) \;=\; \int p(\theta)\,\mathrm{d}\theta \int_{[0,1]^{|E(T)|}} \prod_{i=1}^{N} P(c_i \mid T, t(x), \theta, M)\;\mathrm{d}x, \qquad x_e = e^{-4t_e/3}, \tag{S1}$$

where the inner integral runs over one coordinate $x_e \in (0,1)$ per branch. A uniform x_e is an exponential prior on the branch length t_e with rate $4/3$ (mean $3/4$ expected substitutions per site), independent across the $2n-3$ branches: five for a quartet, seven for five taxa. This is the package's default branch-length prior for every model and every kind of answer, and exact values and intervals are computed under it only. Bayes factors between topologies and between models depend on this choice, and a user who wants a different exponential rate on the estimates sets it through `px.BetaPrior` below. Table S7 lists the parameter priors $p(\theta)$; the declaration is part of the model, and every result's `prior` field prints it. Under K2P every branch carries its own pair of rate-time products $(\alpha_e t_e, \beta_e t_e)$, ten parameters on a quartet, so the ratio $\kappa_e = \alpha_e/\beta_e$ may differ between branches, unlike a single-κ parameterization. In $(\alpha t, \beta t)$ the uniform measure on

PRELIMINARY

Table S6: Refusal catalog. Messages are abridged from real sessions (. . .); each names the boundary and the remedy.

exception	raised when	message excerpt and remedy
`RegisterUnavailable`	exact K2P beyond its size limit	`K2P ... N=68 > ... cutoff 40 (... C(N_B=43, m=25) = 2.03e+16 madds) [unlock: truncate ..., model='K2P-gate', or register='R3']`; at $m = 17$: `m=17 > measured envelope 11 (... 12 is hour-scale grey)`
	no exact engine at `register="R1"` (HKY85, GTR+Γ, GM, plug-ins)	`proven-class obstruction: irrational eigenstructure ...` (Section S3.3 states the obstruction precisely); 16^K terms for GM; names the kinds available
	interval requested without `python-flint`	names the library and the install line
	GM estimate	withheld in 0.2.3 (Section S3.3)
	tree-free row mixing kinds under `"auto"`	`... K2P delivers R3, JC delivers R1 ... [unlock: allow_mixed_registers=True, or register='R3']`
	more than 32 taxa; fewer than 4; `topology="all"` on 5+ taxa	names the ceiling, `subset=`, or `px.quintet`
	tie-proof search past 400,000 nodes; orbit past 20,000; `budget` outside $[\text{floor}, 10^7]$	refused before or instead of a partial answer; the constant is named
`ValueError`	contradictory `register`/`target`; unknown `target`; option the computation cannot use	`register="R3", target="exact"` refused; neither keyword takes precedence
	Newick with branch lengths, bad leaf, unresolved tree; one tree spelled twice in `compare`; `taxa=` as a `set`	"a shape is asked for"; iteration order would change the tree named
	`px.PlugIn(name='HKY85')`;	`... a user plug-in takes its own name (e.g. 'my HKY85')`; refused at construction
	`px.BetaPrior` shape < 1 or $> 10^6$	
`px.AlphabetError`	nucleotide model on protein rows; `'JC-SR4'` on nucleotide rows	names `alphabet=` and `px.recode_sr4`
`MissingTaxaError`	`taxa=` names a sequence the file lacks	lists the names the file holds
`DatasetArgumentError`	`verify_certificate(dataset=)` lacks a certificate taxon or was read under another alphabet	the argument is refused; the certificate is not marked failed
FASTA reader `ValueError`s	bare `>`, text before the first header, duplicate name, empty file, unequal lengths	the offending entry and lengths are named
`PinIntegrityError`; `BundleLayoutError`; `NonEditableBuildRefused`	a reference file fails its hash; package directory without `pins/`; `pip install .`	re-unpack; install from the source tree with `-e`
scan row `status: refused`	a malformed window in `phyloexact.hill.scan`	`refusal.name CorruptWindow`; the run continues and exits 0

PRELIMINARY

the region of Table S7 equals 2/3 times a product of exponential densities with means 1/2 and 1/6. Hence κ has prior median 3 on each branch, the mean branch length is 5/6, and Bayes factors divide out the total mass $(2/3)^5$. The one alternative (experimental in 0.2.x, Section S1.7) is `prior=px.BetaPrior(a, b)`, a product-Beta density on the estimator's cube coordinates for the JC69 and K2P estimates at `mode='v2'`. The result then carries no error figure, because the measured classes cover the default prior only. For JC69 the cube coordinates are the x_e of Eq. S1, so `px.BetaPrior(a, 1)`, with density $a\,x^{a-1}$, is an exponential prior of rate $4a/3$ on each branch length; for example $a = 7.5$ gives rate 10. For K2P the coordinates pass through that model's own transform to the physical region, and the output prints the exact parameters used.

Table S7: Declared priors. The branch-length prior of Eq. S1 applies to every row. "Mass" is the total prior mass that Bayes factors divide out (Section S2.3).

model	parameter prior	mass
JC69, JC-SR4	none beyond the branches	1
JC+Rr	integer rate classes $1, \ldots, r$; class weights $\sim$ Dirichlet$(1, \ldots, 1)$	1
K2P (`'K2P'`, default)	per edge, uniform (Lebesgue) measure on the physical region $\{0 < x < 1,\ 0 < y < \sqrt{x}\}$ of the spectral variables $x = e^{-4\beta t}$, $y = e^{-2(\alpha+\beta)t}$; unnormalized (see text)	$(2/3)^5$, ln $=$ -2.027
K2P (`'K2P-rect'`)	Lebesgue on the unit square per edge, partly unphysical (Z may be negative); explicit opt-in	1
K3ST (`'K3ST'`)	Lebesgue on the physical image of the three rate-time variables; volume 1/4 per edge	$(1/4)^5$, ln $=$ -6.931
HKY85	$\kappa \sim$ LogNormal($\mu = 1.0$, $\sigma = 1.25$ in ln κ), the BEAST 2 default (Bouckaert et al., 2019); $\pi \sim$ Dirichlet$(1, 1, 1, 1)$; rate matrix normalized to one expected substitution per site	1
GTR+Γ_4	exchangeabilities $\sim$ Dirichlet(1^6); $\pi \sim$ Dirichlet(1^4); shape $\alpha \sim$ Exponential(mean 1); four equal-weight mean-rate categories (Yang, 1994)	1
GM (general Markov)	root π and every row of every edge matrix $\sim$ Dirichlet$(1, 1, 1, 1)$; root at the join of the topology's first pair	1
user plug-in	uniform on the plug-in's own unit cube $[0, 1]^d$; the plug-in's transform is its prior	1
`px.BetaPrior(a,b)`	product Beta(a, b) on the estimator's cube coordinates, $1 \le a, b \le 10^6$; JC69 and K2P estimates only; for JC69, Beta$(a, 1)$ on x_e is Exponential(rate $4a/3$) on t_e; no error figure	1 (K2P keeps $(2/3)^5$)
topologies	uniform 1/3 over the three quartet topologies unless `topology_prior=`	

S3.2 Models by kind of answer

Where an exact cell of Table S8 says "no", the reason is a property of the model class stated in Section S3.3, and a request for the exact value returns it. The notes below give, per model, what a user needs beyond the table.

S3.3 Per-model notes

JC69, JC-SR4 and the exact computation. Under JC69 each branch has transition probabilities $(1 + 3x_e)/4$ (no change) and $(1 - x_e)/4$ (each change) with $x_e = e^{-4t_e/3}$. The probability of one site pattern on a quartet is therefore 256^{-1} times an integer polynomial in $x_1, \ldots, x_5$ of degree

PRELIMINARY

Table S8: Models by kind of answer in 0.2.3, four taxa, with the size limits inside which each kind is delivered. N usable columns; m variable columns for the topology; N_B the base-support size of the K2P engine. "Tie classes" are the symmetry classes of Section S1.3 in which a tie proof for this model holds.

model (label)	exact value	proven interval	estimate	tie classes
JC69 (`'JC69'`)	$N \leq 150$; under `"auto"` inside the measured (N, m) limits of Table S11; forced at any m; graphics-processor path for m roughly 100 to 190	two-sided, width $\sim 10^{-35}$ log units, $N \leq 150$	yes; any $n \geq 4$; measured error for $N \leq 32$, $m \leq 9$	all three
JC69 on SR4-recoded protein (`'JC-SR4'`)	as JC69 on the recoded rows	as JC69	as JC69	all three
JC + integer rates (`'JC+R'`, `'JC+R<r>'`)	micro scale: $r \leq 2$; $N \leq 14$ at $r = 1$, $N \leq 8$ at $r = 2$	always two-sided; $N \leq 75$ ($r = 2$), 28 ($r = 3$), 15 ($r = 4$), 9 ($r = 5$)	user plug-in only	`folded`
K2P (`'K2P'`, `'K2P-rect'`, `'K2P-gate'`)	micro scale: $N \leq 35$, $m \leq 11$ (pair limits $N \leq 40$, $m \leq 11$, $N_B \leq 24$); minutes to hours near the edge	not built	yes (`'K2P'`); measured error for $N \leq 35$, $m \leq 11$	`raw`, `folded_k2p`
K3ST (`'K3ST'`, `'K3ST-rect'`)	micro scale: $N \leq 5$	not built	user plug-in only	`raw`
HKY85 (`'HKY85'`)	no: likelihood not polynomial in x_e (Section S3.3)	at the rationalized maximum-likelihood point or a `pinned_point=`; label by what closed	yes, no error figure	`raw`
HKY85, κ free (`'HKY85-free-kappa'`, `'-mlpi'`)	no	κ integrated under its prior with $\pi = 1/4$ or π at the fitted point; $\kappa + \pi$ free refused ($18^4 = 104{,}976$ parameter boxes)	yes (float comparison)	`raw`
GTR+Γ_4 (`'GTR+G4-Yang'`)	no	at the rationalized fitted point or a `pinned_point=`; the 13 free parameters refused (18^{13} boxes)	yes, no error figure	`raw`
GTR+Γ_4 at the engine's fixed point (`'GTR+G'`)	no	two-sided at 0.200 log units on tRNA-Gln (long run)	yes	`raw`
GM, general Markov (`'GM'`)	no: at least 16^K terms for K distinct patterns	no: at least 2^{63} boxes	withheld in 0.2.3	none
user `px.PlugIn`	only with a rationality argument (none included)	not built	yes, labeled without an error figure	`raw` if leaf-exchangeable

PRELIMINARY

at most one in each variable. The likelihood of N columns is the product of N such polynomials, and only the 15 pattern classes equivalent under relabeling of the four bases (`px.FOLD_LABELS`: `xxxx`, `xxxy`, …, `xyzw`) need distinct factors. phyloexact expands that product into an integer table of coefficients indexed by the exponent vector $u = (u_1, \ldots, u_5)$, $0 \le u_e \le N$, and integrates term by term under the uniform prior on each x_e, where $\int_0^1 x^u \, \mathrm{d}x = 1/(u+1)$. The sum is carried out in exact integer arithmetic, which gives Z as a ratio of two integers whose denominator divides $256^N \operatorname{lcm}(1, \ldots, N+1)^5$; the certificate states this bound as `denominator_bound` and `px.verify_certificate` recomputes it. The engine handles the $N_B = N - m$ columns that are constant within both sister pairs of the topology separately from the m columns that are not. On tRNA-Gln ($N_B = 43$, $m = 25$) the two coefficient tables hold 61,754 and 4,422,478 terms. The run time is therefore governed by m and grows roughly as $N(m+1)^5$ (Section S3.5). On a graphics processor the same sum is evaluated modulo several word-size primes and the integers are recovered by Chinese remaindering; the K2P engine uses two spectral variables per branch and plain rational arithmetic. The companion paper gives the general statement and proof for the group-based models (Schwartz et al., 2026a), and the docstring of `phyloexact/lanes/k3st.py` carries it for Kimura's three-parameter model, of which K2P and JC69 are special cases. The exact computation is four-taxon only through `px.evidence`; the JC69 estimate serves any taxon count by pruning on the unrooted tree the Newick names. `'JC-SR4'` applies the same computation to amino-acid rows after SR4 recoding (Susko and Roger, 2007); `N_used` then counts the recoded rows' usable columns, and the recoded rows' hash is carried too.

JC with integer rate classes (JC+R). A class-j site evolves with all rates multiplied by j, which maps x_e to x_e^j and keeps the likelihood polynomial; discrete-gamma rates would be irrational and are not this model. Exact values exist at micro scale under a proven bound $\operatorname{den}(Z) \mid 256^N \operatorname{lcm}(1..rN+1)^5 \, (N+r-1)!/(r-1)!$, and complete in seconds up to $N \le 14$ ($r = 1$) and $N \le 8$ ($r = 2$). No locus-scale exact computation exists at any r. The cost of the exact mixture core was measured to grow as $n^{6.7}$ (two classes) to $n^{8.2}$ (four) in the block size n, about 10^{19} multiply-adds at tRNA scale. In practice the answer is the proven interval, a tensor Gauss–Legendre quadrature with $\lceil (rN+1)/2 \rceil$ nodes per branch axis in ball arithmetic. The rule is exact at that polynomial degree, which makes the interval two-sided in every run. Relative widths are 10^{-36} to 10^{-32} on tRNA-Gln prefixes (Table S9). The rate ladder $(1, \ldots, r)$ fixes the overall scale, so a global rescaling is a different model; state this beside any mixture-versus-plain comparison.

Table S9: Proven intervals for JC+R on prefixes of the tRNA-Gln quartet: quadrature grid and relative width $(Z_{\mathrm{hi}} - Z_{\mathrm{lo}})/Z_{\mathrm{lo}}$. Every row is two-sided; the exact micro value is contained wherever it exists ($r = 2$, $N = 8$; $r = 1$, $N = 14$).

r	N	grid	relative width
2	8	9^5	7.0×10^{-36}
2	30	31^5	9.5×10^{-33}
3	16	$25^5 \times 9$	1.6×10^{-35}
4	8	$17^5 \times 6 \times 5$	4.1×10^{-36}
1	68	35^5	1.8×10^{-35}

K2P. Three labels name three integration domains: `'K2P'` (the physical region, default), `'K2P-rect'` (the unit square, an opt-in on which Z may be negative) and `'K2P-gate'` (the projection onto JC69). The exact engine works in `Fraction` arithmetic with a cost that grows

PRELIMINARY

steeply in m and N_B; a 24-column quartet at $m = 6$ is minutes to hours. Exact values are delivered up to $N = 35$ with $m \le 11$ and refused outright above $N = 40$, $m = 11$ or $N_B = 24$. Inside those limits `"auto"` runs the exact engine without the seconds budget that governs JC69; `px.compare_models(ds, T, ("K2P","JC69"))` on a short quartet can therefore run for an hour, while `register="R3"` answers at once. The estimate uses a 36-class kernel (patterns symmetric under purine/pyrimidine-preserving relabelings), is N-independent, and reports $\ln Z$ in the same Lebesgue convention as the exact engine. Because the default K2P prior is unnormalized with mass $(2/3)^5$, every Bayes factor and every tree-free difference adds $5\ln(3/2) = 2.027$ log units to the raw K2P log marginal likelihood before comparing. `compare_models` and `pail_row` do this and say so in their output.

K3ST. The three-substitution-type model (Kimura, 1981) is exact on two integration domains with denominators dividing $256^N \mathrm{lcm}(1..N+2)^{15}$. Its sufficient statistic is the 64 translation classes and no base-cone compression is implemented, so the measured limit is $N \le 5$ (support 650,240 terms at $N = 5$). It is a correctness reference and a template, and the refusal beyond $N = 5$ names the user plug-in as the alternative.

HKY85 and GTR+Γ_4. These models are not group-based. At a fixed rational point (κ, π) the HKY85 rate matrix has rational but unequal eigenvalues λ, so each transition probability mixes exponentials $e^{\lambda t}$ of three different rates. In the variable x_e those exponents are not integers, and the likelihood is therefore not the polynomial of degree at most N per branch on which the integer computation above relies. The marginal likelihood at such a point is still rational, since each factor $e^{\lambda t}$ integrates to $4/(4-3\lambda)$ under the branch prior, but 0.2.3 has no engine of that form and refuses the request. Once κ is integrated under its continuous prior, and in general for GTR+Γ_4 with its fitted exchangeabilities and Γ rates, the marginal likelihood is irrational. The proven interval fixes the substitution parameters at one exact rational point and brackets the five-dimensional branch-length integral there by rigorous subdivision. It is therefore a plug-in value at that point and not a marginal likelihood over κ, π or α; the free-κ form below integrates κ. Second-order centered forms are integrated exactly per axis, and the volume identity of the partition is confirmed in exact arithmetic when the boxes are assembled. By default the point is the maximum-likelihood fit on the package's own likelihood, rationalized to κ in hundredths and π in thousandths (GTR: exchangeabilities and α in hundredths, four rate categories). `pinned_point=` supplies the user's own rational point instead, and the field `pinned_by` says which. A fitted point is one model per topology, and `topology="all"` and the tree-free calls therefore refuse the fitted form and accept one caller-fixed point for all three topologies. With `'HKY85-free-kappa'` the interval also integrates κ under its LogNormal prior. The half-line $[0, \infty)$ is tiled into κ-boxes with rational endpoints, and $Z(\kappa)$ is enclosed uniformly on each box from the HKY85 closed form. The prior mass w of each box is bounded in ball arithmetic, and the total is $\sum w_{\mathrm{lo}} Z_{\mathrm{lo}} \le Z \le \sum w_{\mathrm{hi}} Z_{\mathrm{hi}}$ in exact rationals. Freeing π as well would need one five-cube run per (κ, π) box, $18^4 = 104{,}976$ of them at the default resolution, and is refused with that count; the 13 free parameters of GTR+Γ_4 are refused likewise. The Γ_4 discretization uses four equal-weight mean-rate categories, and its reference is `mpmath` at high precision, since published incomplete-gamma routines exist whose continued fraction is wrong for $\alpha \gtrsim 3.4$ (Section S3.8). The estimate for these models integrates all free parameters under Table S7 and carries no error figure, since no exact values exist to measure it against; the interval is their one answer with a guarantee.

PRELIMINARY

General Markov (GM). The 63-parameter general Markov model (Barry and Hartigan, 1987) has a proven rational marginal likelihood, but the exact sum has at least 16^K terms for K distinct site patterns, about 1.1×10^{12} at $K = 10$, and real loci have $K \geq 38$. A plain interval over the 63-cube cannot tighten before every axis has been halved, at least 2^{63} boxes. A particle estimator for GM exists in the code but is withheld in 0.2.3, and `px.evidence(..., 'GM', register='R3')` refuses on every install. The reasons are that a resampling estimator has no run-to-run reproduction bound that `px.verify_certificate` could hold it to, and that no set of exact values measures its error at locus scale. The generic importance-sampling estimator is disabled for GM outright, because against exact micro-scale values it read 3.0 to 5.7 log units low with healthy-looking diagnostics.

Plug-in examples. Three worked plug-ins under `examples/` run as estimates today and document the obstruction between each model and an exact or interval answer. Two are a covarion switch on JC69 (Tuffley and Steel, 1998) ($d = 7$) and a two-profile CAT-like mixture (Lartillot and Philippe, 2004) ($d = 12$). The third is a multispecies-coalescent quartet likelihood (Rannala and Yang, 2003) integrated over the 25 coalescent histories by quadrature ($d = 3$, 5,400 gene trees per parameter point). Section S3.6 walks through the first.

S3.4 The estimate and its error statement

The estimator. The estimate is multi-start mixture importance sampling on the unit cube of Eq. S1. Four random-walk pilot chains start from dispersed points. Every chain's sample and its images under the quartet's automorphisms (swapping the members of a sister pair, exchanging the two pairs) are candidate modes. These are scored, de-duplicated and turned into proposal components: product-Beta at `mode='v1'`, Beta marginals with a Gaussian copula at `'v2'`. The importance stage is a deterministic mixture over those components evaluated on scrambled Sobol points (Owen, 1997; Dick et al., 2013) in four independent replicates; the pooled mean is the point and max − min over the replicates is `replicates.spread_nats`. The symmetric images matter because a quartet whose two row pairs are near-identical, scored on the topology that separates them, has a bimodal integrand, and a single-chain estimator reads $\ln 2$ low there at every budget. A 5% uniform component guards the proposal tail for K2P.

The error figure. The figure an estimate prints comes from a measurement file distributed with the package. For that file every estimator mode at every budget level was run over ten seeds on 55 quartets with exact values, 2,680 runs in all. The quartets are saturated synthetic alignments up to $N = 32$, $m = 9$ (JC69) and $N = 35$, $m = 11$ (K2P), and a family of symmetric-row quartets scored on both the separating and the joining topology. At each (model, mode, budget) the class tolerance is three times the largest |estimate − exact| observed, rounded up to two significant digits; a level whose tolerance exceeds 0.25 log units is dropped. A result then prints `error_nats` = max(class tolerance, 3 × replicate spread), and a run whose replicates disagree by more than the class tolerance prints no figure, with cause `replicates`. Table S10 gives the measured values and Fig. S1 every run behind them. The result's first note calls the figure an indicative accuracy measured on that class and states that it is not a bound.

When no figure is printed. A tolerance transfers only to requests inside the swept class, and every estimate states the decision in `calibration.regime` (the class limits, this request's N, m, taxon count and prior, `inside`, and the reasons). Three causes give `error_nats None`, and the first note names which applies (the printed label is `UNGRADED (<cause>)`). Cause `regime` means

PRELIMINARY

Table S10: Measured error of the estimator against exact values, in log units, from the seed-sweep file distributed with the package (10 seeds per cell). "Printed error" is the class tolerance a result at that budget carries ($3 \times \max|\text{dev}|$, rounded up). A budget between levels carries the tolerance of the largest level at or below it.

model	`mode`	budget	runs	max |dev|	mean |dev|	printed error
JC69	`v1` (default)	10^4	240	0.0400	0.00368	0.13
JC69	`v1`	10^5	240	0.00454	0.00052	0.014
JC69	`v2`	10^4	240	0.180	0.00609	dropped (0.55)
JC69	`v2`	10^5	240	0.00532	0.00054	0.016
JC69	`survey`	10^4	240	0.0203	0.00310	0.061
JC69	`survey`	10^5	240	0.00325	0.00057	0.0098
K2P	`v1` (default)	10^5	310	0.0106	0.00172	0.032
K2P	`v1`	5×10^5	310	0.00522	0.00055	0.016
K2P	`v2`	10^5	310	0.0352	0.00244	0.11
K2P	`v2`	5×10^5	310	0.00607	0.00060	0.019

PRELIMINARY

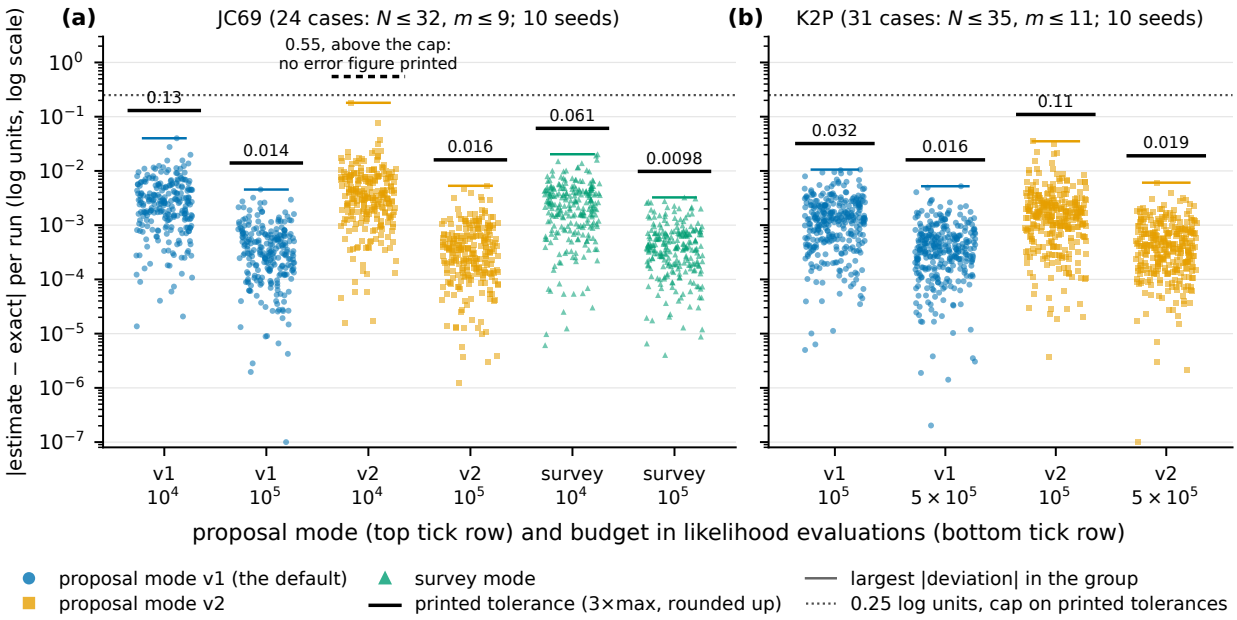

Figure S1: Absolute deviation of the 0.2.3 estimator from the exact value for all 2,680 runs of the seed-sweep file (ten seeds; log scale), by proposal mode and budget. (a) JC69 at 10^4 and 10^5 evaluations; (b) K2P at 10^5 and 5×10^5. Colored ticks: the largest deviation in each group. Black bars: the printed tolerance of Table S10 ($3\times$ that maximum, rounded up), dashed where it exceeds the 0.25-log-unit cap and no error figure is printed (JC69, `v2`, 10^4). Dotted line: 0.25 log units.

the alignment is longer or more variable than any swept case, has more than four taxa, or uses a caller's `prior=`. Cause `no_rung` means the model is not swept (HKY85, GTR+Γ_4, user plug-ins), and `replicates` is the disagreement case above. A compound that reads a Bayes factor, a posterior or a difference from such a component carries no figure either, and `strict=True` refuses it. The tRNA-Gln quartet ($N = 68$, $m = 25$) is outside the JC69 class, and its estimates in Section S1.4 therefore print their replicate spread only, as do genome windows (Section S4.5). To quote a figure on another data class, re-measure on it with `python3 scripts/r3_seed_sweep.py`; `python -m certified_evidence.gate`, run inside `pins/phylo_r3_package`, re-runs the estimator's own six-case JC69 grid against its exact values.

S3.5 Cost planning

Exact JC69 on one thread. Table S11 is the measurement behind the default routing. Each cell is a saturated synthetic quartet, the most expensive representative of its (N, m). A natural alignment of the same N and m costs about as much; tRNA-Gln at $N = 68$, $m = 25$ took 295 s on an idle host and 356 s on a fully loaded one. Under `"auto"` a request is exact when $m \le m_{\max}(N')$ for the smallest grid $N' \ge N$, where $m_{\max}$ is the largest m whose cells all completed within 480 s. Outside it `"auto"` returns the estimate with a note quoting the nearest measured cell, and the exact computation still runs when forced. tRNA-Gln at $m = 32$ exceeded ten minutes on one loaded thread here and took 1,743 s (29 min) in an earlier sweep. At $m = 33$ it took 1,576 s (26 min) in one validation run and 5,991 s (1.7 h) under shared load in another. At $N = 32$ the time grows roughly as m^5 (a factor of 1,200 from $m = 5$ to 30), more slowly at larger N (77-fold at $N = 68$), and at small m it also grows steeply with N (600-fold from $N = 32$ to 150 at $m = 5$; Table S11).

Table S11: Wall-clock seconds for one exact JC69 quartet marginal likelihood on one thread, by usable columns N and variable columns m (saturated synthetic quartets; the benchmark server of Section S3.7, one call per core, Python 3.12). The last row is the largest m delivered as exact under `register="auto"`; forced exact runs at any m. Blank cells were not measured.

	$N = 32$	$N = 48$	$N = 68$	$N = 100$	$N = 150$
$m = 5$	0.56	2.5	11.6	66.1	341
$m = 10$	1.77	5.48	14.7	80.8	403
$m = 15$	13.7	18.8	35.9	124	729
$m = 20$	74.6	86.2	137	321	
$m = 25$	253	233	383	761	
$m = 30$	670	774	896		
$m_{\max}$ under `"auto"`	25	25	25	20	10

Exact JC69 on a graphics processor. With `torch` and a CUDA device the exact value is computed modulo several word-size primes and reconstructed by Chinese remaindering, one prime at a time. On one 80-GB H100-class card the measured rate is 8.45 s per prime at $m = 89$ and 82.4 s per prime at $m = 191$; $m = 215$ ran out of memory, which brackets the single-card ceiling. The module refuses off-device with a message naming CUDA.

Estimates. In contrast, the estimate's cost is set by `budget` (pilot search included) and is independent of N and nearly of m; each call also carries about 2 s of fixed setup, so the time is not proportional to the budget. On one loaded thread of the benchmark server a tRNA-Gln estimate

PRELIMINARY

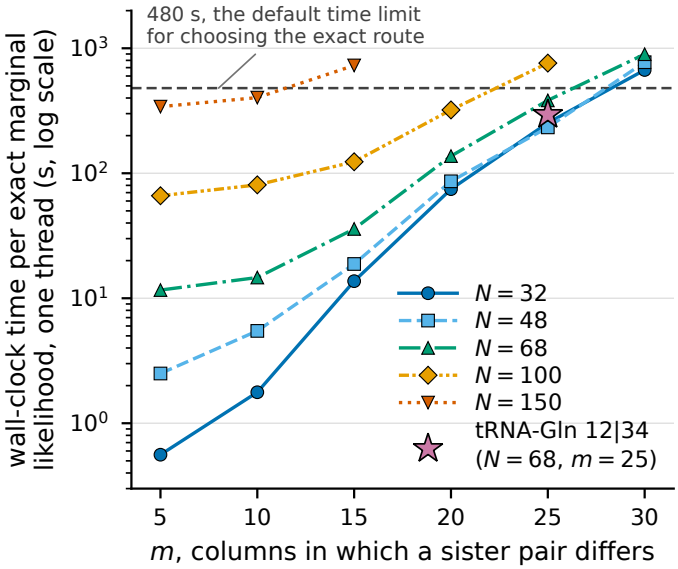

Figure S2: Wall-clock time of one exact JC69 quartet marginal likelihood on one thread against the variable-column count m, one curve per alignment length N (the data of Table S11; log scale, since the times span three decades). Dashed line: the 480 s limit that sets the default routing. Star: the tRNA-Gln `12|34` quartet ($N = 68$, $m = 25$, 295 s). At $N = 32$ the time grows roughly as m^5, more slowly at larger N.

took 3 to 6 s at budget 10^5 (JC69; 5.9 s for K2P) and 31 s at 2.5×10^6. A user plug-in on tRNA-Phe took 1.6 to 3.8 s at 2×10^4, and a three-topology posterior at 10^5 per topology 13 s. In the nine-problem benchmark of Section S3.7, run the same way on the pure-`numpy` path under full load, the mean over 36 runs was 3.5 s at 10^5 and 33 s at 2.5×10^6. An earlier release's compiled kernel on 16 threads ran the larger budget in 4.9 s on tRNA-Gln and 7.0 s on the five-taxon tRNA-Phe.

Proven intervals. The JC69 and JC+R intervals cost a fixed quadrature grid, $\lceil (N+1)/2 \rceil^5$ nodes for JC69 and $(N+1)^5$ at $r = 2$; a tRNA-length window at $r = 2$ is therefore a multi-process run. The fixed-point HKY85 and GTR+Γ_4 intervals refine adaptively until `minutes=` expires on `procs=` workers, and Table S12 shows what closes at three budgets. A two-sided interval tight enough to decide a Bayes factor of a few log units on a 68-column quartet is a long resumable run (18.8 million leaf boxes for width 0.200 log units). Bayes factors smaller than about 10^{-3} log units call for the exact value.

Memory and other costs. The exact fit ceiling at alignment scale costs little, 2,000 taxa by 10^5 sites taking 2.3 s and 0.8 GB (Section S4.7). A tie proof on an ordinary alignment is milliseconds at every $n \leq 32$ (0.09 s on tRNA-Phe); rechecking one with `px.verify_certificate` for a highly symmetric taxon set can take minutes at $n \geq 15$. We did not measure the memory of the exact CPU computation or of the estimate; in the timing sweep of Section S3.7 every program, the estimate included, ran inside a 16 GB address-space limit. The benchmark server has 732 GB, and the single-card graphics-processor computation ran out of 80 GB at $m = 215$. The window scan's throughput is in Section S4.5.

PRELIMINARY

Table S12: Fixed-point proven intervals against budget. Rows 1 to 8: a 12-column quartet fixture; rows 9 and 10: the tRNA-Gln quartet ($N = 68$). "JC limit" is the HKY85 form at $\kappa = 1$, $\pi = 1/4$, whose interval must and does contain the exact JC69 value. Width in log units of $\ln Z$; "upper bound" means the lower endpoint did not close (label `R2-one-sided`).

	model at fixed point	budget	leaf boxes	result
1	JC limit	1 min × 8 procs	211,996	two-sided, width 4.46; contains exact
2	HKY85, fitted	1 min × 8 procs	223,666	upper bound only
3	GTR+Γ_4, fitted	1 min × 8 procs	173,548	upper bound only
4	JC limit	5 min × 32 procs	1,199,030	two-sided, width 0.521; contains exact
5	HKY85, fitted	5 min × 32 procs	1,208,068	two-sided, width 1.99
6	GTR+Γ_4, fitted	5 min × 32 procs	1,130,130	upper bound only
7	GTR+Γ_4, engine point	5 min × 32 procs	1,069,589	upper bound only
8	GTR+Γ_4, engine point	long resumable run	14,641,639	two-sided, $[-47.685, -47.537]$, width 0.147
9	HKY85, fitted ($\kappa = 413/100$), tRNA-Gln	3 min × 1 proc		upper bound $\ln Z \leq -230.7919$
10	GTR+Γ_4, engine point, tRNA-Gln	long resumable run	18,766,693	two-sided, $[-255.837, -255.636]$, width 0.200

S3.6 Adding a model

The contract. A model enters phyloexact by supplying five things (user models are experimental in 0.2.x, Section S1.7). (i) A vectorized log-likelihood on a unit cube whose coordinates encode its parameters and their prior transform, with its pattern compression declared. (ii) A declaration of which kinds of answer it can receive; an exact value needs a rationality argument and an interval needs a rigor note for its bound chain. (iii) Comparisons against an independent implementation and against the special cases it reduces to. (iv) Its sufficient statistics for posterior-predictive tests, and (v) the primary citation and the parameter conventions. Points (ii) to (v) are how the built-in models were admitted; a user's plug-in needs only point (i) to run as an estimate today.

The call.

```
model = px.PlugIn(loglik_fn, d, name="JC69+I-toy")   # loglik_fn(U: array (n, d)) -> array (n,) of
    log-likelihoods
rep   = px.evidence(ds, "12|34", model, register="R3", budget=20000, seed=0)
```

`loglik_fn` receives n points of $[0,1]^d$ and returns n log-likelihoods of the whole alignment. The map from cube coordinates to branch lengths and parameters is the plug-in's own and defines its prior, since the uniform measure on the cube pushes forward to whatever the transform declares. Using $x_e = e^{-4t_e/3}$ for the first five coordinates gives the package's standard branch prior. The output names the model as `plugin:<name>`; a name that is a built-in label (`'HKY85'`), begins with one, or is blank is refused at construction.

What comes back. An estimate with its four replicates and their spread; `error_nats` is `None` with cause `no_rung`, because no set of exact values measures a user model's error. A worked toy in 25 lines of `numpy`, JC69 plus an invariant-sites fraction $p_{\text{inv}} \sim U(0,1)$ ($d = 6$) on the tRNA-Phe quartet, returned $\ln Z = -158.286$ (spread 0.031) in 1.6 s at budget 2×10^4. The same function with p_{inv} fixed at 0 is JC69, and its estimate differed from the exact JC69 value by -0.00265 log units (spread 0.0037), which is the control every plug-in author should run. `px.verify_certificate` tests the label, the structure and the replicate block of a plug-in result but cannot reproduce the value, since no code in the package computes the user's likelihood. It returns `ok=True` with

PRELIMINARY

that named skip, and `strict=True` refuses it. Tie proofs apply to a plug-in whose leaves are exchangeable (`px.compare` tries the `raw` class first), and the ceiling-gap meter and `compare_models` accept plug-in marginal likelihoods with the weaker-kind rule.

The covarion example. `examples/covarion_plugin.py` implements a two-state switch on JC69 in which each site is ON (evolving under JC69) or OFF (invariant), switching along every branch at rates $\nu\pi_{\text{on}}$ and $\nu(1-\pi_{\text{on}})$. Its cube has $d = 7$ coordinates, five branches under the standard transform, $\nu = -\ln(1-u_6)$ (Exponential, mean 1), and $\pi_{\text{on}} = u_7$ (uniform), and its 8-state generator splits into four closed-form 2×2 blocks. The file states two identities that the validation suite tests. Setting $\pi_{\text{on}} = 1$ gives JC69 pointwise, and the estimate then matches the exact JC69 value within the measured tolerance; setting $\nu = 0$ gives JC69 with invariant sites. It also explains why no exact value exists, namely that the hidden switch destroys the base symmetry the exact computation folds on and the block eigenvalues are irrational in (ν, π_{on}). `python3 examples/covarion_plugin.py` runs it on tRNA-Gln and returns $\ln Z = -259.528$ at budget 2×10^4 in 6.9 s. The CAT-like and coalescent examples follow the same pattern and state their own obstructions.

Requirements for a stronger kind. An exact value for a new model needs a proof that Eq. S1 is rational under the declared prior in a form an engine can sum, and an engine that returns the integer pair with an a-priori denominator bound that `px.verify_certificate` can recompute. The K3ST module (`phyloexact/lanes/k3st.py`, whose docstring is the proof) is the template. An interval needs a bound chain written down beside the code, as `RIGOR_JCR_R2.md` and `RIGOR_FREE_KAPPA.md` are for the two newest interval engines. An error figure for a plug-in's estimate needs exact values of that model on a swept class; without them the honest label is the one the package prints.

S3.7 Benchmark design and full tables

Problems. The benchmark of Section 5 of the main text has nine fixed-topology problems on three alignments of two mitochondrial tRNA loci (Table S13). The tRNA-Gln quartet (mouse, rat, chicken, *Xenopus laevis*; 68 gapless columns from the Rfam RF00005 seed alignment) is scored on its three topologies. The hominid tRNA-Phe quartet (human, chimpanzee, gorilla, orangutan; 69 columns) is scored on its three, and the tRNA-Phe quintet adding gibbon (67 columns) on three of its fifteen. Every engine targets the same integral: JC69 with independent exponential (rate 4/3) priors on the five or seven branch lengths of the fixed topology. The exact reference values are distributed with their datasets under `pins/parity/` and `pins/certified_table/`, the six quartet values as integer pairs and the three quintet values as 60-digit logarithms (Table S13). The six quartet values are recomputed by the exact engine of 0.2.3 in the validation suite. The three quintet values were computed by the exact five-taxon integration of the companion paper (Schwartz et al., 2026a); 0.2.3 itself refuses exact five-taxon values (Section S1.3). Two pairs of problems are exact ties by symmetry (tRNA-Phe `12|34=14|23`; quintet $((1,2),3,(4,5)) = ((2,3),1,(4,5))$).

Estimators and settings. The phyloexact 0.2.3 estimator ran on all nine problems at budgets 10^5 and 2.5×10^6 with seeds 1 to 4, 36 runs per budget. Each run was one call `px.evidence(ds, T, "JC69", register="R3", budget=B, seed=s, procs=1)` in one process on one thread at the default `mode`. The release-0.1.0 estimator and its ablations ran the same seeds and budgets in the original campaign, and Table S18 keeps their rows for comparison. The ablations are plain Monte Carlo with the same proposal, prior sampling, a textbook Laplace approximation, and nested sampling by `dynesty` (Speagle, 2020). The nineteen estimators of Fourment

PRELIMINARY

Table S13: The nine benchmark problems and their exact reference values (JC69, default prior). Topology keys follow Section S1.2 in the taxon order given; the quintet key *ab-cd-e* is the unrooted tree $((a, b), e, (c, d))$. $\ln Z$ is the nearest double of the exact value; the repository holds the six quartet values as integer pairs and the three quintet values as 60-digit logarithms.

alignment	taxa (order)	topology	N	m	exact $\ln Z$
tRNA-Gln	mouse, rat, chicken, *Xenopus*	`12|34`	68	25	−261.4030641503
		`13|24`	68	32	−275.9398267
		`14|23`	68	33	−276.9593357
tRNA-Phe	human, chimpanzee, gorilla, orangutan	`12|34`	69	10	−161.9379803099
		`13|24`	69	10	−161.9366608916
		`14|23`	69	10	−161.9379803099 (tie with `12|34`)
tRNA-Phe + gibbon	human, chimpanzee, gorilla, orangutan, gibbon	12-45-3	67		−189.6451165484
		23-45-1	67		−189.6451165484 (tie with 12-45-3)
		13-45-2	67		−190.2076498014

et al. (2020) ran in their own program, physher at tag `marginal-v1.1`, at that paper's default settings, 36 runs per method (nine problems, four seeds). MrBayes 3.2 stepping-stone (Ronquist et al., 2012; Xie et al., 2011), RevBayes stepping-stone (Höhna et al., 2016) and LoRaD (Wang et al., 2023) ran on the tRNA-Gln and quintet problems at two budget levels matched to the in-house estimator by proposal count; floating-point operation counts were not matched. The lower level is 2.55 and 2.53 million proposals for MrBayes and RevBayes against 2.5×10^6 likelihood evaluations, with 12 runs per method and level (three topologies, four seeds). On the quintet the RevBayes accounting was recounted for seven branches (2,550,100 and 20,400,100 proposals at the two levels, 1.0000× the MrBayes levels; LoRaD at 0.78×). MrBayes used `prset brlenspr=unconstrained:exponential(1.3333333) statefreqpr=fixed(equal) topologypr=fixed`. MrBayes also used `lset nst=1 rates=equal` and `mcmcp nruns=2 nchains=1 samplefreq=500 diagnfreq=50000`, with the stepping-stone power schedule left at the program default. One caveat on the quintet MrBayes cells of 12-45-3 is given in Table S16, note a. Each RevBayes script drew every branch length from `dnExponential(4.0/3.0)`, moved it with `mvScale(bl[i], lambda=1.0, weight=1.0)`, fixed the topology through `treeAssembly` under `fnJC(4)`, and read the estimate from `steppingStoneSampler` on the power-posterior file. LoRaD read fixed-topology posterior samples drawn by MrBayes with the same `lset` and `prset` lines and `savebrlens=yes`, and was called as `lorad_estimate(params, colspec, 0.5, "left", 0.5)`: training on the first half of the kept sample, coverage 0.5. That coverage was chosen on the lower-level tRNA-Gln chains, among coverages 0.1 to 0.9 and two training modes, as the one with the smallest across-chain standard deviation, and then held fixed. LoRaD itself reports no Monte Carlo standard error. The sampled MrBayes log-likelihoods that LoRaD reads were checked against an independent pruning computation on every chain. The physher runs used the experiment templates of Fourment et al. (2020) (repository `marginal-experiments`) with the branch-length prior rate changed from 10 to 4/3 and no other change to operators, budgets, step schedules, burn-ins or optimizer tolerances. Table S14 lists the versions, replicates and seeds of every program, and Table S15 the sampler settings of the comparison programs. Table S16 gives the three comparison programs' errors on the quintet topologies at both levels.

PRELIMINARY

Table S14: Programs and versions of the benchmark as recorded in the run files, with the number of estimates per problem and budget level and the random seeds. The input files and driver scripts are deposited with the release archive.

program	version	estimates per problem and level	seeds
phyloexact (this paper)	0.2.3 under Python 3.12.3; one process, one thread, no compiled kernel	4 at each of the budgets 10^5 and 2.5×10^6 (36 runs per budget)	`seed=` 1 to 4
phyloexact (original campaign)	0.1.0	4 at each of the same two budgets (36 runs per budget)	seeds 1 to 4; timing sweep seeds 1 to 4
MrBayes	3.2.7a, 64-bit Linux build (Ubuntu package `3.2.7a-7build2`)	4: two invocations with two internal runs each (`nruns=2`)[a]	one invocation with the automatic clock seed, one with `set seed=8675309 swapseed=42`[b]; timing sweep `seed`=`swapseed`=1 to 4
RevBayes	1.4.0, Linux x86-64 release binary	4: four scripts, one estimate each	`seed(1001)`, `2002`, `3003`, `4004` in every cell except the tRNA-Gln upper level, `5005` to `8008`[b]; timing sweep `1001` to `4004`
LoRaD	R package `lorad` 0.0.1.0 (CRAN) under R 4.3.3; posterior samples from MrBayes 3.2.7a	4: two MrBayes invocations with two chains each, one estimate per chain[a]	MrBayes seeds as in the MrBayes row[b]; timing sweep seeds 1 to 4 with `swapseed` 101 to 104
physher	tag `marginal-v1.1` (commit `7782970`), built with gcc 13.3.0 and GSL 2.7.1; templates from `marginal-experiments` commit `41124a1`	4 per method at one level (36 runs per method)	1001, 2002, 3003, 4004, written into each configuration as `"init": {"seed":s}`

[a]One MrBayes or LoRaD invocation thus yields two estimates, and one RevBayes or phyloexact run yields one (the "estimates per invocation" column of Table S17). [b]Because the second MrBayes invocation reuses `seed=8675309 swapseed=42` in every cell and RevBayes reuses its four seeds, estimates are seed-matched across topologies within a level. On the quintet topology 12-45-3 the automatic-seed MrBayes invocations of the two levels also drew the same seed (Table S16, note).

PRELIMINARY

Table S15: Sampler settings of the comparison programs at the two budget levels ("lower" matched to 2.5×10^6 likelihood evaluations by proposal count, "upper" eight times larger). MrBayes makes one proposal per generation and RevBayes one single-branch proposal per branch per generation (five on a quartet, seven on the quintet). The cost of a LoRaD estimate is the MrBayes chain it reads. The physher methods ran at one level, the defaults of Fourment et al. (2020).

program, estimator	steps and power schedule	sampling per step, lower / upper level	burn-in	proposals or steps per estimate, lower / upper
MrBayes, stepping-stone (`ss`)	`nsteps=50`; schedule parameter at the program default ($\alpha = 0.4$, not set in the input); steps run from the posterior toward the prior	`ss ngen=2550000`: 50,000 generations per step, sampled every 500 (100 samples) / `ss ngen=20400000`: 400,000 generations per step (800 samples)[a]	initial burn-in of one step length (50,000 / 400,000 generations); the first quarter of every step (12,500 / 100,000 generations) discarded	2,550,000 / 20,400,000 proposals
RevBayes, stepping-stone on power posteriors	`powerPosterior(..., cats=50, alpha=0.4, sampleFreq=10)`	`run(generations=10000)` per stone on the quartets and `7186` on the quintet / `80000` and `58186`; sampled every 10	`burnin( generations=5000, tuningInterval=200)`	quartets 2,525,000 / 20,025,000; quintet 2,550,100 / 20,400,100
LoRaD on MrBayes posterior samples	none (one posterior chain per estimate); training fraction 0.5, coverage 0.5[a]	`mcmcp ngen=2000000 samplefreq=100` / `ngen=16000000 samplefreq=800`: 20,001 samples per chain at either level	first quarter of each chain discarded (15,001 samples kept)	2,000,000 / 16,000,000 generations (0.78× the MrBayes levels)
physher, generalized stepping-stone	50 steps; reference distribution a per-branch gamma fitted to a preliminary 10^6-step posterior chain	10^6 steps per step, logged every 1,000 (1,001 rows)	first 100 logged rows of the preliminary chain and of every step discarded (901 samples per step)	5.1×10^7 steps in all
physher, stepping-stone, path sampling, modified path sampling	50 steps at Beta$(0.3, 1)$ quantiles	10^6 steps per step, logged every 1,000	first 100 logged rows of every step discarded (901 samples per step)	5.0×10^7 steps
physher, bridge sampling and the harmonic-mean family	none	one 10^6-step posterior chain, logged every 1,000	first 100 logged rows discarded (901 samples)	1.0×10^6 steps
physher, the other methods	template defaults of Fourment et al. (2020) (nested sampling: 50 live points; the importance-sampling and naive Monte Carlo methods: 10,000 draws); evaluations as in Table S18			

[a] The quintet runs of MrBayes, RevBayes and LoRaD (Table S16) used the step counts, chain lengths, sampling frequencies and seed scheme given here and in Table S14, and the same LoRaD call. The LoRaD log-likelihood check there required a largest discrepancy below 10^{-3} and a mean below 3×10^{-4} log units.

PRELIMINARY

Table S16: MrBayes stepping-stone, RevBayes stepping-stone and LoRaD on the three tRNA-Phe quintet topologies at both budget levels (settings in Tables S14 and S15). Cells give the mean signed error (estimate minus exact $\ln Z$, log units) and the standard deviation over the $n = 4$ estimates. The row "upper, mean/s.d." divides the MrBayes mean error at the upper level by its standard deviation. Topologies 12-45-3 and 23-45-1 tie exactly (Table S13), so agreement between those two columns is a consistency check and not independent evidence.

		12-45-3		13-45-2		23-45-1	
program	level	mean	s.d.	mean	s.d.	mean	s.d.
MrBayes 3.2 stepping-stone	lower	−0.0325	0.0343	−0.0149	0.0293	−0.0293	0.0360
	upper	−0.0302	0.0090	−0.0250	0.0079	−0.0369	0.0097
	upper, mean/s.d.[a]	−3.3		−3.2		−3.8	
RevBayes stepping-stone	lower	+0.0151	0.0196	+0.0031	0.0824	+0.0174	0.0277
	upper	+0.0074	0.0137	−0.0151	0.0156	−0.0226	0.0046
LoRaD	lower	+0.0059	0.0163	−0.0053	0.0085	−0.0062	0.0084
	upper	−0.0057	0.0075	+0.0052	0.0079	+0.0162	0.0186

[a] On 12-45-3 the automatic-seed MrBayes invocations at the lower and the upper level were launched within the same second and drew the same seed (`Seed=4471740`). The automatic seeds of the other two topologies all differ. For that topology the comparison between levels is therefore not between independent samples, while within each cell the two invocations, and hence the four estimates, remain independent. The LoRaD upper-level cell on 23-45-1 is pulled by one chain at −189.6103 against −189.617 to −189.650 for the other three.

Machine and timing. All runs, in both campaigns, used one Linux server with 96 logical cores (Intel Xeon Platinum 8581C at 2.30 GHz) and 732 GB of memory, the benchmark server of the main text. The timing sweep of Table S17 belongs to the original campaign; it held every program to one thread (`OMP`, `OPENBLAS`, `MKL`, `BEAGLE` and `NUMEXPR` thread counts set to 1; 99% CPU per run confirmed). It ran at `nice 10` with a 16-GB address-space limit and was timed with `/usr/bin/time -v`, four runs per cell. The server carried other work during that sweep (load average 32 to 95 on its 96 cores), so those medians are not idle-machine times. Its phyloexact rows are the release-0.1.0 pure-Python estimator on one thread; that release's compiled kernel on 16 threads was timed separately on the same machine. The 0.2.3 re-run ran at `nice 10` on the fully loaded server on the pure-`numpy` path; its walls, in the last row of Table S17, are loaded-machine figures and not a timing measurement. At matched wall-clock the earlier release's compiled single-core kernel evaluates 4.21 million quartet likelihoods per second against about 1.12 million for physher's C likelihood on the same alignment. That ratio carries two caveats. First, physher exposes no evaluation counter, and its rate is therefore reconstructed from run times (two reconstructions agree to about 1%). Second, the integrands differ, a generic site-pattern likelihood with proposal overhead against the 15-class folded JC69 integrand.

The full accuracy table. Table S18 extends Table 3 of the main text to every estimator family measured, sorted by mean absolute deviation from the exact value. Its unnumbered first row is this release's estimator (0.2.3) at the same seeds and budget as row 1, the release-0.1.0 estimator; rows 1 to 21 are the original campaign's table.

S3.8 K2P, incomplete-gamma and version notes

K2P estimates against exact values. The full multi-program comparison exists for JC69 only. However, the earlier release's K2P estimator was measured against 19 exact K2P values ($N \leq 32$ and one $N = 34$ case), with a pooled mean deviation of 1.5×10^{-4} log units at 2.5×10^{6} evaluations

PRELIMINARY

Table S17: Wall-clock seconds per invocation at the 2.5×10^6 budget level on the benchmark server. Rows 1 to 5: the original timing sweep (median and range over four runs, host under partial load, every program on one thread except the 16-thread row). Last row: the 0.2.3 re-run of this paper (mean over four seeds per topology, one thread, pure-`numpy` path, host under full load), given for orientation and not comparable as a timing.

program	tRNA-Gln quartet	tRNA-Phe quintet	estimates per invocation
phyloexact 0.1.0, compiled, 16 threads, 2.5×10^6	4.9	7.0	1
phyloexact 0.1.0, Python, 1 thread, 2.5×10^6	25.95 (22.16 to 36.58)	31.38 (29.77 to 36.74)	1
MrBayes 3.2 stepping-stone	7.21 (6.31 to 7.55)	3.86 (3.80 to 3.88)	2
RevBayes stepping-stone	13.03 (12.78 to 13.49)	15.23 (11.12 to 18.10)	1
LoRaD	6.81 (5.94 to 7.31)	6.18 (4.44 to 6.64)	2
phyloexact 0.2.3, `numpy`, 1 thread, loaded host, 2.5×10^6	30.7 to 36.2	38.9 to 40.4	1

(worst 6.8×10^{-4}). At matched evaluations physher's generalized stepping-stone at its own settings deviated by 2×10^{-3} to 7×10^{-3} log units on the same values, at about 0.6× the wall-clock. On tRNA-Gln under K2P no exact value exists at $N = 68$. There physher stepping-stone (50 steps, about 5×10^7 evaluations, four seeds) and the estimate agreed within 0.017 log units, with seed spreads of 0.030 and 0.0002 respectively. The 0.2.3 measurement for K2P is Table S10 (21 synthetic cases to $N \leq 35$, $m \leq 11$). In addition, the eighteen-locus K2P-against-JC69 column of Section 6.2 of the main text uses the 0.2.3 K2P estimator at budget 5×10^5, seeds 1 to 4, against exact JC69 values. Each locus and seed took 56 s on one loaded thread. Against the earlier release's column the largest per-locus change is 0.005 log units, the corpus sum is -100.128 ± 0.0024 log units (earlier -100.12 ± 0.0023), and no locus changes its preferred model (JC69 at 16 of 18).

The incomplete-gamma note. We assembled fixed-parameter reference likelihoods for HKY85 and GTR+Γ_4 against another program at 30 points each, requiring agreement within 2×10^{-6} log-likelihood units. In doing so we found that physher 1.9 returns wrong log-likelihoods under $+\Gamma$ for shape $\alpha \gtrsim 3.4$. The continued-fraction loop of its incomplete-gamma routine has an off-by-one defect that shifts the third and fourth category rates by about 0.2. Reference points from that program were therefore restricted to $\alpha \leq 3$, and phyloexact arbitrates its own discretization against `mpmath` at high precision. JC69 results, including every benchmark number above, use no Γ path and are unaffected; we have not assessed whether the defect affects the published values of the 2020 study.

Versions behind the numbers. The exact reference values are version-independent integers. The phyloexact accuracy rows of Table 3 in the main text and the first row of Table S18 are the 0.2.3 estimator of Section S3.4, re-run for this paper on all nine problems. Pooled over the 36 runs at 2.5×10^6 evaluations its mean absolute deviation from the exact values is 9.8×10^{-5} log units (median 1.4×10^{-5}; worst 1.41×10^{-3}, on tRNA-Gln `13|24`). At the default 10^5 it is 6.5×10^{-4} log units (worst 8.53×10^{-3}). By family, the tRNA-Phe quartets and quintets average below 2×10^{-5} log units at the larger budget and the three tRNA-Gln topologies ($m = 25$ to 33) 2.8×10^{-4}. None of these runs prints an error figure, because all nine problems lie outside the swept class ($N = 67$ to 69; Section S3.4). Each result quotes its four-replicate spread instead, 1.6×10^{-5} to 1.5×10^{-3}

PRELIMINARY

Table S18: Every estimator against the exact values: mean |estimate − exact| and its standard deviation in log units, pooled over nine problems and four seeds (36 runs) unless marked tRNA-Gln (12 runs). Evaluations per run are approximate for the comparison programs. physher rows are the methods of Fourment et al. (2020) at that paper's defaults. The unnumbered first row is the phyloexact 0.2.3 estimator re-run for this paper, with its largest single-run deviation in place of a standard deviation. Rows 1 to 21 are the original campaign; rows marked 0.1.0 are the earlier release's estimator and its ablations.

	method (program)	mean |dev|	sd	evaluations
	multi-start mixture importance sampling (phyloexact 0.2.3, this release)	0.00010	(max 0.00141)	2.5×10^6
1	randomized quasi-Monte Carlo importance sampling (phyloexact 0.1.0)	0.00006	0.0001	2.5×10^6
2	plain Monte Carlo, same proposal (0.1.0 ablation)	0.00064	0.0009	2.5×10^6
3	generalized stepping-stone (physher)	0.0026	0.0040	5.1×10^7
4	LoRaD (tRNA-Gln and tRNA-Phe)	∼0.005	0.007	2.0×10^6
5	bridge sampling (physher)	0.018	0.021	1.0×10^6
6	stepping-stone (RevBayes; tRNA-Gln)	∼0.017	0.02 to 0.04	2.5×10^6
7	stepping-stone (MrBayes; tRNA-Gln)	∼0.024	0.02	2.6×10^6
8	modified path / stepping-stone / path sampling (physher)	0.027 to 0.030	0.033	5.0×10^7
9	GLIS (physher; heavy tail, max 0.31)	0.034	0.077	1.1×10^4
10	variational Bayes importance sampling (physher)	0.107	0.263	1.2×10^4
11	nested sampling (0.1.0 ablation `dynesty` / physher)	0.14 / 0.29	0.36	2×10^5 / 1.2×10^6
12	Laplace approximations GL / BL (physher)	0.19 / 0.30	0.26 / 0.45	7×10^2
13	prior sampling, quasi-random / pseudo-random (0.1.0 ablation)	0.39 to 0.53		2.5×10^6
14	Laplace with importance sampling / Laplace (0.1.0 ablation, textbook implementation)	0.98 / 1.51		$\sim 2.5 \times 10^6$
15	Laplace approximation LL (physher)	1.01	0.62	7×10^2
16	variational lower bound, ELBO (physher; below the exact value in 36 of 36 runs)	1.24	0.45	2.2×10^3
17	naive Monte Carlo (physher; worst tail −19.0)	4.2	5.7	10^4
18	harmonic mean (Newton and Raftery, 1994) (physher / 0.1.0 ablation)	∼7.5	3.0	10^6
19	stabilized harmonic mean / CPO (physher)	8.8 / 8.9	3.0 / 3.4	10^6
20	posterior predictive density (physher)	13.2	3.1	10^6
21	maximum-likelihood / maximum-a-posteriori plug-ins (physher)	14.5 / 15.6	3.8 / 4.4	5×10^2

PRELIMINARY

log units at the larger budget. The two tied pairs return bit-identical estimates under 0.2.3, the estimator being a function of the folded site-pattern data and the seed, so the nine problems are seven distinct computations. The release-0.1.0 estimator (one pilot chain, one fitted proposal) reached 5.6×10^{-5} and 5.1×10^{-4} log units at the same two budgets in the original campaign. Its row, its ablations, the comparison programs' rows and the timing sweep of Table S17 are kept from that campaign; the 0.2.3 estimator's measured error on its own swept class is Table S10. No general-Markov estimate is printed anywhere, because 0.2.3 withholds that estimator.

S3.9 Source of the MrBayes stepping-stone offset

The MrBayes 3.2.7a stepping-stone estimates of Tables S16 and S18 lie 0.016 to 0.03 log units below the exact values on every topology and alignment tested; standard errors below are over independent estimates. The offset is unchanged, within its standard error, by the number of stones (50, 100, 200), the chain length per stone (one to eight times the settings of Table S15), doubled initial and within-stone burn-in, the direction of the path, and double- in place of single-precision likelihoods. It does not shrink with alignment length: on the three tRNA-Gln topologies (68 sites) it is -0.0164, -0.0170 and -0.0129 log units (standard errors 0.0032 to 0.0035), equal within error, and on a 390-site JC69 alignment it is -0.036 (0.004). At the branch-length vectors that MrBayes samples, its printed log-likelihood agrees with an independent pruning computation to 10^{-5} log units and its log prior with the stated exponential density to 10^{-9}, and a run with a constant likelihood returns a log marginal likelihood of exactly zero, so the likelihood, the prior bookkeeping and the stepping-stone arithmetic are not the source. Compared stone by stone with per-stone ratios computed by quadrature, the MrBayes contributions fall short by 3 to 10×10^{-4} log units at every stone with β above 0.01 and by almost nothing below it, as expected if the chain samples slightly longer trees at every power of the likelihood.

On a fixed topology MrBayes 3.2 updates branch lengths with three proposals, `Multiplier(V)` (66.7 per cent of proposals), `Nodeslider(V)` (23.3 per cent) and `TLMultiplier(V)` (10 per cent). With `propset Nodeslider(V)$prob=0`, issued after any `mcmcp` command that changes `nruns`, the offset disappears. Over eight of the exact problems, with 72 estimates per setting, the mean deviation from the exact values is -0.0260 log units (standard error 0.0027) with the default proposals and -0.0026 (0.0019) without the node slider, with the same sign in every problem; an earlier set of 70 estimates per setting gave -0.0268 (0.0014) and -0.0002 (0.0013). Because the offset is equal across topologies of one alignment, it largely cancels in Bayes factors between them. RevBayes, whose branch-length proposals are per-branch scale moves, converges on the exact values (Table S16).

The node slider is `Move_NodeSlider` in `src/proposal.c` of the MrBayes source (lines 7982 and 7989 at tags v3.2.7 and v3.2.7a; lines 8420 and 8427 at tag v3.2.8 and in the development branch as of 18 September 2026). It multiplies the sum M of two adjacent branch lengths by $c = e^{\lambda(u-1/2)}$, redraws one of the two lengths uniformly on the admissible window of width W, sets the other to the remainder, and assigns the log proposal ratio $\ln(W'/W) + 2\ln(M'/M)$. Because the second length is redrawn rather than rescaled, the map from (M', x'_q) to the new pair has unit Jacobian and the Hastings ratio is $M'W'/(MW)$, that is $\ln(W'/W) + \ln(M'/M)$, which is the expression in MrBayes versions 3.1.2 to 3.2.3; the second factor of M'/M entered with release 3.2.4. Used alone, the kernel as coded satisfies detailed balance with respect to the target density multiplied by M. A direct re-implementation of the routine, applied to two independent exponential branch lengths of rate 4/3, samples their sum from a Gamma(3) distribution instead of Gamma(2) (mean 2.247, standard error 0.005, against 2.25; Kolmogorov–Smirnov distance to Gamma(3) below 0.003) and returns Gamma(2) when the factor 2 is replaced by 1 (mean 1.499, standard error 0.004,

PRELIMINARY

against 1.5). MrBayes itself shows the same behaviour when run without data. With four taxa, the topology fixed, `prset brlenspr=unconstrained:exponential(1.0)`, the other two branch-length proposals disabled and `mcmc data=no`, version 3.2.7a samples a mean tree length of 7.03 (standard error 0.01; eight chains of 8×10^6 generations) against the prior's 5, every branch 1.40 to 1.41 instead of 1. At rate 10 the sampled mean is 0.703 (0.001) against 0.5, and under the default `unconstrained:gammadir(1.0,0.1,1.0,1.0)` it is 30.3 (0.1) against 10. With the node slider disabled the prior is recovered (ratio of sampled to prior mean 1.0004 to 1.0005), and in the default mixture of the three proposals the sampled prior tree length is 0.2 to 0.5 per cent too long. On our alignments (4 to 5 taxa, 67 to 390 sites) the posterior mean tree length with the move on exceeds that with it off by 0.05 to 0.11 per cent, about 0.005 posterior standard deviations. The effect is far below any model-choice threshold and does not bear on trees or branch lengths inferred with MrBayes in ordinary use.

S3.10 Data behind the example analyses

The tRNA quartets. The tRNA-Gln quartet is distributed as `pins/phylo_gkz/realdata/trnaq_quartet.fasta` with entries `V00711.1/3842-3772` (mouse), `X14848.1/3820-3750` (rat), `X52392.1/5172-5102` (chicken) and `M10217.1/5909-5841` (*Xenopus laevis*), 68 columns, none dropped. The hominid tRNA-Phe quartet is distributed as `pins/parity/datasets/hominid-mt-trnaphe.fasta` (73 columns, 4 gapped, $N = 69$) and the quintet with gibbon as `hominoid5-mt-trnaphe.fasta`, with RefSeq accessions in their provenance sidecars.

The eighteen hominid tRNA loci. The loci were excised from the mitochondrial reference genomes of human (NC_012920), chimpanzee (NC_001643), gorilla (NC_001645) and orangutan (NC_001646) at their annotated tRNA coordinates. They were aligned to the human sequence by Needleman–Wunsch star alignment (match 1, mismatch −1, gap −2), giving aligned lengths of 66 to 75 columns. The panel is the sixteen loci informative for the human–chimpanzee–gorilla trichotomy plus tRNA-Ala and tRNA-Ser(UCN), fixed before any marginal likelihood was computed; tRNA-Phe, -Lys, -Ser(AGY) and -Thr were excluded by that prior choice. These are the loci of Section 6.2 of the main text, all eighteen distributed under `pins/pail_demo/`; the six scan windows of Section S4.5 are five of them plus the tRNA-Phe quartet from `pins/parity/datasets/`. The JC69 values of that section are exact, and its K2P column is described in Section S3.8; a held-out design on these loci is reported in the companion paper (Schwartz et al., 2026a).

The window-scan examples. The six-window scan of Section S4.5, also drawn as the scan figure of the main text (Section 6.3), used budget 20,000, base seed 20260711, one process, and one whole tRNA locus per window. The throughput figures of Section S4.5 come from two other runs at the same per-window call. One is an *Anopheles* scan (81,113 loci scored as three quartets each, 243,339 rows, 48 cores), the other a survey of two-kilobase windows from a human, chimpanzee, gorilla and orangutan genome alignment. Every phyloexact output tabulated in this supplement can be regenerated from the distributed files with the commands printed in its section.

S4 Certificates and genome-scale tools

S4.1 Purpose of a certificate

Every number phyloexact returns comes with a certificate, a JSON object stating which kind of answer this is, on what data, under what model and prior, and by which engine. It carries enough

PRELIMINARY

intermediate quantities for a separate routine, `px.verify_certificate`, to recompute the claim from the alignment. It guards against honest error, such as the wrong file, the wrong taxon order, a result quoted at a stronger guarantee than it has, a transcription slip, or a corrupted install. It is not a security boundary against someone who can write both the document and the repository, and the package says so. Three design rules apply. No decision depends on a field the document states about itself; every reproduction is exact or within a derived bound; and every test not performed is named in the result.

S4.2 Fields

Table S19 lists the fields every certificate carries and Table S20 the fields each kind adds. The top-level key set is closed per kind, and an extra key makes `px.verify_certificate` fail. Four abridged examples follow, all from the sessions of Section S1.4.

Table S19: Fields common to every certificate.

field	content
`register`	the kind delivered: `R1`, `R2`, `R2-one-sided`, `R3`, `R0`; compounds carry the weakest of their components
`kind`	present on compounds only: `topology-sum`, `model-comparison`, `model-grade`, `adequacy-check`, `prior-predictive`, `pail-marginal`, `pail-row`
`model`, `prior`	the model label as delivered (a fixed-point interval names its point) and the prior in words, with exact parameters where a caller set them
`dataset_sha256`, `dataset`	hash of the normalized rows (Section S1.2); {`kind`, `N_used`, `dropped_columns`, `alphabet`} and, for recoded input, the recoding and the recoded rows' hash
`taxa`, `topology`	the taxa in the caller's order and the canonical topology key (`null` on a tie-proof certificate without `topology=`)
`suite_version`	`phyloexact-0.2.3`; read only to refuse formats the routine predates or post-dates
`capability_snapshot`	the import-time probe of Table S1 plus library build strings; informational, read by nothing
`wall_seconds`	the producing call's wall; informational
`r0_precheck`	on marginal-likelihood certificates: the tie-proof summary computed before any arithmetic (group orders, forced ties per class, ties involving this topology)
`notes`	human-readable notes on how the number was computed; on an estimate the first note is the error label, which `px.verify_certificate` rebuilds and compares

An exact value (tRNA-Gln, JC69, `12|34`; integers abridged).


```
{"register": "R1", "model": "JC69", "topology": "12|34",
 "prior": "per-edge uniform x_e ~ U(0,1) (JC x-parameterization; equals Exp(4/3) branch-length
    prior)",
 "taxa": ["V00711.1/3842-3772", "X14848.1/3820-3750", "X52392.1/5172-5102", "M10217.1/5909-5841"],
 "dataset_sha256": "825982082dce20d8...", "dataset": {"kind": "sequences", "N_used": 68,
    "dropped_columns": 0, "alphabet": "nucleotide"},
 "value": {"num": "6482252858411398...5420889", "den": "2175876696778021...48000000000"},
 "denominator_bound": {"bound": "den(Z) | 256^68 * lcm(1..69)^5 (Lemma-1 form)", "divides": true},
 "engine": {"pin": "r1_engine", "path": "pins/phylo_k2p_engine/engine.py", "sha256": "18d044c9...",
```


PRELIMINARY

Table S20: Fields added per kind of answer, and how `px.verify_certificate` tests each.

kind	fields	how tested
exact	`value` {`num`, `den`} decimal strings; `denominator_bound` {`bound`, `divides`}; `engine` {`pin`, `path`, `sha256`, `mode`, `entry`, `stats`: N, N_B, m, support sizes}	bound recomputed from N and the lattice form; engine identity resolved from the package's hash table; $Z > 0$ and $Z \leq$ the exact fit ceiling; `stats` against the dataset; full exact recomputation up to $N = 40$ unasked, beyond on request
interval	`enclosure` {`Z_lo`, `Z_hi` exact rationals, float views}; `one_sided`; `box_path`; `pinned_point` {κ, π, s, α, `ncat`, `pinned_by`}; `ml_fit`; `partition_gate`; `node_count`; `prec_bits`; `rigor_doc`; `budget_minutes`; `method`	label rule (two-sided iff $Z_{\mathrm{lo}} > 0$); engine and rigor-note hashes; containment of the exact value wherever one exists (JC69, JC-SR4, JC+R inside its exact limits, a point fixed at the JC limit); free-κ: tiling, box masses and total recomputed from the per-box values
estimate	`point`; `budget`; `seed`; `n_evals`; `estimator_mode`; `replicates` {`R`: 4, `logZ`: [4 values], `spread_nats`}; `calibration` {file id, path, sha256, anchor summary, `tolerances` {file, rule, mode}, `regime`}; `r2_available` when the model also offers an interval	seeded reproduction of the point and the four replicates on the user's machine, held to a bound derived from N, the state count and the reduction depth of the budget; the label rebuilt from the distributed measurement file and the dataset and compared for equality; that file opened through the package's hash table, with no path named by the certificate opened
tie proof	per class (`raw`, `folded_k2p`, `folded`): `generators`, `order`, `order_identity`, `chain_orbit_sizes`, `invariant` classes, `orbits`, `orbit_stabilizer_identity`, `forced_ties`, `elements_listed` (to order 720), `register_statement`, `search_nodes`; `n_topologies`; `action_convention`; `cross_check` (n = 4, 5); in `px.compare`: a `forced_tie` block with the permutation	group rebuilt from the generators (Schreier–Sims), every orbit re-closed, the orbit-stabilizer identity recomputed as arithmetic; with the dataset, every generator and the permutation re-applied to the rows, invariant classes recomputed, brute force over all permutations for $n \leq 6$, re-search for $n \geq 7$
compound	`summary` or `comparison`/`meter`/`row` block; `sub_certificates` embedding every component in full	every component re-tested; components must share dataset hash, taxa, model family and prior; sums, posteriors, winners, Bayes factors, differences and error figures recomputed from the components and compared with the summary

PRELIMINARY

```
            "stats": {"N": 68, "N_B": 43, "m": 25, "base_support": 61754, "mixed_support":
    4422478}},
 "r0_precheck": {...}, "notes": ["CPU exact route ..., N=68 <= 150; inside the measured auto
    envelope (m=25 <= 25 for N <= 68 ...)"],
 "suite_version": "phyloexact-0.2.3", "capability_snapshot": {...}, "wall_seconds": 355.877}
```


A proven interval (tRNA-Gln, HKY85 at the fitted point, three minutes on one process).


```
{"register": "R2-one-sided", "one_sided": true, "box_path": "v3", "budget_minutes": 3,
 "model": "HKY85 at the pinned point (kappa=413/100, pi=['257/1000', '31/250', '133/500',
    '353/1000']; no rate heterogeneity), rationalized from the ML fit ...",
 "pinned_point": {"kappa": "413/100", "pi": ["257/1000", "31/250", "133/500", "353/1000"], "ncat":
    1, "pinned_by": "ml_fit"},
 "enclosure": {"Z_lo": "-3116985526...723/6961731899...216", "Z_hi":
    "6734853901...433/1148130695...", ...},
 "partition_gate": {...}, "node_count": ..., "prec_bits": ..., "rigor_doc": {...}, "engine": {...},
 "notes": [..., "one-sided: the certified lower endpoint did not reach positive within the budget
    (minutes=3); a certified upper bound"], ...}
```


An estimate (tRNA-Gln, JC69, `12|34`, budget 10^5, seed 0).


```
{"register": "R3", "model": "JC69", "topology": "12|34", "point": -261.40315247963173, "budget":
    100000, "seed": 0, "estimator_mode": "v1",
 "replicates": {"R": 4, "logZ": [-261.4033056171121, -261.40302358243497, -261.4032480414158,
    -261.4030327093354],
              "spread_nats": 0.00028203467712728525},
 "calibration": {"calibration_id": "R3-cal-synth-v2", "calibration_sha256": "0592b2d9...",
                 "anchor_grading_summary": "6 JC cases vs R1-exact anchors; mean|dev|=2.8e-03
    @1.0e+04 evals, mean|dev|=3.4e-04 @1.0e+05 evals",
                 "tolerances": {"record_id": "R3-seed-sweep-v2", "record_sha256": "3d1cee25...",
    "mode": "v1", ...}, "regime": {...}},
 "notes": ["R3 estimate (float Monte Carlo), UNGRADED (regime): ... N = 68 usable sites exceeds
    the record's largest swept case N = 32; m = 25 ... exceeds ... m = 9 ...; replicate spread
    0.000282035 nats over 4 internal replicates; no error figure is claimed for this number", ...],
 "r2_available": {...}, ...}
```


A tie proof (hominid tRNA-Phe; `px.decidability`).


```
{"register": "R0", "topology": null, "n_taxa": 4, "n_topologies": 3, "taxa": ["human", "chimp",
    "gorilla", "orangutan"],
 "model": "raw register: ANY leaf-exchangeable model (GTR, CAT, mixtures); folded_k2p register:
    ts/tv-symmetric models ...; folded register: base-symmetric (JC/Neyman)",
 "prior": "any iid branch prior",
 "stabilizer": {
   "raw":        {"order": 1, "generators": [], "orbits": [["12|34"], ["13|24"], ["14|23"]],
    "forced_ties": [], ...},
   "folded_k2p": {"order": 1, "generators": [], ...},
   "folded":     {"order": 2, "generators": [[2, 1, 0, 3]], "orbits": [["12|34", "14|23"],
    ["13|24"]], "forced_ties": [["12|34", "14|23"]],
                  "invariant": {"description": "multiset of agreement counts with every other
    row", "classes": [["human", "gorilla"], ["chimp"], ["orangutan"]]},
                 "orbit_stabilizer_identity": [{"orbit_size": 2, "topology_stabilizer_order": 1,
    "product": 2, "holds": true}, ...],
                 "register_statement": "forced ties hold under base-symmetric models (JC /
    Neyman-k) with iid branch priors", ...}},
 "cross_check": {"engine": "... (full S_n enumeration, n = 4, 5)", "agrees": true}, ...}
```


PRELIMINARY

The tie proof reads as follows. Exchanging human and gorilla (`[2, 1, 0, 3]`) leaves the multiset of base-symmetric pattern classes unchanged and carries `12|34` to `14|23`. Those two marginal likelihoods are therefore equal under JC69 and every base-symmetric model, which the two exact values of Section S1.4 confirm digit for digit. Under models that distinguish the bases no relabeling works, and all three topologies are free.

S4.3 `px.verify_certificate`

```
px.verify_certificate(cert, dataset=None, recompute="auto", strict=False)  ->  dict
```

`cert` is a certificate dict (or one read back from JSON); `dataset` is a `px.Dataset`, a FASTA path, a {taxon: sequence} mapping or a counts mapping (a path, a mapping or a superset is experimental in 0.2.x, Section S1.7). The taxa named in `cert` are selected and ordered first, and the file in another row order, or the superset the quartet was taken from, therefore passes too. The result is a dict with `ok`, the kind, the list `checks` of test rows (`check`, `status` pass/fail/skipped, `kind` structure/value/refusal, `info`), and the counts `n_performed`, `n_skipped`, `n_skipped_informational`, `n_skipped_in_legs`.

Three modes. Under `recompute="auto"` (the default) every value is recomputed inside the routine's own cost ceilings, derived from the dataset and the label alone; no cost field in `cert` is consulted. Beyond a ceiling the value test is a named skip; `ok` can then be `True` with the value unverified, and the skip row says so. `recompute=True` closes the skip by re-running the computation whatever it costs. Under `strict=True` any skipped test anywhere in the tree of components makes `ok=False`, with the skipped rows listed. A pipeline that must not accept an unverified value passes `strict=True`, and `recompute=True` where the cost is acceptable. Rows typed informational (a test not applicable to this kind) do not count as skips.

Worked outcomes. On the exact tRNA-Gln value above, 18 tests pass in 0.06 s and one is skipped by name, `"UNVERIFIED beyond the recompute envelope (N=68 ... RECOMPUTE_N_MAX = 40 ...): the VALUE IS NOT VERIFIED here - it rests on the engine identity and the denominator bound only; pass recompute=True ..., or strict=True to refuse ..."`. `strict=True` accordingly returns `False`, the expected outcome for an exact value above 40 sites until `recompute=True` is run. On the exact model comparison for the 16-column prefix both marginal likelihoods are inside the ceiling; 26 tests pass with no skips in 127 s (the two exact recomputations), and `strict=True` passes. On the three-topology estimate for tRNA-Gln 29 tests pass in 10 s, every point and replicate bit-reproduced from budget and seed. On the HKY85 interval 26 pass and 2 are named skips; the partition identity is declared by the engine and not re-derived, and no exact reference exists away from the JC point. The result is `ok=True` non-strict, and `strict=True` refuses it by design. On the plug-in estimate 27 pass with 1 named skip (no code in the package reproduces a user's likelihood), and `strict=True` refuses. On the tie proof 68 tests pass in 0.003 s, and `strict=True` passes.

Tests per kind. The last column of Table S20 summarizes the tests; three points need more words. An estimate is a seeded reproduction on the user's machine, held to a bound B. B is derived from the site count, the number of states, and the pairwise reduction depth of the budget ($\lceil \log_2 n \rceil + 16$ bits of slack). B is 1.4×10^{-13} log units at budget 10^7 and $N = 1$, so a relative nudge of 3×10^{-10} to a point fails on every build. Estimates at budgets up to 300,000 reproduce under `"auto"`; a larger budget is refused under `"auto"` and `strict` (the budget is part of the

PRELIMINARY

claim) and reproduces only under `recompute=True`. Whether an estimate is inside the measured class is decided by `px.verify_certificate` itself, from the distributed measurement file and the dataset. A claimed figure the file does not support fails, and a compound whose headline reads off a component without a figure is refused under `strict=True`. An interval's containment test runs wherever an exact reference exists, though a too-narrow interval that still contains the reference is not detectable. Away from the JC point a fixed-point interval rests on the hash-bound engine and its partition identity; both are stated limits. Integers longer than 200,000 digits are refused before parsing.

Version window. Every result names the version that emitted it. Three kinds from earlier versions are refused by name as a version gap with the advice `"re-emit"`; they are not reported as tampering. They are model comparisons and model grades from 0.2.0 (their headline changed at 0.2.1) and every estimate from before 0.2.2, with every compound embedding one (the estimator changed). Everything else from 0.2.0 passes under this routine, and inside the window the version label is informational. A later or malformed `suite_version` is refused too.

S4.4 Integrity of the reference files

Every reference file the package tests itself against (exact values, engines, measurement files, datasets) lives under `pins/` with its hash listed in `phyloexact/lanes/_pins.py` and is opened only through that table. A mismatch raises `PinIntegrityError` whichever path served the bytes. The validation suite plants a one-byte corruption to confirm the refusal fires and then restores the file. Results and scan rows name these files by bundle-relative path with the hash beside it, the hash being the authority. The measurement files carry a producer block (source-tree hash, script hash, dataset hashes, platform string, time stamps, and no host name), and the suite binds each file to the code that produced it.

S4.5 The window-scan tool

The scan (experimental in 0.2.x, Section S1.7) applies the quartet call to many windows of one four-taxon alignment and writes one JSON row per window. A row holds the posterior over the three topologies (an estimate), a tie flag from the window's own tie proof, the exact fit ceiling, and m. It is crash-safe and resumable, and one malformed window does not stop the run.

Input. Rows mode takes a JSON file of aligned windows:

```
{"taxa": ["human", "chimp", "gorilla", "orangutan"],
 "windows": [{"id": "mt-tRNA-phe", "rows": ["GTTTATGTAGCTTACCTCC-T-CAAAGC...",
    "GTTTATGTAGCTTACCCCC-T-...", "...", "..."]},
            {"id": "mt-tRNA-gln", "rows": ["TAGGATGGGGTGTGATAGGTGGCACGG...", "...", "...",
    "..."]}, ...]}
```

Rows pass through the one nucleotide normalization (upper-cased, `U` read as `T`) and the one missing-data convention (a column with a gap or ambiguity letter dropped and counted in the row's `dropped`). Fold-counts mode (`--windows`) takes an $(n_w, 15)$ `.npy` array of base-symmetric class counts with columns in the public order `px.FOLD_LABELS`, or a JSON list of {label: count} mappings. That order is `xxxx xxxy xxyx xxyy xxyz xyxx xyxy xyxz xyyx xyyy xyyz xyzw xyzx xyzy xyzz`; `--help` prints it and the manifest states it. A rows file of the wrong top-level shape is refused before anything is scanned.

PRELIMINARY

Command and options.

```
$ python -m phyloexact.hill.scan --rows windows.json --out scan_dir --procs 1
scan: 6 scored + 0 refused -> scan_dir/scan_out.jsonl (14.0s, 0.43 windows/s); anchors queued: 2;
    manifest: scan_dir/MANIFEST.json
```

Table S21: Options of `python -m phyloexact.hill.scan`.

option	default	meaning
`--rows FILE` / `--windows FILE`		input: aligned windows (JSON) or fold-class counts (`.npy` or JSON); exactly one
`--meta FILE`, `--slice I0:I1`		fold mode: window metadata; a window index range for sharding a chromosome across processes
`--out DIR`		output directory (`scan_out.jsonl`, `MANIFEST.json`, `anchors_queue.jsonl`)
`--budget B`	20,000	likelihood evaluations per window for the posterior estimate; tested against the estimator's floor and 10^7 ceiling before any output
`--seed S`	20260711	base seed; chunk c uses $S + c$
`--chunk K`	512	windows advanced together through the estimator; reproduction needs the same chunk composition and seed
`--procs P`	1	worker processes
`--anchor-m-max M`	7	windows with $\max_T m_T \leq M$ are queued in `anchors_queue.jsonl` as candidates for an exact run (`px.evidence`, `register="R1"`); the scan does not run them itself
`--force-rescan`	off	discard rows already in `--out`; by default a row is kept (resume) only if its `counts_sha256` is this input's window, and a directory holding another input's rows is refused
`--label L`	`w`	window id prefix (fold mode); ids are `L+index`

Output. Each line of `scan_out.jsonl` is one window; Table S22 lists the fields and Table S23 shows a six-window run on hominid tRNA loci distributed with the package. `status` is `scored`, `tied` or `refused`. A window is `tied` in three cases: its tie proof lists all three pairs (a conserved window), the estimate's top margin is exactly 0, or the estimate's mode falls inside a pair the proof ties. Then `r3.winner` is `null`, `r3.tie.reason` says why, and the weight-file tools skip the window instead of writing an arbitrary topology. Identical windows within a chunk (any row order, any U/T or case spelling) are estimated once and share the row's numbers. `MANIFEST.json` lists the input file and its hash, the taxa, the normalization applied, the parameters, the hashes of the code and reference files used, and library versions. It also gives the run counts (`windows_requested`, `_scanned`, `_tied`, `_refused`, `anchors_queued`, `wall_seconds`, `windows_per_second`) and the output paths with their hashes; the weight-file tool `verify_cert` (Section S4.6) binds to it by its bytes.

Throughput. On two-kilobase genome windows the scan measured 24.6 ms per window on 16 cores (2,048 windows in 50.4 s) and 300 ms per window on one core. A run of 243,339 window-topology rows (81,113 *Anopheles* loci by three quartets) took about 105 minutes on 48 cores, 38.6 windows per second with the full per-window output. Both throughput figures were measured with an earlier version of the scan tool. The tRNA example above is slower per window (0.43 per second on one process) because six windows do not fill a chunk.

PRELIMINARY

Table S22: Fields of one `scan_out.jsonl` row.

field	content
`id`, `status`, `suite`	window id; `scored` / `tied` / `refused` (with `refusal.name`, e.g. `CorruptWindow`); `hill-h1/v1`
`N`, `dropped`, `m`, `m_by_topology`, `counts_sha256`	usable columns, dropped columns, m for the leading topology and for all three, hash of the window's pattern counts (the resume key)
`r0`	the window's tie proof in the `raw` and `folded` classes: `order`, `nontrivial` permutations, `forced_ties`, class statement
`r3`	`posterior` {topology: probability}, `winner`, `logZ` per topology, `margin_nats`, `tie` {`tied`, `reason`}, `model`, `prior`, `engine`, `register: "R3"`, `register_statement` and `error_bars: "measured-empirical off-anchor"` (the posterior is an estimate whose error measurement covers 67- to 69-column tRNAs only), `calibration_reference`
`ceiling`, `gap`	the exact fit ceiling on the 4^4 raw patterns as an integer pair with its ln and ln per site; the ceiling gap of the winning topology's estimate against it (`gap_nats_per_site`, kind `misspec-gap`, the weaker kind of its two inputs)
`anchor`	{`eligible`, per-topology m, the rule}: whether the window was queued for an exact run
`chunk`	`base_seed`, `seed`, `budget`, `chunk_index`, `index_in_chunk`, `size`, `r3_wall_seconds`

Table S23: A six-window scan over hominid mitochondrial tRNA loci distributed with the package (human, chimpanzee, gorilla, orangutan; one locus per window; `--procs 1`, budget 20,000; 14 s). The posterior over `12|34` (human+chimpanzee), `13|24` (human+gorilla) and `14|23` (human+orangutan) is an estimate; the ceiling gap is against the exact ceiling. At tRNA-Phe the exact posterior is 0.3332/0.3336/0.3332 (Section S1.4) and the tie proof lists `12|34`∼`14|23`.

window	N	dropped	m	winner	P(`12|34`)	P(`13|24`)	P(`14|23`)	gap (log units/site)
tRNA-Phe	69	4	10	`13|24`	0.333	0.334	0.333	0.511
tRNA-Gln	72	0	6	`12|34`	0.910	0.045	0.045	0.478
tRNA-Leu(CUN)	71	0	2	`12|34`	0.964	0.018	0.018	0.399
tRNA-Asp	67	2	13	`14|23`	0.001	0.001	0.998	0.686
tRNA-Pro	68	2	11	`13|24`	0.021	0.971	0.008	0.697
tRNA-Val	69	0	11	`13|24`	0.269	0.450	0.280	0.517

PRELIMINARY

Labels at genome scale. Genome windows at $m \approx 120$ to 190 lie outside the class on which the estimator's error was measured, so every window posterior carries the off-class statement and no error figure. The tie flags and the fit ceilings are exact at any size. Per-window products are not a multispecies-coalescent method, and the manifest says so.

S4.6 Quartet weight files

```
$ python -m phyloexact.hill.weights --scan scan_dir/scan_out.jsonl --manifest
    scan_dir/MANIFEST.json --out weights_dir --label mt-tRNA [--taxa A,B,C,D]
$ head -2 weights_dir/mt-tRNA.wastral.nwk
((human,gorilla)0.33401200355754823,(chimp,orangutan)0.33401200355754823);
((human,chimp)0.9100964936451336,(gorilla,orangutan)0.9100964936451336);
$ head -3 weights_dir/mt-tRNA.wqfm.wqt
((human,chimp),(gorilla,orangutan)); 0.33299399822122594
((human,gorilla),(chimp,orangutan)); 0.33401200355754823
((human,orangutan),(chimp,gorilla)); 0.33299399822122594
$ python -m phyloexact.hill.verify_cert weights_dir/mt-tRNA.wastral.nwk.certificate.json
    --manifest scan_dir/MANIFEST.json
{... "n_checks": 21, "failed": [], "not_performed": [], "verdict": "VERIFIED", "exit": 0}
```

The emitter (experimental in 0.2.x, Section S1.7) writes input files for the weighted quartet species-tree methods wASTRAL (Zhang and Mirarab, 2022) and wQFM (Mahbub et al., 2021) from a scan output and its manifest. The wASTRAL file has one support-annotated Newick per window with the winner's posterior on the internal edge (run wASTRAL with `--mode 2 -x 1 -n 0`); the wQFM file has three weighted quartets per window. Taxa come from `--taxa` or the manifest's `source.taxa`; tied windows are skipped and listed. Each file gets a sidecar `*.certificate.json`. It states the weight definition (`r3.posterior` under a uniform topology prior), the kind (`R3`) and its error statement, the format grammar and the program version the format was tested against, `rows_scored`, `rows_skipped` and `skipped`. It also gives the source scan and manifest by path and hash, `taxa_default`, and the output's line count and hash. Every sidecar carries two fixed sentences. A scope sentence says that the guarantee covers the input file and not the tree a downstream method builds from it. A conditional sentence says when such weights favor the true quartet, namely under the multispecies coalescent with JC69, asymptotically in the number of loci, proven for the caterpillar species shape only. The weight-file writers have been exercised on synthetic fixtures and small examples like this one; no genome-scale species-tree run with these files is reported.

`verify_cert`. `python -m phyloexact.hill.verify_cert CERT (--manifest M | --scan S --taxa A,B,C,D) [--weights W] [--json OUT]` binds the weight file the caller holds to the scan output the caller holds. Every line is re-rendered from the scan rows under the caller's leaf names and compared byte for byte. The scan and manifest hashes stated in the sidecar must equal the caller's files, and `taxa_default` must equal the caller's taxa; the paths written in the sidecar are not opened. Exit codes: 0 `VERIFIED`; 3 `FORMAT-VERIFIED` (no scan output supplied, or `--no-sources`); 4 `FORMAT-VERIFIED` with a scan output but no caller-held taxa (value binding not performed); nonzero with `failed` rows otherwise. A manifest re-serialized with different whitespace is another file and fails the binding, by design.

S4.7 Exact fit ceilings at alignment scale

PRELIMINARY

```
Z, cert = px.hill.sup_ceiling_from_counts({"AAAA": 25, "AAAC": 1, "CCCC": 12, ...})     # pattern
    -> count
pc      = px.hill.pattern_counts_from_fasta("alignment.fasta", alphabet="protein")     # any taxon
    count, any alphabet
Z, cert = px.hill.sup_ceiling(pc)                                                      # gap-mask
    strata handled
Z, cert = px.hill.sup_ceiling_partitioned(seqs, order, parts=[("p1", (0, 500)), ("p2", (500,
    1200))])  # (name, half-open range) per partition
```

The ceiling is the unconstrained multinomial likelihood $\prod_k n_k^{n_k}/N^N$ over the distinct site-pattern counts (Goldman, 1993), an upper bound on the fit of every independent-sites model under every prior. It is returned as an exact `Fraction` with a certificate. The partitioned form `px.hill.sup_ceiling_partitioned` is experimental in 0.2.x (Section S1.7). Patterns are hashed column bytes, so the taxon count and alphabet are unrestricted. A nucleotide alphabet reads `U` as `T`, gapped columns are dropped or stratified by gap mask, and an alignment with no usable column is refused (no empty product is returned). At 2,000 taxa by 10^5 sites (4,149 distinct patterns) the ceiling took 2.27 s and 0.80 GB; the integers have about $N \log_{10} N$ digits and are written in sub-quadratic time. This is the quantity `px.grade_model` gaps a model's marginal likelihood against. At supermatrix scale it bounds the fit any model in the class can reach, and the companion paper on precision phylogenetics reports what such ceilings show on published animal matrices (Schwartz et al., 2026b). A related tool compares a published maximum-likelihood value with the ceiling of its own alignment, since a reported log-likelihood above the ceiling is impossible. It requires that program's likelihood conventions to be established first, which we have done for IQ-TREE 2 only.

References

Barry, D. and Hartigan, J. A. (1987). Statistical analysis of hominoid molecular evolution. *Stat. Sci.*, 2:191–207.

Bollback, J. P. (2002). Bayesian model adequacy and choice in phylogenetics. *Mol. Biol. Evol.*, 19:1171–1180.

Bouckaert, R., Vaughan, T. G., Barido-Sottani, J., Duchêne, S., Fourment, M., Gavryushkina, A., Heled, J., Jones, G., Kühnert, D., De Maio, N., Matschiner, M., Mendes, F. K., Müller, N. F., Ogilvie, H. A., du Plessis, L., Popinga, A., Rambaut, A., Rasmussen, D., Siveroni, I., Suchard, M. A., Wu, C.-H., Xie, D., Zhang, C., Stadler, T., and Drummond, A. J. (2019). BEAST 2.5: an advanced software platform for Bayesian evolutionary analysis. *PLoS Comput. Biol.*, 15:e1006650.

Dick, J., Kuo, F. Y., and Sloan, I. H. (2013). High-dimensional integration: the quasi-Monte Carlo way. *Acta Numer.*, 22:133–288.

Fourment, M., Magee, A. F., Whidden, C., Bilge, A., Matsen, IV, F. A., and Minin, V. N. (2020). 19 dubious ways to compute the marginal likelihood of a phylogenetic tree topology. *Syst. Biol.*, 69:209–220.

Goldman, N. (1993). Statistical tests of models of DNA substitution. *J. Mol. Evol.*, 36:182–198.

Hasegawa, M., Kishino, H., and Yano, T. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, 22:160–174.

PRELIMINARY

Höhna, S., Landis, M. J., Heath, T. A., Boussau, B., Lartillot, N., Moore, B. R., Huelsenbeck, J. P., and Ronquist, F. (2016). RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Syst. Biol.*, 65:726–736.

Johansson, F. (2017). Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Trans. Comput.*, 66:1281–1292.

Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. In Munro, H. N., editor, *Mammalian Protein Metabolism*, volume III, pages 21–132. Academic Press, New York.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16:111–120.

Kimura, M. (1981). Estimation of evolutionary distances between homologous nucleotide sequences. *Proc. Natl. Acad. Sci. USA*, 78:454–458.

Lartillot, N. and Philippe, H. (2004). A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. *Mol. Biol. Evol.*, 21:1095–1109.

Mahbub, M., Wahab, Z., Reaz, R., Rahman, M. S., and Bayzid, M. S. (2021). wQFM: highly accurate genome-scale species tree estimation from weighted quartets. *Bioinformatics*, 37:3734–3743.

Newton, M. A. and Raftery, A. E. (1994). Approximate Bayesian inference with the weighted likelihood bootstrap. *J. R. Stat. Soc. Ser. B*, 56:3–26.

Owen, A. B. (1997). Scrambled net variance for integrals of smooth functions. *Ann. Stat.*, 25:1541–1562.

Rannala, B. and Yang, Z. (2003). Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics*, 164:1645–1656.

Ronquist, F., Teslenko, M., van der Mark, P., Ayres, D. L., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M. A., and Huelsenbeck, J. P. (2012). MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. *Syst. Biol.*, 61:539–542.

Schwartz, M. D., Edwards, S. V., and Lewis, P. O. (2026a). Exact Bayesian evidence for phylogenetic trees. Manuscript in preparation.

Schwartz, M. D., Edwards, S. V., and Lewis, P. O. (2026b). Resolving ancient branchings using precision phylogenetics. Manuscript in preparation.

Speagle, J. S. (2020). dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Mon. Not. R. Astron. Soc.*, 493:3132–3158.

Susko, E. and Roger, A. J. (2007). On reduced amino acid alphabets for phylogenetic inference. *Mol. Biol. Evol.*, 24:2139–2150.

Tavaré, S. (1986). Some probabilistic and statistical problems in the analysis of DNA sequences. *Lectures on Mathematics in the Life Sciences*, 17:57–86.

Tuffley, C. and Steel, M. (1998). Modeling the covarion hypothesis of nucleotide substitution. *Math. Biosci.*, 147:63–91.

PRELIMINARY

Wang, Y.-B., Milkey, A., Li, A., Chen, M.-H., Kuo, L., and Lewis, P. O. (2023). LoRaD: marginal likelihood estimation with haste (but no waste). *Syst. Biol.*, 72:639–648.

Xie, W., Lewis, P. O., Fan, Y., Kuo, L., and Chen, M.-H. (2011). Improving marginal likelihood estimation for Bayesian phylogenetic model selection. *Syst. Biol.*, 60:150–160.

Yang, Z. (1994). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J. Mol. Evol.*, 39:306–314.

Zhang, C. and Mirarab, S. (2022). Weighting by gene tree uncertainty improves accuracy of quartet-based species trees. *Mol. Biol. Evol.*, 39:msac215.

PRELIMINARY