Package 'fastphylosig'

November 2, 2026

Type Package

Title Fast Phylogenetic Signal Statistics

Version 0.3.5

Description Provides one-command and method-specific workflows for phylogenetic signal in continuous, binary, and categorical traits. Compiled tree-pruning algorithms implement Blomberg's K and Pagel's lambda without constructing a dense phylogenetic covariance matrix in production. Fritz and Purvis D retains separate random and Brownian null models, while categorical Delta combines discrete ancestral-state likelihoods with MCMC and permutation diagnostics. The package also provides reusable prepared trees, explicit tree/data matching, trait-wise missing-value handling, conservative tree diagnostics, bounded simulation memory, and structured timing and Monte Carlo uncertainty metadata. Clade scans describe local and overall signal in continuous, binary and multistate traits, with optional model-reference tests of signal dispersion or deviation. Methods follow Blomberg, Garland and Ives (2003) <doi:10.1111/j.0014-3820.2003.tb00285.x> for K, Pagel (1999) <doi:10.1038/44766> for lambda, Fritz and Purvis (2010) <doi:10.1111/j.1523-1739.2010.01455.x> for D, and Borges et al. (2019) <doi:10.1093/bioinformatics/bty800> for categorical Delta.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ape, parallel, Rcpp, stats

LinkingTo Rcpp, RcppArmadillo

Suggests caper, expm, mvtnorm, phytools, testthat

URL https://github.com/yinwen-ecology/fastphylosig, https://yinwen-ecology.github.io/fastphylosig/

BugReports https://github.com/yinwen-ecology/fastphylosig/issues

SystemRequirements C++17, OpenMP (optional)

Config/testthat/edition 3

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Author Yin Wen [aut, cre]

Maintainer Yin Wen <wenyinsci@126.com>

Repository CRAN

Date/Publication 2026-11-02 16:20:10 UTC

Contents

fastphylosig-package . . . 2
cache_info . . . 3
check_tree . . . 4
fast_ace . . . 5
fast_d . . . 7
fast_delta . . . 9
fast_hetero . . . 12
fast_k . . . 19
fast_lambda . . . 20
fast_signal . . . 21
match_phylo_data . . . 23
match_tree_data . . . 24
plot.fastphylosig_heterogeneity . . . 24
plot_signal . . . 26
prepare_tree . . . 27
resolve_tree . . . 28
result_methods . . . 29

Index **31**

`fastphylosig-package` *Fast phylogenetic signal calculations*

Description

One-command and method-specific workflows for selected phylogenetic signal statistics. The method-explicit `fast_signal(tree, data, method, ...)` entry point checks and matches the inputs, then runs K, lambda, D, or Delta. Continuous-trait Blomberg's K and Pagel's lambda preserve the `phytools::phylosig()` targets for `se = NULL`; the binary Fritz and Purvis D statistic follows `caper::phylo.d()`; categorical Delta uses accelerated discrete ancestral-state likelihoods and reports MCMC diagnostics.

Details

Start with fast_signal for a standard analysis. Advanced users can run tree and data preparation explicitly, then call fast_k, fast_lambda, fast_d, fast_delta, and fast_ace. Repeated analyses can reuse prepare_tree. Utilities include cache_info, check_tree, resolve_tree, match_tree_data, and plot_signal.

Production K and lambda calculations use compiled tree-pruning algorithms. K randomizations are evaluated in bounded chunks, and lambda optimization is completed within one compiled batch call. Public utilities provide method-specific tree checks, conservative representation normalization, progress output, runtime reporting, and structured results without exposing numerical-engine choices.

Missing values are handled trait by trait. Retained-tip patterns are grouped, and species matching/removal counts are reported. D retains separate random and Brownian null P values. K, D, and Delta report Monte Carlo precision where applicable; Delta additionally reports ESS, split R-hat, and estimate MCSE.

plot_signal() provides statistic-specific diagnostics: a permutation null plot for K, a profile-likelihood plot for lambda, a two-null Fritz and Purvis D calibration plot, and a permutation null plot for Delta.

Clade-scale signal

fast_hetero reports local signal estimates, coverage, trait-wise failures and optional complete-tree comparisons. K dispersion can be calibrated against BM; lambda uses an approximate common-local-crown Gaussian reference. Binary D's count-conditioned ER reference is experimental. Delta is recommended for local/overall description and MCMC diagnostics; its explicit experimental heterogeneity options remain available. Ordinary signal probabilities and heterogeneity-reference probabilities answer separate questions.

---

cache_info *Inspect a prepared-tree cache*

Description

Reports structural and numerical cache use without modifying the prepared tree. Dense numerical entries are managed by a byte-budget least-recently-used cache.

Usage

```
cache_info(ctx)
```

Arguments

ctx	An object returned by prepare_tree().

Value

A list containing budget, bytes_used, numbers of structural and numerical entries, hit/miss/eviction counters, and an entries data.frame. The table reports retained tip count, structural, numerical and total bytes, VCV, Cholesky and eigensystem availability, and last-access order for each key.

Examples

```
set.seed(1)
tree <- ape::rtree(20)
ctx <- prepare_tree(tree, cache_budget = 64 * 1024^2)
cache_info(ctx)
```

---

check_tree *Check a phylogenetic tree before signal calculations*

Description

Inspect a phylogenetic tree without reordering, pruning, compiling, or otherwise changing it. Readiness is reported separately for each selected signal method. K, lambda, and D can use compatibility paths for polytomies; Delta follows fast_ace() and requires a rooted, fully-dichotomous tree with strictly positive branch lengths. The summary reports both the conventional degree-two rooted flag and the structural root used by the compatibility kernels, so a root polytomy or an unrooted encoding is visible rather than silently treated as a binary rooted tree.

Usage

```
check_tree(tree, signal = c("K", "lambda", "D", "Delta"))
```

Arguments

tree	A phylo object or an object returned by prepare_tree().
signal	Character vector selecting one or more methods. The default checks "K", "lambda", "D", and "Delta".

Value

An object of class fastphylosig_tree_check. Its ready field is a logical scalar and ready_by_signal is a named logical vector. tree_summary contains tip, topology, rootedness, ultrametricity, branch-length, polytomy, unary-node, and zero/near-zero branch diagnostics. Zero internal branches are conditional for lambda; zero terminal branches require user review at the lambda upper boundary. The issues field is a data frame with code, severity, signal, message, check, problem, action, and auto_fixable columns. Severity is one of INFO, WARNING, ERROR, or USER_ACTION_REQUIRED. Representation-only canonical node-ID diagnostics may set auto_fixable = TRUE; biological rooting, topology, and branch-length repairs are never automatic. The input tree is never modified.

Examples

```
if (requireNamespace("ape", quietly = TRUE)) {
  tree <- ape::read.tree(text = "((a:1,b:1):1,(c:1,d:1):1);")
  check_tree(tree)
  check_tree(tree, signal = c("K", "lambda"))
}
```

`fast_ace` *Fast discrete ancestral-state reconstruction*

Description

Optimized subset of `ape::ace()` for discrete maximum-likelihood ancestral-state reconstruction. The statistical target follows `ape::ace(type = "discrete", method = "ML")` for `"ER"` and `"ARD"` models, while the repeated pruning likelihood calculation is implemented in Rcpp/RcppArmadillo. ARD full-tree ancestral probabilities use the row-generator transition convention. Rate standard errors use observed likelihood information. These two corrections intentionally differ from legacy `ape::ace()` downpass and deviance-Hessian standard errors.

Usage

```
fast_ace(x, phy = NULL, type = "discrete", method = "ML", CI = TRUE,
  model = c("ER", "ARD"), kappa = 1, ip = 0.1, marginal = FALSE,
  prepared = NULL, progress = interactive())
```

Arguments

`x`	Categorical trait vector with one value per tip. Names, when present, must be unique and have exactly the same species set as `phy$tip.label`; their order may differ and values are reordered to tree-tip order.
`phy`	A rooted, fully dichotomous `"phylo"` object with finite, strictly positive branch lengths.
`type`	Currently only `"discrete"` is supported.
`method`	Currently only `"ML"` is supported.
`CI`	Whether to return ancestral likelihoods in `lik.anc`.
`model`	Discrete transition model, `"ER"` or `"ARD"`; the default is `"ER"`.
`kappa`	Optional branch-length exponent, matching `ape::ace()`.
`ip`	Initial value for transition-rate optimization.
`marginal`	Whether to return marginal likelihoods. The default `FALSE` matches the default path used by `ape::ace()`.
`prepared`	An optional object returned by `prepare_tree()`. When supplied, it supplies a validated phylogenetic tree and takes precedence over `phy`. V2 integrity is checked before cached evidence is reused. The ACE likelihood itself uses the tree edges and does not require the dense covariance cache.
`progress`	Whether to report method-level progress stages and elapsed runtime. Defaults to `interactive()`. Setting this to `FALSE` suppresses progress messages; warnings and errors are still signalled.

Details

Discrete fits retain the spectral pruning arithmetic for ordinary eigensystems. Defective or ill-conditioned eigensystems use deterministic matrix-exponential recovery; recovery counts are distinct from final optimizer nonconvergence. Returned ancestral probability rows must be finite, normalized and inside [0,1]. Only machine-scale boundary excursions may be clipped and renormalized; material invalidity triggers a stable recomputation or an explicit failure. These abnormal paths can change estimates; no random starts or random numerical recovery are used.

Likelihood-only fits (`CI = FALSE`, including Delta's internal ACE fits) reuse a private native workspace. Sufficiently large uncached branch-length workloads evaluate independent complex exponent vectors in OpenMP batches bounded by 2 MiB. Matrix products, pruning accumulation and optimizer calls retain their sequential order. No random draws occur in the parallel region. The `fastphylosig.ace_threads` option defaults to `0L` and selects at most two threads. Set this option to `1L` for serial fitting, or a positive integer to request more threads, capped by the OpenMP runtime limit. Small fits, builds without OpenMP and nested OpenMP calls use the serial path. Exactly repeated branch lengths use an exponent-vector cache bounded by 64 MiB per fitting workspace, with capacity determined by state count and actual element size. This bounds exponent storage, not total memory; mixed-length fits batch only the remaining uncached branches using a compact worklist. The activation threshold counts uncached work, so a large tree with very few uncached branches does not incur parallel startup costs. The default `CI = TRUE` path is unaffected by this thread option.

With `marginal = FALSE`, ancestor probabilities condition on all observed tips under a uniform root prior; `marginal = TRUE` returns scaled subtree likelihoods. In the full-tree downpass, a child's contribution for each parent state is PL, where $P = \exp(Qt)$ has parent states in rows. Impossible ARD parent states with zero posterior and zero child contribution send zero messages. A positive posterior with zero contribution remains a failure. ER retains its existing floating-point arithmetic and Delta random stream.

The deviance objective is negative twice log likelihood, so its Hessian divided by two is the observed information. Returned rate standard errors account for this factor. Boundary or unidentifiable fits can still have unavailable standard errors; correcting this scale does not make Wald inference valid at boundaries.

Value

An `ace`-like object containing `loglik`, `rates`, numerical rate standard errors in `se`, `index.matrix`, optimizer status, and, when `CI = TRUE`, `lik.anc`. A `timing` attribute and list field report elapsed runtime in seconds.

See Also

`fast_delta()`, `fast_signal()`

Examples

```
set.seed(5)
tree <- ape::rtree(30)
x <- ape::rTraitDisc(tree, model = "ER", k = 3, rate = 1,
                     states = letters[1:3])
fast_ace(x, tree, model = "ER")
```

fast_d *Fast caper-compatible phylo.d calculation*

Description

Computes the Fritz and Purvis D statistic for binary traits using the same null-model structure as `caper::phylo.d()`. The contrast-sum calculation is moved into C++ for repeated simulation work. Brownian null states are simulated directly along tree edges and then thresholded by the observed state proportion, avoiding dense VCV construction for this step. Binary labels are recoded to equally spaced 0/1 states, making D invariant to the original labels. For ordinary binary trees without controlled state matrices, null generation, thresholding, traversal, and accumulation are fused in chunks. Peak null-state memory is therefore O(n times `chunk_size`) rather than O(n times `nsim`). Controlled and polytomy cases use the compatibility path.

Usage

```
fast_d(tree, x = NULL, test = TRUE, nsim = 1000, se = NULL,
  return_sim = test, verbose = TRUE, rnd.bias = NULL, permutations = NULL,
  random_states = NULL, brownian_states = NULL, ncores = 1, X = NULL,
  permut = NULL, prepared = NULL, chunk_size = 128,
  keep_null = return_sim, progress = interactive())
```

Arguments

`tree`	An object of class `"phylo"` with finite, strictly positive branch lengths and no single-child internal nodes. Safe representation-only differences in root numbering and edge order are normalized automatically. Ordinary binary trees and rooted polytomies are supported; ultrametricity is not required, while unsafe topology changes remain errors.
`x`, `X`	Named binary trait vector or matrix/data.frame of binary traits. `X` is a compatibility alias for `x`; supply only one. Species identifiers are matched by name and reordered to tree-tip order.
`test`	Whether to return randomization and Brownian P values.
`nsim`, `permut`	Number of simulations. `permut` is a `caper::phylo.d()`-style compatibility alias for `nsim`; when supplied, it takes precedence.
`se`	Unused; binary states should be supplied directly in `x`.
`return_sim`	Whether random and Brownian simulation vectors may be returned. Its default is `test`; actual retention also requires `keep_null = TRUE`.
`verbose`	Whether to report species matching.
`progress`	Whether to report method-level progress stages and elapsed runtime. Defaults to `interactive()`. Setting this to `FALSE` suppresses progress and species-matching messages; warnings and errors are still signalled.
`rnd.bias`	Optional sampling weights for the random null model.
`permutations`	Optional permutation matrix for the random null model.

random_states, brownian_states
: Optional state matrices for controlled debugging.

ncores
: Number of OpenMP threads for the C++ contrast-sum simulation kernel. Binary trees can use multiple threads, although scaling is often modest because traversal is memory-bandwidth bound. Polytomies fall back to the serial vectorized path to preserve caper-compatible handling.

prepared
: An optional object returned by `prepare_tree()`. When supplied, its V2 integrity is validated before cached tree calculations are reused and `tree` is ignored.

chunk_size
: Integer from 32 to 512 controlling the number of null states generated and traversed per streaming block.

keep_null
: Whether full random and Brownian null-statistic vectors may be retained. Both `return_sim = TRUE` and `keep_null = TRUE` are required. Set either to `FALSE` to keep only online means, tail counts, D, and P values.

Value

For vector input, a compact `phylo.d`-like object without caper's `NodalVals` or embedded `comparative.data`. In addition to the legacy `Pval1`/`Pval0` fields, it reports the null-specific aliases `P_random`/`P_Brownian`, their Bernoulli Monte Carlo standard errors `MCSE_P_random`/`MCSE_P_Brownian`, and `nsim_requested`, `nsim_successful_random`, `nsim_successful_brownian`, `nsim_failed_random`, and `nsim_failed_brownian`. The two P values are never combined. For matrix/data.frame input, a data.frame with one row per binary trait and the same fields (with the legacy `Pval1_fast`/`Pval0_fast` columns retained for compatibility).

Rooted polytomies use the serial compatibility path. Unary internal nodes are not supported, including unary nodes exposed after species matching or trait-wise missing-value pruning.

Every trait also has a machine-readable `status` (`"ok"`, `"partial"`, `"invalid_trait"`, `"insufficient_data"`, or `"failed"`) and a human-readable `message`/`note`. These fields distinguish invalid traits and incomplete simulation sets from a valid result.

The MCSE uses `sqrt(p * (1 - p) / nsim_successful)` separately for each null. Thus the reported MCSE is exactly zero when the estimated P is 0 or 1, and is `NA` when no finite simulation succeeded. Non-finite simulation statistics are excluded from that null's denominator and explicitly reported in the successful/failed counts and `note`; they are not silently counted as non-extreme values. With `test = FALSE`, P and MCSE columns are returned as `NA`, although simulation counts still describe the null calibration used to compute D. Results expose a `timing` attribute with at least `total_elapsed` (seconds); vector results also provide a `timing` list field.

See Also

`fast_signal()`, `check_tree()`

Examples

```
set.seed(3)
tree <- ape::rtree(30)
z <- ape::rTraitCont(tree)
x <- as.integer(z > stats::median(z))
names(x) <- tree$tip.label
```

```
fast_d(tree, x, nsim = 20, return_sim = FALSE, verbose = FALSE)
```

`fast_delta` *Fast Delta statistic for categorical traits*

Description

Computes the single-tree Delta statistic for categorical traits. Ancestral state probabilities are estimated with `fast_ace()`, node uncertainty is transformed to entropy, and Delta is the posterior `mean(beta / alpha)` from two chains. The lognormal proposal includes its Hastings density ratio, targeting the Beta likelihood with independent exponential priors on raw alpha and beta. Ancestral reconstruction, entropy, and MCMC calculations use Rcpp/RcppArmadillo. The ape comparison path is retained internally for regression tests, not exposed as a user engine.

Since version 0.3.5, the shared ACE downpass corrects the ARD child-to-parent contribution under the row-generator convention. This can change ARD Delta, local/overall summaries, permutation values and subsequent RNG relative to 0.3.4, including ordinary descriptive use. ER retains its previous arithmetic and random stream. The ACE rate-standard-error correction does not enter Delta, which does not estimate or use those standard errors.

Tiny entropy roundoff excursions are repaired only when the corresponding probability row is finite, bounded in [0,1] and normalized within a small tolerance. Repaired endpoints use the existing jitter, emit a warning and are described in the result note. Healthy calculations retain their original numerical and RNG behavior; formerly failed calculations can continue with a different subsequent RNG state.

Usage

```
fast_delta(tree, x = NULL, test = FALSE, nsim = 1000, se = NULL,
  mcmc_sim = 10000, thin = 10, burn = 100, lambda0 = 0.1,
  proposal_sd = 0.5, entropy = c("LSE", "SE", "GINI"),
  model = "ARD", permutations = NULL, return_sim = test,
  verbose = TRUE, ncores = 1, X = NULL, prepared = NULL,
  progress = interactive())
```

Arguments

`tree`	A conventionally rooted, fully binary `"phylo"` object with finite, strictly positive branch lengths, or a compatible context returned by `prepare_tree()`. The method does not require ultrametricity.
`x`, `X`	Named categorical trait vector or matrix/data.frame. Matrix and data.frame row names must be species names. `X` is a compatibility alias for `x`; supply only one. Species identifiers are matched by name and reordered to tree-tip order.
`test`	Whether to compute a permutation P value.
`nsim`	Number of trait permutations when `test = TRUE`.
`se`	Unused; retained so unsupported measurement-error input fails clearly. Use `proposal_sd` for the MCMC proposal.

mcmc_sim	Number of MCMC iterations per chain and Delta calculation.
thin	MCMC thinning interval.
burn	First MCMC iteration eligible to be saved.
lambda0	Rate of the exponential priors on alpha and beta.
proposal_sd	Standard deviation of the log-normal random-walk proposal.
entropy	Entropy transformation: "LSE", normalized Shannon entropy "SE", or normalized Gini impurity "GINI".
model	Discrete transition model used by fast_ace(): "ER" or "ARD".
permutations	Optional integer permutation matrix with exactly nsim rows and one column per retained tip.
return_sim	Whether to retain simulated Delta values.
verbose	Whether to report species matching.
progress	Whether to report method-level progress stages and elapsed runtime. Defaults to interactive(). Setting this to FALSE suppresses progress and species-matching messages; warnings and errors are still signalled.
ncores	Number of worker processes for the Delta permutation test. Small jobs can be slower in parallel because process startup is not free. Replaying stochastic results requires a fixed worker count; changing it repartitions worker RNG streams and is not draw-for-draw reproducible.
prepared	Optional object returned by prepare_tree(); its V2 integrity is validated before cached evidence is reused.

Details

Native ACE scratch can be reused across fits within one analysis. The fastphylosig.ace_cache_bytes option limits retained storage, with a default of 64 * 1024^2 bytes. Zero disables cross-fit caching. Active fits and allocator overhead are outside this limit. The cache is released on return/error and is absent from results. PSOCK workers use owned per-fit scratch. Version 0.3.2.9000 corrects both posterior aggregation and MH acceptance. Results and RNG trajectories intentionally differ from earlier versions. ESS_Delta and split_Rhat_Delta assess the beta/alpha sequence. These are traditional split R-hat and bounded-lag ESS, not rank-normalized or tail diagnostics. The separate rank-normalized fields use pooled mid-ranks, split chains and Geyer's positive/monotone ESS sequence with FFT autocovariance. The sampler still uses two chains; longer runs and independent chain checks are appropriate for final inference. The warning thresholds screen for poor runs; they do not establish convergence. For final inference, use longer runs, check all three sequences, and require MCSE small relative to the differences being interpreted.

The output records optimizer_warning, numerical_recovery, numerical_recovery_evaluations, entropy_recovery, entropy_recovery_rows, other_warning, calculation_failed and failure_stage separately. Recoveries are distinct from optimizer warnings and calculation errors. Single-state traits remain unavailable and are not counted as calculation errors. Defective or ill-conditioned eigensystems use stable matrix exponentials. Successful default fits retain their optimization path. An unfinished default ARD fit in the Delta workspace receives at most one parameter-scaled nlminb continuation and, only if needed, one bounded L-BFGS-B attempt, each limited to 150 iterations. They reuse the existing C++ objective and do not draw random numbers. Original optimizer returns, budgets and candidate selection are retained. Optional optimizer_retry, optimizer_recovered

and optimizer_retry_details fields appear only when a retry occurred; remaining optimizer warnings are not hidden or relabelled. Common multistart and public ACE rate fitting are unchanged. ACE permits only machine-scale probability boundary recovery; Delta validates every probability row before entropy or endpoint jitter. These abnormal paths can change estimates and subsequent random streams. Within the corrected estimator, ordinary paths retain their arithmetic and draw order. Numerical recovery is reported, not treated as a convergence failure or evidence of model adequacy.

Delta reuses the likelihood-only native ACE fitting workspace. Large uncached branch-length workloads may use at most two OpenMP threads by default for independent exponent calculations, combining exact repeated-length caching with batches of the remaining branches. Set the fastphylosig.ace_threads option to 1L to keep those fits serial; see fast_ace() for the thread option and bounded storage. Pruning, optimizer calls on successful fits and MCMC retain their original order. When ncores > 1, PSOCK permutation workers use one inner ACE thread to avoid nested parallelism; the observed fit still follows the option. Changing this ACE option does not reallocate Delta's random streams. Changing ncores does.

Value

For vector input, a phylo_delta-compatible object; matrix/data.frame input returns one row per trait. Results report Delta, alpha/beta chain means and SDs, ESS, split R-hat, batch-means MCSE_Delta for the ratio trace, requested and successful iteration counts, and safe RNG metadata. Tests also report P, MCSE_P, failed fits, and successful permutation counts. Inspect the diagnostics before interpreting short runs. Results expose a timing attribute with at least total_elapsed (seconds); vector results also provide a timing list field. Per-trait status and message/note fields distinguish successful, partial, invalid-trait, insufficient-data, and failed calculations.

Fewer than two species shared globally by the tree and data is a call-level error. After matching and trait-wise NA pruning, Delta returns a per-trait insufficient-data result when a trait retains fewer than two species. A batch with one or more traits retaining 2–19 species emits one warning and still returns estimates. The metadata records all per-trait retained/removed counts and sample-size status (including the fewer-than-two error), plus the affected trait details, in attr(result, "analysis_metadata")$delta_retained_species.

For a finite Delta estimate, non-finite chain diagnostics, ESS below 20, or traditional split R-hat above 1.1 emit one batch warning. The alpha, beta and beta/alpha sequences also have rank-normalized/folded rank_Rhat_*, ESS_bulk_* and ESS_tail_*. Screening flags rank R-hat above 1.01, bulk/tail ESS below 100 per original chain, unavailable diagnostics, or relative_MCSE_Delta = MCSE_Delta / Delta above 0.05. These thresholds are diagnostic heuristics, not a convergence guarantee. Constant traces have unavailable modern diagnostics. This warning never reruns a chain or alters requested iterations. Per-trait diagnostic warning details are retained in attr(result, "analysis_metadata")$delta_mcmc_diagnostics.

The public defaults include test = FALSE, nsim = 1000, return_sim = test, mcmc_sim = 10000, thin = 10, burn = 100, and model = "ARD". Thus permutation P, MCSE_P, and retained null values are absent unless testing is enabled; null values additionally require return_sim = TRUE. MCSE_P uses the finite successful permutation count. These are computational defaults rather than a guarantee of adequate mixing for every data set. Users should increase the chain when ESS, split R-hat, or Monte Carlo error is inadequate.

See Also

fast_signal(), fast_ace()

Examples

```
set.seed(4)
tree <- ape::rtree(25)
x <- sample(letters[1:3], ape::Ntip(tree), replace = TRUE)
names(x) <- tree$tip.label
fast_delta(tree, x, mcmc_sim = 10000, thin = 10, burn = 100,
           verbose = FALSE)
```

---

fast_hetero *Scan phylogenetic signal across clades*

Description

Computes K, lambda, binary D or categorical Delta on crowns selected independently of the traits. Describes local signal strength and, optionally, its deviation from the complete-tree estimate. Automatic selection uses maximal, non-overlapping clades within the size bounds. Explicit nodes also permit nested clades for descriptive exploration. Node numbers always refer to the tree supplied to this function, before any internal canonicalization. For prepared contexts they refer to tree$tree. Local differences describe the observed pattern of phylogenetic conservatism at the selected scale; they do not identify the evolutionary mechanism. Delta is recommended for local/overall description only. Its explicitly requested heterogeneity tests are retained for experimental reproducibility.

Usage

```
fast_hetero(
  tree,
  data,
  method = c("K", "lambda", "D", "Delta"),
  nodes = NULL,
  min_tips = 20L,
  max_tips = 200L,
  test = FALSE,
  nsim = 999L,
  simulation_chunk = 32L,
  keep_null = FALSE,
  max_total_tips = 1000000L,
  progress = interactive(),
  overall = FALSE,
  overall_test = FALSE,
  overall_nsim = nsim,
  d_nsim = 1000L,
  delta_control = list(),
  clade_test = FALSE,
```

```
  clade_nsim = overall_nsim,
  lambda_reference = c("REML", "ML")
)
```

Arguments

tree	A phylo tree or a prepared tree context.
data	Vector, matrix or data frame, named by species. K/lambda require numeric traits; D accepts two states and Delta two or more states (including factors/character labels). Unnamed data with one row per tree tip are assumed to be in tip order.
method	"K" (default), "lambda", "D" or "Delta" (Δ accepted).
nodes	Optional unique internal node numbers in the supplied tree. Explicit nodes override the automatic size bounds; min_tips still sets the minimum number of observed species per trait and clade.
min_tips	Minimum observed species per clade (default 20, at least 3).
max_tips	Maximum total clade size for automatic selection, a finite integer at least 3. It must be at least min_tips only when nodes = NULL.
test	Optional reference test of signal variation (default FALSE). K provides a BM reference; lambda provides an approximate common-local-lambda reference; binary D's conditional ER extension is experimental. Delta heterogeneity testing is experimental and not recommended; use test = FALSE, overall = TRUE for Delta description. K/lambda/D calibrate between-crown dispersion; Delta calibrates local/overall log deviations and implies overall = TRUE. Requires non-overlapping clades and at least two estimable crowns. It is not a test of evolutionary-rate heterogeneity.
nsim	Number of heterogeneity replicates when test = TRUE, at least 2.
simulation_chunk	Maximum K BM replicates evaluated together. Lambda, D and Delta use one replicate workspace and ignore this setting.
keep_null	Retain the simulated discrepancy values in the result.
max_total_tips	Limit on the sum of selected clade sizes. This guards against quadratic work when many nested nodes are requested.
progress	Whether to report calculation stages.
overall	Calculate the signal on the complete supplied tree, using all observed tips for each trait, including tips outside the selected crowns. Optional to keep large-tree scans inexpensive.
overall_test	Test the complete-tree signal. Implies overall = TRUE. K uses tip randomizations; lambda uses the existing likelihood-ratio test against lambda = 0; D uses its random/Brownian signal tests; Delta uses tip randomizations. These differ from the heterogeneity test.
overall_nsim	Number of K/Delta overall randomizations, at least 1. Defaults to nsim; unused for lambda and D.
d_nsim	Random/Brownian replicates for D normalization (default 1000, at least 2), including when test = FALSE. Also used for overall D tests. Increase independently of nsim to reduce normalization Monte Carlo noise.

delta_control Named list of specialist Delta controls: model (ARD or ER), entropy (LSE, SE or GINI), mcmc_sim, thin, burn, lambda0 and proposal_sd. Defaults match fast_delta(). MCMC is not automatically shortened. reference_fit = "multistart" provides deterministic common ARD fitting. reference_unavailable = "conservative" explicitly treats only single-state reference discrepancies as upper-tail events; default strict handling retains NA. Both options affect the reference test, not observed Delta. The optional reference_fit = "profile" uses a deterministic ER log-rate grid and Brent search for the common reference only; it requires model = "ER". The default remains "nlminb". This experimental alternative changes Q and reference probabilities, and supports the exact independent endpoint. Observed local/overall estimators are unchanged. Ignored for other methods. reference_test = "nuisance" instead enables an experimental finite-sample nuisance-domain bound and requires test = TRUE and an explicit state_space (a vector, or a named list with every trait). In nuisance mode only, statistic = "between_clades" uses SD(log(local Delta)); the default is "local_global". nuisance_control$proposal_counts = TRUE adds a fixed half-weight count-universal sequential proposal to the unchanged Mk proposal mixture. Both options are experimental and disabled by default. The count proposal uses no observed state counts and does not change the common Mk null. nuisance_control sets gamma (0.001), tolerance (0.01), max_boxes (128), optional rate_bounds, fixed proposal_intensities (c(0.01, 0.1, 1, 10, Inf)), proposal_family ("auto"), boundary_power (24, allowed 8–30) and max_bank_bytes (64 MiB). correlation ("auto", "none", "taylor", "forest" or "transient") retains shared rate coordinates through finite three-state ARD Taylor models with bounded remainders. Auto also shares the maximum-forest stationary projection in mixing wide/unbounded boxes, with certified finite-branch errors. "taylor" requests nonpoint finite boxes; "forest" requests the mixing projection subject to size/error safeguards. "transient" shares the full transient near three-state multiclass boundaries, including shared slow-edge first-jump terms and a bounded Poisson remainder. Auto also considers this route when neither earlier model is used. Both require a small finite bound on the four remaining rates and a small bank. Auto/transient also use complete-domain probability-mass constraints when informative; their upper bound may still equal one. Subtree-repeat messages share indices across rectangles. Fixed verified witnesses may improve the maximization lower bound. Nuisance reference_model$search also retains raw bounds, final width, stop_reason, P_one_reason, gamma_clipped and bank_diagnostics. A certified final lower bound of one permits early termination; a large unresolved envelope is reported separately and never implies homogeneity. Other boxes retain the existing complete boundary envelope. The default plugin path is unchanged. These controls must be specified before inspecting the traits. ER supports 2–8 declared states and its complete nonnegative rate domain. ARD requires exactly three declared states; its complete unbounded domain is covered by forest/resolvent intervals with a uniform transition error enclosure. proposal_family = "auto" adds six fixed asymmetric components for ARD; "ER" retains the original proposal. More components can improve rare-state and fast-pair weights at a cost to ER efficiency. boundary_power = p gives a transition enclosure error of 32/2^p, not a P-value error bound. Finite ARD bounds are a 6-by-2 lower/upper matrix

in native rate order (B->A, C->A, A->B, C->B, A->C, B->C for state_space = c("A","B","C")). Their validity only covers that predeclared domain. `reference_fit` and `reference_unavailable` apply only to the default `reference_test = "plugin"`.

`clade_test` Calculate each crown's ordinary signal test (default FALSE). These probabilities do not test differences between crowns. D reuses the normalization draws; K/lambda/Delta use their specialist test conventions. Only estimable crowns meeting `min_tips` receive probabilities.

`clade_nsim` Number of K/Delta local randomizations, at least 1; defaults to `overall_nsim`. D uses `d_nsim`; lambda needs no simulation count.

`lambda_reference`
Fit the common lambda reference with pooled REML (default) or ML. Local and overall lambda estimates continue to use ML. REML reduces nuisance-fitting bias in tested middle-lambda scenarios but can be conservative near zero. Both reference tests are approximate.

Details

For K, dispersion is the unweighted sample standard deviation of log(K) across estimable clades. For lambda it is the sample standard deviation of lambda; D uses SD(D), and Delta uses SD(log(Delta)). These dispersion summaries are descriptive. Clade size, tree shape and trait-specific missingness are preserved in all references. Larger K/lambda/Delta and smaller D usually describe stronger phylogenetic clustering within a method. Report the observed difference and its scale even without a reference test; a reference P evaluates its stated model. Signal strength alone does not establish stabilizing selection, niche conservatism as a process, or a lower evolutionary rate.

Lambda uses a parametric bootstrap under a fitted shared lambda Gaussian model, with a separate profiled mean and diffusion variance for each crown. Location/scale invariance permits unit-variance, zero-mean simulations. The common lambda domain is the intersection of the crown-specific domains. The null is common lambda measured from each retained crown root, not common lambda on the complete tree. On an ultrametric tree with crown height h and ancestral depth s, whole-tree lambda in `[0,1]` induces the contrast-equivalent local parameter lambda*h/(h+(1-lambda)*s). Unequal crown ages can therefore give different local parameters under a common whole-tree model. Non-ultrametric crowns need not admit a scalar mapping; pruning that moves the crown root also changes the comparison scale. Predefine comparable biological scales and interpret local lambda on them. The common reference uses pooled REML by default; `lambda_reference = "ML"` retains the historical fit. Local and overall lambda remain ML estimates. The test is approximate because the common parameter is estimated. Historical ML calibration found 8% rejection in a middle-lambda scenario. Independent REML confirmation gave 5.2% there (1000 datasets, 199 replicates), 1.2% at zero and 4% at one (500 datasets each). Near-zero behaviour remains conservative. Treat this as a model diagnostic; exact nominal error control is not established.

D's experimental reference uses a shared-rate symmetric two-state ER Mk model with stationary root probabilities (1/2, 1/2), conditioned on the observed state count in each estimable crown. The common rate is fitted using the same conditional likelihood. Polynomial pruning and downward sampling preserve state counts without rejection. D normalization is shared between observed and simulated data; the test conditions on these Monte Carlo means. Minority coding and a fixed orientation preserve binary-label invariance. The positive rate domain starts at 1e-8 divided by the mean crown branch length; infinity is the independent-state endpoint. D testing supports up to 1000 observed tips per crown; its count probabilities require more work than linear pruning. The

ER reference is a new model-reference heterogeneity extension, not the original D signal test. D requires strictly positive, bifurcating crowns. Small or strongly imbalanced binary crowns can give unstable D normalization; the original D paper evaluated robustness on datasets with more than 50 species.

Delta estimates each crown with the same ACE, entropy and MCMC estimator as `fast_delta()`. Local state spaces drop unobserved states; state counts and frequencies retain original labels. Single-state crowns are unavailable, not assigned zero. With `overall = TRUE`, signed log(local/overall) ratios and their equal-crown root mean square are reported. The whole-tree estimate is calculated directly, not averaged from the crowns. Small crowns and differing state support may affect comparisons; the original Delta study recommends at least 20 species. Traditional and rank-normalized/folded R-hat, bulk/tail ESS, ratio MCSE and relative MCSE are retained for local and overall Delta. Ratio diagnostics enter reference-replicate warning counts.

Delta heterogeneity tests are outside the recommended workflow. Their implementation remains available for explicit experimental calls; the installed HETERO_NUISANCE_zh.md records the research controls. An explicitly requested plugin reference uses RMS(log(Delta_local/Delta_overall)), distinct from between-crown dispersion. One common ER/ARD Mk rate matrix is fitted on the original tree, integrating missing tips, with a uniform root mixture. Each complete-tree replicate preserves the observed NA masks and selected crowns, and refits all local and overall Delta estimators. The fitted parameters and root convention are in `reference_model`. Transition probabilities are cached by exact branch length in C++ (up to 64 MiB), and ACE workspaces are reused by retained tree/state count. Rates are reference-model parameters, not evidence of rate shifts. Analysis-owned ACE scratch is bounded by `options(fastphylosig.ace_cache_bytes = 64 * 1024^2)`; zero disables cross-fit caching. Active fits and allocator overhead are outside this bound. ARD Delta uses the corrected ACE downpass since 0.3.5. Local/overall estimates and experimental tests can differ from 0.3.4; ER is unchanged. The historical calibration includes random/missing and unequal crowns, 2–8 states and asymmetric models. High-state and sparse ARD reference inference remains exploratory; state scarcity can prevent a reference P. These checks do not establish general nominal error control. This approximate reference check has numerical tests and a bounded three-state whole-tree calibration: two ER scenarios gave 4.5% and 5% rejection (200 datasets each, 199 replicates, default MCMC). Before the ARD downpass correction, one asymmetric ARD scenario was conservative with 12.5% unavailable results; that result does not validate the corrected ARD estimator. In the historical runs, rare-state data can make all reference tests unavailable. These cases do not establish validity across tree shapes, state counts or missingness. `reference_fit="multistart"` optionally fits common ARD Q using the existing prepared ACE objective with deterministic scaled starts and continuation. Attempts and deviance gains are reported; it does not change observed Delta. This changes the reference Q, null draws, P and subsequent RNG when selected. `reference_unavailable="conservative"` optionally counts undefined single-state reference discrepancies in the upper tail, preserving every requested draw. It returns an explicitly labelled conservative upper tail; Delta itself remains undefined for those crowns. Actual calculation errors still make P unavailable. Defaults retain nlminb and strict failure handling. `P_heterogeneity_lower`/`P_heterogeneity_upper` describe the unknown reference tail contribution, not a confidence interval; they coincide when complete. All size-eligible crowns must be estimable; state-based exclusions are not silently fixed. Any failed replicate makes the reference probability unavailable. `nsim_MCMC_warning` reports replicates with MCMC flags. `nsim_optimizer_warning` counts optimizer warnings; `nsim_fit_warning` is its compatibility column. `nsim_numerical_recovery` counts ACE analytic recovery and `nsim_entropy_recovery` counts entropy roundoff recovery, separately from `nsim_other_warning`, `nsim_calculation_failure` (actual errors/invalid estimates) and `nsim_unavailable` (all unavailable replicates, including single-

state crowns). Counts flag a replicate if any local or overall fit is affected; categories can overlap and must not be summed. Local/overall results retain the corresponding flags and error stage. Unfinished default ARD fits within Delta receive bounded deterministic optimizer recovery. When used, optional `nsim_optimizer_retry` and `nsim_optimizer_recovered` count affected reference replicates, separately from remaining warnings; observed fits retain optional retry details. Healthy output structures and the MCMC path are retained. Recovery can change estimates and subsequent random streams. a finite estimate does not imply reliable MCMC. Inspect these alongside observed diagnostics before interpreting the reference probability.

Delta's opt-in nuisance test by default retains that discrepancy and the existing ACE/entropy/MCMC estimator. It declares the complete biological state space, including possible states absent at the tips, and a uniform root. One trait-independent mixture proposal bank is reused for every Q. Harrison's corrected importance rank includes the observation's weight. A likelihood-ratio confidence set with noncoverage at most `gamma` and Berger–Boos maximization handle common-Q uncertainty. Conservative C++ likelihood intervals cover continuous parameter boxes, including ER and three-state ARD infinite-rate boundaries. Row-sum constraints tighten pruning intervals. Finite ARD Taylor models share six coordinates across branches, states and reference likelihoods; signed tail/other contrasts retain numerator/denominator correlations. Nonlinear and rounding errors remain enclosed. Wide and unbounded mixing boxes can share their maximum- forest stationary projection across terminal branches. A coupling error encloses finite-branch transients; common Bernstein polynomials certify likelihood and weighted-tail bounds. Near multiclass boundaries, two common compact coordinates retain the whole transient of a two-edge generator; shared first-jump terms retain the four slow edges and only higher jump counts enter a Poisson remainder. Neither a spectral gap nor finite base rates are required. Whole-domain probability-mass constraints may further tighten the upper bound. Unsupported nonmixing boxes retain the previous enclosure. Fixed feasible witnesses improve only the supremum's lower bound. A budget limit retains the complete-domain upper bound rather than a candidate maximum. Finite-sample error control is per trait, unconditional over original inputs and Monte Carlo draws, and assumes the stated homogeneous model, fixed tree/missingness/crowns, preplanned domain/proposal and identical observed/reference estimator. It does not imply conditional control among successful runs or adequate power. Six-dimensional ARD may still have a large gap or return one; finite ARD domains must have an independent scientific justification. Single-state reference discrepancies enter the upper tail with Delta remaining NA; actual calculation failures retain NA and are never redrawn. `P_heterogeneity` is the conservative upper bound. `P_nuisance_lower`, `P_nuisance_upper` and `nuisance_gap` describe numerical maximization of the fixed-bank test, not confidence limits or binomial Monte Carlo error; `MCSE_P_heterogeneity` is NA. Reference diagnostics, domain, proposal, remaining boxes, timing and best-point weight/ESS diagnostics are retained in `reference_model`. `max_bank_bytes` guards the integer pattern bank, not total process memory. See the installed `HETERO_NUISANCE_zh.md` guide.

BM simulations have unit diffusion variance. K is invariant to trait location and scale, so neither ancestral means nor diffusion variance need to be fitted. For disjoint clades, independent simulations on their crown subtrees give the same joint distribution of K as one BM simulation on the complete tree: shared ancestral increments only add clade constants.

For the default plugin tests, the upper-tail probability is `(1 + sum(H_sim >= H)) / (nsim + 1)`. Its Monte Carlo error is `sqrt(nsim * q * (1 - q)) / (nsim + 1)`, where `q = sum(H_sim >= H) / nsim`; this is separate from Delta's MCMC error. It tests excess dispersion relative to the stated reference, not a universal null of equal signal under all evolutionary models. Other homogeneous processes can reject these references. BM intervals are reference envelopes, not confidence intervals for the observed K. Failed simulations make the probability unavailable rather than silently changing its

null. Nodes must be chosen without looking at trait values for calibration to apply. Adjust for multiple traits when interpreting multiple probabilities.

Tips outside selected clades, including small remnants of imbalanced trees, are reported in `selection$uncovered_tips`. No dense whole-tree distance or covariance matrix is constructed. Work depends on the sum of clade sizes, not just the size of the complete tree. Simulations run serially under R's RNG; use `set.seed()` for reproducibility. A practical initial scan uses size-bounded disjoint crowns, with `min_tips = 50` and `max_tips = 200` (use more than 50 observed tips for D). These are computational starting points, not calibrated statistical minima. Compare topology-selected bounds before inspecting trait estimates; wider bounds can improve coverage but change the scale of the question. Reuse `selection$nodes` across methods on the same source tree. Highly imbalanced trees may admit only one eligible crown; larger bounds cannot guarantee multiple disjoint crowns. Biologically specified nodes are useful when clade identity is the question, and must be non-overlapping for testing. Predefine the bounds and inspect coverage, retained counts and state support. Do not select a partition by its trait estimates or smallest P value. If multiple partitions are explored, report all as sensitivity analyses; their individual probabilities do not correct for that selection. See the installed `HETERO_GUIDE_zh.md` for a complete workflow.

Clade traversal, extraction, NA grouping, signal batches and calibration use C++ kernels. NA pruning retains ape's behavior. K tree factors and workspaces are reused within a simulation. D shares prepared contrast factors with `fast_d()`; lambda shares the likelihood evaluator and owned scratch buffers with `fast_lambda()`. D count probabilities and fitted transitions are reused across replicates; fixed-count D means need not be resimulated. D conditional likelihood is specific to this reference and is not interchangeable with the ordinary Mk likelihood of `fast_ace()`. Delta reuses `fast_ace()` likelihoods and `fast_delta()` entropy/MCMC kernels. K local tests reuse the scan's compiled NA groups and `fast_k()`'s bounded- memory C++ permutation path. The shared ACE tree validator runs in C++; its ancestral downpass defers independent row normalizations until needed, preserving repeated floating-point normalization and the Delta RNG stream. Scans and Delta tests share retained NA-mask workspaces. Local Delta randomizations shuffle only the crown data and skip unused trace diagnostics; observed estimates and common-Mk replicates retain their diagnostics. `simulation_chunk` is an upper bound: native batches are reduced to fit a 64 MiB budget for dominant block buffers, with at least one replicate per block. This is not a total memory limit; inputs, outputs and fixed tree arrays require additional RAM.

Value

A `fastphylosig_heterogeneity` list containing `clades` (one row per clade and trait), `heterogeneity` (one row per trait), `selection` (node IDs, descendant labels, coverage and overlap), and optionally `null` (reference discrepancy vectors). `clades` also reports optional local signal tests, state composition for D/Delta and local/overall contrasts when requested; `heterogeneity` separates descriptive `dispersion`, `local_global_deviation`, and the reference `test_statistic`/`statistic` with `P_heterogeneity`. D contrasts use local minus overall, without taking logarithms of D. `plot()` draws the clade estimates for one trait. `plot(fit, tree = tree)` colours the selected crowns on the supplied tree. `overall` contains complete-tree estimates and optional signal tests when requested. Plots annotate these separately from the dispersion statistic and its heterogeneity test. Plotting never computes tests or advances the RNG.

Examples

```
set.seed(4)
```

```
tree <- ape::rtree(80)
x <- setNames(rnorm(80), tree$tip.label)
fit <- fast_hetero(tree, x, min_tips = 10, max_tips = 30,
                   test = TRUE, nsim = 19, progress = FALSE)
fit$clades
fit$heterogeneity
```

`fast_k` *Fast Blomberg's K phylogenetic signal*

Description

Computes Blomberg's K using the validated tree-pruning production engine. This is the fixed-method specialist counterpart of `fast_signal(..., method = "K")`. The result class and shape are kept compatible with the former continuous `fast_signal()` API. Matching, trait-wise missing-value handling, and tree preparation follow the same validated path as the unified dispatcher.

Usage

```
fast_k(tree, x = NULL, test = FALSE, nsim = 1000, se = NULL,
  permutations = NULL, return_sim = FALSE, verbose = TRUE, ncores = 1,
  X = NULL, prepared = NULL, trait_chunk = 64, simulation_chunk = 128,
  keep_null = return_sim, progress = interactive(), data = NULL)
```

Arguments

`tree`	A connected `"phylo"` object with a unique structural root and finite, strictly positive branch lengths, or a context returned by `prepare_tree()`. A conventional degree-two root, binary topology, and ultrametricity are not required.
`x`, `X`, `data`	Named continuous trait vector or trait matrix/data.frame. The `data` spelling is an alias for `x`; supply only one spelling.
`test`	Whether to compute the permutation significance test.
`nsim`	Number of randomizations when `test = TRUE`.
`se`	Currently only `NULL` is supported.
`permutations`	Optional integer permutation matrix with exactly `nsim` rows and one column per retained tip.
`return_sim`	Whether to return simulated K values.
`verbose`	Whether to report species matching.
`ncores`	Number of OpenMP threads for randomization.
`prepared`	Optional object returned by `prepare_tree()`; when supplied it is used in place of `tree` after V2 integrity validation.
`trait_chunk`	Positive integer number of traits processed per C++ block.
`simulation_chunk`	Positive integer number of randomizations processed per streaming block.
`keep_null`	Whether to retain the full K null distribution.
`progress`	Whether to report method-level progress and elapsed runtime.

Value

For vector input, a phylosig-compatible result (an atomic numeric when test = FALSE); matrix/data.frame input returns one row per trait. Results include timing metadata and a small workflow attribute. Per-trait status and message/note fields identify invalid, insufficient, or undefined traits; an atomic no-test result exposes the same information as attributes.

See Also

fast_signal(), fast_lambda()

Examples

```
tree <- ape::rtree(20)
x <- stats::rnorm(20)
names(x) <- tree$tip.label
fast_k(tree, x, verbose = FALSE)
```

---

fast_lambda *Fast Pagel's lambda phylogenetic signal*

Description

Computes Pagel's lambda using the validated tree-pruning production engine. This is the fixed-method specialist counterpart of fast_signal(..., method = "lambda"). The result class and shape are kept compatible with the former continuous fast_signal() API. When a profile is requested, the grid runs from zero to the tree-specific allowed maximum. The nominal 95 percent profile interval uses max(logL) - 0.5 * qchisq(0.95, df = 1) as its cutoff; bounds are found by linear interpolation at threshold crossings and fall back to grid endpoints when a crossing is absent. The interval is therefore an approximate grid/profile interval, not an exact confidence interval.

Usage

```
fast_lambda(tree, x = NULL, test = FALSE, se = NULL,
  verbose = TRUE, ncores = 1, X = NULL,
  lambda_profile = NULL, lambda_profile_points = 101,
  prepared = NULL, trait_chunk = 64, progress = interactive(), data = NULL)
```

Arguments

tree	A connected "phylo" object with a unique structural root and finite, non-negative branch lengths, or a context returned by prepare_tree(). Zero terminal branches are not supported; zero internal branches are allowed with a numerical warning. A conventional degree-two root, binary topology, and ultrametricity are not required.
x, X, data	Named continuous trait vector or trait matrix/data.frame. The data spelling is an alias for x; supply only one spelling.
test	Whether to compute the likelihood-ratio test against lambda zero.

se	Currently only NULL is supported.
verbose	Whether to report species matching.
ncores	Accepted for API compatibility. The current fixed-lambda and optimization kernels run serially; increasing this value does not enable parallel lambda evaluation.
lambda_profile	Whether to store a lambda likelihood profile. The default stores it for vector input and skips it for batch input.
lambda_profile_points	Number of regular profile grid points (at least 5). The fitted lambda, and lambda 1 when allowed by the tree, are inserted in addition to the regular grid.
prepared	Optional object returned by prepare_tree(); when supplied it is used in place of tree after V2 integrity validation.
trait_chunk	Positive integer number of traits processed per C++ block.
progress	Whether to report method-level progress and elapsed runtime.

Value

For vector input, a phylosig-compatible result; matrix/data.frame input returns one row per trait. Results include timing metadata and a small workflow attribute. Per-trait status and message/note fields distinguish valid fits from invalid, insufficient, or undefined trait-level results.

See Also

fast_signal(), fast_k()

Examples

```
tree <- ape::rtree(20)
x <- stats::rnorm(20)
names(x) <- tree$tip.label
fast_lambda(tree, x, verbose = FALSE)
```

---

fast_signal *Unified fast phylogenetic signal dispatcher*

Description

Provides the one-command entry point for phylogenetic signal analysis and dispatches to exactly one of the four public production functions: fast_k(), fast_lambda(), fast_d(), or fast_delta(). The selected function owns validation, progress output, runtime timing, numerical calculation, and result construction. Method aliases are normalized to "K", "lambda", "D", or "Delta". Ordinary calls perform tree inspection, conservative representation handling, explicit species matching, and trait-wise missing- value grouping before the selected production kernel runs. No biological tree repair is inferred from the method or trait type.

Usage

```
fast_signal(tree, data = NULL, method = NULL, ..., x = NULL, X = NULL,
  prepared = NULL, trait_chunk = 64)
```

Arguments

tree	A "phylo" object or a context returned by prepare_tree().
x, data, X	Named trait vector or trait matrix/data.frame. The data spelling is accepted by the unified dispatcher; x and X are retained as compatibility aliases. Matrix and data.frame row names must be species names. Supply only one spelling.
method	Exactly one method selector. Supported aliases are K/k, lambda/Lambda, the Unicode U+03BB alias, D/d, and Delta/delta. The method is explicit and is never inferred from the trait type; omitting it or supplying an unknown selector is an error.
prepared, trait_chunk	Compatibility formals for prepared tree contexts and C++ trait blocks; prepared contexts undergo V2 integrity validation, and the formals are forwarded only to targets that support them.
...	Arguments for the selected production function. K and lambda accept the arguments documented by fast_k() and fast_lambda(); D and Delta accept those documented by fast_d() and fast_delta(), respectively. Unknown or inapplicable arguments are reported as errors. Legacy positional arguments after method remain supported. When data, x, and X are supplied more than once, the call fails rather than choosing one silently.

Details

For a first analysis, supply a tree, named trait data, and one method selector to this function. Advanced users can call fast_k(), fast_lambda(), fast_d(), or fast_delta() directly when they need method-specific simulation, profile, null-retention, MCMC, or diagnostic controls. fast_ace() is the specialist ancestral-state ML function used by the Delta workflow.

Value

The original result class and shape returned by the selected production function (for example phylosig, phylo.d, or phylo_delta). A small workflow attribute records the canonical method and production function. Lambda results may include an approximate profile and nominal 95 percent interval: the profile uses a regular grid from zero to the tree-specific allowed maximum, with the fitted value (and one when allowed) inserted. Its cutoff is max(logL) - 0.5 * qchisq(0.95, df = 1); limits are estimated by linear interpolation at profile threshold crossings, or by grid endpoints when no crossing is available. This is a grid/profile approximation rather than an exact confidence interval.

See Also

fast_k(), fast_lambda(), fast_d(), fast_delta(), fast_ace(), check_tree(), match_tree_data(), prepare_tree()

Examples

```
set.seed(2)
tree <- ape::rtree(30)
X <- matrix(stats::rnorm(30 * 3), 30, 3,
            dimnames = list(tree$tip.label, paste0("trait_", 1:3)))
fit <- fast_signal(tree, data = X, method = "K", test = FALSE,
                   verbose = FALSE)
summary(fit)
fast_signal(tree, data = X[, 1], method = "lambda", verbose = FALSE)
```

---

match_phylo_data *Deprecated alias for* match_tree_data

Description

This compatibility alias is deprecated. Use `match_tree_data()` instead. The function returns the same object as `match_tree_data()` and emits a deprecation warning when called.

Usage

```
match_phylo_data(tree, X, prune = TRUE, verbose = TRUE)
```

Arguments

`tree`	An object of class `"phylo"` or a prepared tree context.
`X`	Named vector, matrix, or data.frame. Species names must be supplied as names or row names unless rows are already in `tree$tip.label` order.
`prune`	Whether to drop tree tips that are absent from `X`.
`verbose`	Whether to print a short matching message.

Value

See match_tree_data.

Examples

```
set.seed(6)
tree <- ape::rtree(10)
X <- data.frame(trait = stats::rnorm(10), row.names = tree$tip.label)
# This compatibility alias is deprecated; prefer match_tree_data().
matched <- suppressWarnings(match_phylo_data(tree, X, verbose = FALSE))
matched$report
```

match_tree_data *Match a phylogeny and trait table*

Description

Matches a phylogeny and trait table without silently discarding information. The returned object contains the matched tree, the trait table reordered in tree-tip order, and a report listing matched species, removed tree tips, and removed data rows.

Usage

```
match_tree_data(tree, data = NULL, prune = TRUE, verbose = TRUE, X = NULL)
```

Arguments

tree	An object of class "phylo" or a prepared tree context.
data	Named vector, matrix, or data.frame. Species names must be supplied as names or row names unless rows are already in tree$tip.label order.
X	Deprecated compatibility alias for data. Supply only one of data and X.
prune	Whether to drop tree tips that are absent from X.
verbose	Whether to print a short matching message.

Value

A list with elements tree, X, data, report, matched_species, tree_tips_removed, and data_rows_removed. data is an alias of X for the recommended check-resolve-match-calculate workflow.

Examples

```
set.seed(6)
tree <- ape::rtree(10)
X <- data.frame(trait = stats::rnorm(10), row.names = tree$tip.label)
matched <- match_tree_data(tree, data = X, verbose = FALSE)
matched$report
```

plot.fastphylosig_heterogeneity *Plot clade-scale signal estimates*

Description

Plot clade-scale signal estimates

Usage

```
## S3 method for class 'fastphylosig_heterogeneity'
plot(
  x,
  trait = NULL,
  ...,
  tree = NULL,
  view = if (is.null(tree)) "clades" else "tree",
  color_scale = c("auto", "raw", "log"),
  palette = NULL,
  na_col = "grey80",
  show_legend = TRUE,
  node_labels = FALSE,
  show_summary = TRUE
)
```

Arguments

x	A result from ‘fast_hetero()‘.
trait	Trait column name. Required for multiple traits.
...	Further arguments passed to ‘graphics::plot()‘ (clade view) or ‘ape::plot.phylo()‘ (tree view). For example ‘type = "fan"‘ selects a fan.
tree	The original phylo tree, or prepared context, used in the scan. Required for the tree view. Selected nodes and their descendant labels are checked against the result. Supply the original tree, with its branch lengths.
view	Either ‘"clades"‘ or ‘"tree"‘. Defaults to ‘"tree"‘ when ‘tree‘ is supplied, otherwise ‘"clades"‘.
color_scale	Tree colour scale: ‘"auto"‘ uses log(K/Delta) or raw lambda/D; ‘"raw"‘ and ‘"log"‘ select an explicit scale. Log colours require positive estimates. Legend labels always show original estimates.
palette	Optional vector of at least two colours for the tree view.
na_col	Colour for unselected or unestimable crowns.
show_legend	Draw a colour legend in the tree view.
node_labels	Label selected crown roots with original node numbers.
show_summary	Annotate complete-tree signal/test results (when calculated), and clade dispersion/heterogeneity test results. Tests not performed or unavailable are explicitly identified. No tests are run while plotting.

Value

Invisibly, the clade table for the clade view, or a list with the plotting tree, source-to-plot node mapping, clade table, owner indices and colours for the tree view. BM bars are reference envelopes, not confidence intervals. Tree colours represent crown-clade estimates, not branch-specific estimates. More deeply nested selected crowns take precedence, including unavailable estimates. Crown stems retain the enclosing crown colour.

plot_signal *Plot fastphylosig results*

Description

Creates lightweight base R diagnostic plots for supported fastphylosig result objects. For K and Delta, the x axis is the final phylogenetic-signal value and the density/ridge shows simulated signal values from the permutation test; the right-tail region `simulated >= observed` is shaded to show the P-value tail. For single-trait D results, `null = "auto"` draws a Fritz & Purvis D calibration plot: Brownian threshold and random association null distributions are scaled onto the D axis and shown together as overlaid mountain densities on the same baseline, the observed D is drawn as a vertical line, and reference lines mark D = 0 (Brownian expectation) and D = 1 (random expectation). The internal simulated contrast sums are converted back to D values before plotting. `P_random` and `P_Brownian` are labelled separately; zero extreme counts are displayed as `P < 1/nsim` rather than `P = 0`. For lambda, which uses a deterministic likelihood-ratio test rather than random simulations, the helper draws the stored profile log-likelihood curve when available, marks `lambda_hat`, `lambda = 0`, `lambda = 1`, and an approximate 95 percent profile-likelihood confidence interval. The confidence interval is the region with log-likelihood at least `max(logL) - 0.5 * qchisq(0.95, df = 1)`. Old lambda results without a stored profile fall back to a compact likelihood-ratio summary. Observed signal values, P values, and simulation counts are printed on the plots where applicable.

Usage

```
plot_signal(result, p_col = NULL, alpha = 0.05, main = NULL,
  xlab = NULL, ylab = NULL,
  null = c("auto", "random", "brownian"),
  label = c("none", "significant", "top", "all"), top_n = 10,
  ridge_scale = 0.85, col = NULL, pch = 19, cex = 0.7, ...)
```

Arguments

`result`	Result produced by `fast_signal`, `fast_k`, `fast_lambda`, `fast_d`, or `fast_delta`.
`p_col`	Optional P-value column for data.frame results. For D results the default selects the fitted random-null probability. Recognized canonical and legacy aliases select their corresponding null under `null = "auto"`. An explicitly present alias is honored even if another alias differs. A custom table column is retained under `null = "auto"`; an explicit `null` takes precedence over a P-value column for the other null.
`alpha`	Significance level used for coloring observed signal lines.
`null`	For D calibration plots, which null simulation to plot. `"auto"` draws both the random and Brownian D null distributions for a single-trait `fast_d()` result.
`label`	Which observed traits to label in multi-trait ridge plots.
`top_n`	Number of smallest-P traits to label when `label = "top"`.
`ridge_scale`	Vertical scaling of each density ridge in multi-trait plots.
`main, xlab, ylab, col, pch, cex, ...`	Base plotting arguments.

Value

Invisibly returns the normalized plotting data. K and Delta plots include sim, null_model, n_sim, and tail_direction. Single-trait D overlay plots include sim_random, sim_brownian, Pval1, Pval0, n_sim, plot_type = "D_calibration", P_random_display, and P_Brownian_display. Lambda plots include plot_type = "lambda_profile" when a stored profile is available, plus profile, ci_lower, ci_upper, and LR. Old lambda results without a profile return plot_type = "lambda_lr_summary".

Examples

```
set.seed(7)
tree <- ape::rtree(20)
x <- stats::rnorm(20)
names(x) <- tree$tip.label
k <- fast_signal(tree, data = x, method = "K", test = TRUE, nsim = 20,
                 return_sim = TRUE, verbose = FALSE)
plot_signal(k)
```

---

prepare_tree *Prepare a phylogenetic tree for repeated signal calculations*

Description

Builds a reusable tree context containing linear-size topology and traversal data. Dense covariance matrices, Cholesky factors, lambda spectral data, and trait-specific numerical subsets are cached lazily when first requested. Every external reuse validates the V2 context schema and exact protected snapshot before cached or numerical work.

Usage

```
prepare_tree(tree, max_cached_subsets = 16L,
  cache_budget = 512 * 1024^2)
```

Arguments

tree	A rooted object of class "phylo" with finite, non-negative branch lengths.
max_cached_subsets	Maximum number of full or NA-pruned subset entries retained in the structural cache, including the full-tree entry. Retained for compatibility; dense memory is controlled separately by cache_budget.
cache_budget	Maximum number of bytes retained in the numerical LRU cache. A value of zero permits transient numerical calculations without retaining their dense payloads.

Details

The tree context does not alter the tree or raw trait data. Initial preparation stores a snapshot; later changes to the original phylo object do not update it. On reuse, a versioned exact protected snapshot covers the cached tree's tip labels, edge matrix, branch lengths, and Nnode. Changes to those fields inside the context are rejected before cache or numerical work, while node.label and other non-computational attributes are outside the integrity contract; node.label may be reordered when internal nodes are renumbered. V2 contexts can be saved with saveRDS() and reused after readRDS(); missing, legacy, or unknown context schemas must be rebuilt with prepare_tree(). Initial preparation stores only topology, tip mappings, and traversal data. Public production K and lambda use the tree-pruning engines and do not request dense covariance, Cholesky, or spectral payloads; D, Delta, and ACE do not request them either. The bounded numerical cache remains available to internal validation paths. Results retain the statistical formulas of the corresponding reference functions. See cache_info for cache diagnostics.

Value

An object of class c("fastphylosig_tree", "list"). Pass it as the tree argument to fast_signal(), fast_k(), fast_lambda(), fast_d(), and fast_delta(). For ACE, use fast_ace(x, prepared = ctx). The unified fast_signal() entry point still requires an explicit method.

Examples

```
set.seed(1)
tree <- ape::rtree(20)
ctx <- prepare_tree(tree)
X <- matrix(stats::rnorm(40), nrow = 20,
            dimnames = list(tree$tip.label, c("trait_1", "trait_2")))
fast_signal(ctx, data = X, method = "K", verbose = FALSE)
```

---

resolve_tree *Resolve a phylogenetic tree conservatively*

Description

Checks a tree with check_tree() and makes only safe representation normalizations (edge ordering and internal node numbering) under Canonicalization Contract V2. Tip IDs and labels remain unchanged; the root is n_tip + 1, and remaining ordering is derived from exact UTF-8 tip-label bytes rather than locale collation, source internal-node IDs, or source edge-row order. An unrooted tree is rooted only when the selected signal requires it and an explicit valid outgroup is supplied. The function does not midpoint-root, infer an outgroup, resolve polytomies, collapse unary nodes, modify branch lengths, jitter zero branches, or drop tips. Remaining blocking conditions produce an explicit USER_ACTION_REQUIRED error.

Usage

```
resolve_tree(tree, signal, outgroup = NULL)
```

Arguments

tree	An object of class "phylo".
signal	One of "K", "lambda", "D", or "Delta"; matching is case-insensitive and exactly one value is required.
outgroup	Optional character vector of one or more existing tip labels. It is used only when the selected signal requires a rooted tree.

Details

After every representation or rooting change the tree is checked again. A successful result is idempotent: resolving it again does not change its topology, tip labels, or branch-length paths.

Value

A "phylo" object with class "fastphylosig_resolved_tree" and an attribute named "fastphylosig_resolution". The attribute records the signal, original and final readiness statuses, representation changes, and a logical ready flag.

See Also

check_tree, prepare_tree

Examples

```
tree <- ape::rtree(8)
tree <- resolve_tree(tree, signal = "K")
attr(tree, "fastphylosig_resolution")
```

---

result_methods *Stable result classes and display methods*

Description

The public calculation functions retain their historical fields and legacy classes while adding a common result class. Scalar results inherit from fastphylosig_result after a statistic-specific class (fastphylosig_signal, fastphylosig_d, or fastphylosig_delta) and before the corresponding legacy class. Matrix and data-frame results inherit from c("fastphylosig_table", "fastphylosig_result", "data.frame").

Usage

```
## S3 method for class 'fastphylosig_result'
print(x, ..., digits = getOption("digits"))
## S3 method for class 'fastphylosig_result'
summary(object, ...)
## S3 method for class 'fastphylosig_result'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

x	A scalar or table result returned by one of the fastphylosig calculation functions.
object	A result supplied to summary().
row.names	Optional row names for the data-frame representation.
optional	Logical flag accepted for compatibility with the base as.data.frame() generic.
digits	Number of significant digits used by print().
...	Additional arguments passed to the underlying data-frame printer.

Details

Canonical aliases are additive. K results expose estimate, p_value, p_mcse, n_sim_requested, and n_sim_successful; lambda results expose estimate, logLik, LR, and p_value. D results keep the random and Brownian P values and Monte Carlo errors separate (P_random, P_Brownian, MCSE_P_random, and MCSE_P_Brownian); no combined p_value is created. Delta results expose estimate, p_value, p_mcse, estimate_mcse, chain-specific ESS/R-hat aliases, and simulation or iteration count aliases.

The no-test K scalar remains an atomic numeric with its legacy phylosig class. The display methods derive its estimate directly and do not coerce it to a list. Printing, summarising, and coercing results to a data frame use a compact canonical schema and omit retained simulation vectors.

Value

print() returns the input invisibly. summary() and as.data.frame() return a compact data frame; the latter has one row for a scalar result and preserves all non-simulation columns for a result table.

See Also

fast_signal, fast_k, fast_lambda, fast_d, fast_delta

Index

as.data.frame.fastphylosig_result
(result_methods), 29
as.data.frame.fastphylosig_table
(result_methods), 29

cache_info, *3*, 3, *28*
check_tree, *3*, 4

fast_ace, *3*, 5
fast_d, *3*, 7, *26*
fast_delta, *3*, 9, *26*
fast_hetero, *3*, 12
fast_k, *3*, 19, *26*
fast_lambda, *3*, 20, *26*
fast_signal, *3*, 21, *26*
fastphylosig (fastphylosig-package), 2
fastphylosig-package, 2

match_phylo_data, 23
match_tree_data, *3*, *23*, 24

plot.fastphylosig_heterogeneity, 24
plot_signal, *3*, 26
prepare_tree, *3*, 27
print.fastphylosig_result
(result_methods), 29
print.fastphylosig_table
(result_methods), 29

resolve_tree, *3*, 28
result_methods, 29

summary.fastphylosig_result
(result_methods), 29
summary.fastphylosig_table
(result_methods), 29