Package 'emphasis'

November 18, 2026

Type Package

Title Monte Carlo Inference for Phylogenetic Diversification Models

Version 1.1.0

Description Fits birth-death models of species diversification in which the speciation and extinction rates depend on covariates generated by the process itself, such as the number of lineages, the age structure of the lineages and their evolutionary distinctiveness, to reconstructed phylogenies that contain only extant species. The likelihood is integrated over the unobserved extinct lineages by importance sampling on augmented trees, and the parameters are estimated by Monte Carlo expectation-maximization, by the cross-entropy method, or from a generalized additive model fitted to a simulated likelihood surface, following Richter, Haegeman, Etienne and Wit (2020) <doi:10.1111/stan.12205>. Trees can be simulated under the same models, and incomplete taxon sampling is supported.

License MIT + file LICENSE

URL https://franciscorichter.github.io/emphasis/, https://github.com/franciscorichter/emphasis

BugReports https://github.com/franciscorichter/emphasis/issues

Encoding UTF-8

Language en-GB

Depends R (>= 3.5.0)

Imports ape, DDD, graphics, grDevices, mgcv, nloptr, parallel, progress, Rcpp, RcppParallel, stats, utils

Suggests dglars, nnet, testthat (>= 3.0.0), TreeSim

LinkingTo BH, nloptr, Rcpp, RcppParallel

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Francisco Richter [aut, cre],
Thijs Janzen [ctb] (earlier C++ implementation),
Hanno Hildenbrandt [ctb] (earlier C++ implementation)

Maintainer Francisco Richter <franciscorichter@gmail.com>

Repository CRAN

Date/Publication 2026-11-18 21:50:02 UTC

Contents

auto_bounds 2
compare_models 4
covariate_path 6
diagnose_cem 7
diagnose_gam 9
diagnose_mcem 10
emphasis_pipeline 12
estimate_rates 14
lineage_table 19
predict.emphasis_surface 19
predict_survival 20
print.cem_diagnostics 21
print.emphasis_fit 21
print.emphasis_fits 22
print.emphasis_pipeline 22
print.emphasis_surface 23
print.gam_diagnostics 23
print.mcem_diagnostics 24
simulate_tree 24
survival_mc 28
survival_surface 30
train_GAM 32

Index **34**

`auto_bounds` *Automatically determine parameter bounds via forward simulation*

Description

Starts from a safe centre point (low rates, zero covariates) and expands outward along each parameter axis using bisection to find the feasibility boundary — where trees transition from surviving with sensible size to going extinct or exploding. Much more efficient than blanket LHS sampling over a wide box. Optionally trains a survival GAM over the detected region; the bounds do not depend on it.

Usage

```
auto_bounds(
  tree,
  model = "cr",
  link = "exponential",
  n_test = 5L,
  bisect_steps = 8L,
  margin = 0.5,
  tip_range = c(0.1, 10),
  train_surv_gam = TRUE,
  num_threads = 1L,
  verbose = TRUE,
  rho = 1,
  surv_tries = 20L
)
```

Arguments

`tree`	A `phylo` object or anything accepted by `estimate_rates`.
`model`	Model specification: `"cr"`, `"dd"`, etc.
`link`	Link function: `"exponential"` (default), `rate = exp(eta)`, under which the intercepts `beta_0` and `gamma_0` are log rates; `"linear"`, `rate = max(0, eta)`, under which the parameters are rates and a covariate can drive a rate to zero, where it is clipped.
`n_test`	Number of replicate simulations per candidate point to assess feasibility (default 5).
`bisect_steps`	Number of bisection steps per axis direction (default 8; gives ~1/256 resolution).
`margin`	Fractional expansion of detected bounds (default 0.5).
`tip_range`	Multiplier range for acceptable tip counts relative to the observed tree. Default `c(0.1, 10)`.
`train_surv_gam`	If `TRUE` (default), train a survival GAM on the detected region, from 2000 single forward draws. The GAM is not accurate enough to condition a fit on (experiment E43; see `train_GAM`).
`num_threads`	Parallel threads for batch simulation (default 1).
`verbose`	Print progress (default `TRUE`).
`rho`	Sampling fraction (0, 1]. Default 1 (complete sampling).
`surv_tries`	Draws allowed per simulation, the first included, when judging whether a parameter vector produces surviving clades of a usable size: an extinct clade is redrawn up to this many times in all (passed to `simulate_tree` as `max_tries`). Default 20.

Value

A list with components:

lower_bound Numeric vector of lower bounds (compact layout).

upper_bound Numeric vector of upper bounds (compact layout).

survival_gam A gam object if trained, else NULL.

center The safe centre point used as starting location.

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
# A coarse search without the survival GAM
ab <- auto_bounds(phy, model = "cr", n_test = 3L, bisect_steps = 3L,
                  train_surv_gam = FALSE, verbose = FALSE)
ab$lower_bound
ab$upper_bound

# The default search, which also trains the survival GAM (see train_GAM()
# for why a fit should not be conditioned on it)
ab2 <- auto_bounds(phy, model = "cr", verbose = FALSE)
predict_survival(ab2$survival_gam, as.data.frame(t(ab2$center)))
```

---

compare_models *Compare fitted models by AIC, carrying the Monte Carlo handicap*

Description

Ranks two or more fits from estimate_rates by AIC. Each fit's log-likelihood is a log of a Monte Carlo mean, so it sits below the log-likelihood it estimates; how far below grows as that fit's own effective sample falls, and the amounts do not cancel between models. A model that is harder to sample is therefore penalised for being harder to sample, and the AIC difference between two fits can be smaller than the difference in their biases.

Usage

```
compare_models(..., margin = 2)
```

Arguments

...	Two or more emphasis_fit objects. Names become the row labels; unnamed arguments are labelled from the fitted model.
margin	AIC units within which two models are not told apart. Default 2.

Details

The returned table carries ESS and loglik_gap per fit and AIC_swing, twice the gap relative to the leader's, which is how much of the AIC difference the bias alone could account for.

ranking_at_risk names the models that could overtake the leader once both biases are corrected. A model qualifies only if its own gap is larger than the leader's — so that correcting both closes the distance — and then either the swing reaches its delta_AIC or its draw is heavy enough that the swing understates the gap. A warning is issued with it.

Two things follow that a caller should not have to infer. The attribute is about the ORDER, not about precision: when the leader is itself the heavily-handicapped fit, correcting the biases only widens its lead, so nothing is named and nothing is wrong with that. And it covers bias, not variance. A fit can lead on a log-likelihood estimated from a handful of effective draws — measured: a winner with an effective sample of 6.5 of 300 — and be named by nothing here, because its *expected* position is safe while its realised one is noisy. loglik_se and ESS are the columns for that, and they are in the table to be read.

The row with the lowest AIC is not the model the table chooses. Among nested models the richer one has the lower AIC whenever its two extra parameters raise the log-likelihood by more than 2, which they do on a share of trees generated without them: with two extra parameters the likelihood-ratio statistic is chi-squared on two degrees of freedom under the null, so the richer model takes the minimum on about 13% of null trees, and with three richer models offered beside the null, on more. Measured, on 20 trees a cell from a diversity-dependent generator at 50 to 200 tips: a lineage-level model had the minimum AIC on 2 to 7 of 20, and led by more than 2 AIC on 0 to 2 of 20 (E27). The chosen attribute is therefore the model with the fewest parameters among those within margin AIC units of the minimum, the one with the lower AIC when several share that number of parameters, and within_margin marks those rows; the full ranking is kept so that the margin can be read.

A warning names the chosen model when its own effective sample is below 10 draws: its log-likelihood is then an estimate from a handful of histories and the margin it rests on is noise as much as evidence. The ranking_at_risk attribute covers the bias of unequal precision; this covers its variance.

Value

A data frame, one row per fit, with columns model, n_pars, loglik, AIC, loglik_se, ESS, loglik_gap, gap_heavy, delta_AIC, AICw, AIC_swing and within_margin, ordered by AIC, with attributes chosen (the most parsimonious model within margin of the minimum), chosen_starved (whether that model's effective sample is below 10) and ranking_at_risk.

See Also

estimate_rates

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
ctrl <- list(max_iter = 5, num_trees = 30)
fit_cr <- estimate_rates(phy, method = "mcem", model = "cr",
  control = c(ctrl, list(lower_bound = c(-3, -5), upper_bound = c(1, 0))))
```

```
fit_dd <- estimate_rates(phy, method = "mcem", model = "dd",
  control = c(ctrl, list(lower_bound = c(-3, -0.1, -5, -0.1),
                         upper_bound = c(1, 0.1, 0, 0.1))))
compare_models(cr = fit_cr, dd = fit_dd)
```

`covariate_path` *Selection along a path over the covariates*

Description

Instead of comparing nested models one pair at a time, this traces the differential-geometric LARS path of Augugliaro, Mineo and Wit (2013) over all the covariates a rate can read — `N`, `D` and `ED` — on the same augmented trees an M-step maximises over. Under the exponential link the complete-data log-likelihood is a log-linear point process: one row per lineage and segment between consecutive events, with `N`, `D` and `K` at their values on the segment (each is constant between events), `ED` at the segment's midpoint, the segment's length as exposure, and a count of one when that lineage's own event ends the segment. The speciation and extinction halves separate, so each rate gets its own path.

Usage

```
covariate_path(
  trees,
  log_weights = NULL,
  n_trees = 40L,
  covariates = c("N", "D", "ED"),
  rate = c("speciation", "extinction"),
  criterion = c("BIC", "AIC"),
  n_gamma = 60L
)
```

Arguments

`trees`	A list of augmented-tree data frames with the topology, as `augment_trees` returns them with parent ids, or a single `phylo` taken as fully observed.
`log_weights`	Optional log importance weights, one per tree; without them every tree weighs the same.
`n_trees`	How many trees to use: the `n_trees` of largest weight (the first `n_trees` without weights), renormalised.
`covariates`	Which covariates to offer the path; any of `"N"`, `"D"`, `"ED"`, `"EDc"` (ED centred on the alive lineages), `"K"` (the number of speciation events on the path from the crown).
`rate`	`"speciation"`, `"extinction"` or both.
`criterion`	`"BIC"` or `"AIC"` along the path.
`n_gamma`	Number of points on the path.

Details

The path is the curve on which the Rao score statistics of the active covariates are equal in size and the inactive ones smaller, $|r_j(\beta)| = \gamma$ for j active, traced from the γ at which the first covariate enters down to zero, with the intercept always free. It is solved here on a grid of γ by Newton steps rather than by **dglars**'s predictor–corrector, because the point process needs a true exposure offset, which that package does not take; on a constant-exposure problem the two agree (`tests/testthat/test-lineage-table.R`). The order of entry says which covariates the data ask for first; AIC and BIC along the path pick a point, with BIC's sample size the number of events.

Importance weights enter the likelihood as row weights, one per tree and normalised to sum to one, so the objective is the expected complete-data log-likelihood of one tree — the M-step's Q function — and the sample size behind BIC is the weighted mean number of events per tree.

Value

A list with one element per rate: the path (a data frame with one row per γ: the coefficients, the log-likelihood, the active set size, AIC and BIC), the entry order of the covariates (the γ at which each enters), the coefficients and active set at the criterion's optimum, and the frame's size. A rate with no event in the trees (the extinction rate of a fully observed tree) gets an empty path: `path` is `NULL`, `active` and `entry_order` are empty and `n_events` is 0.

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
cp <- covariate_path(phy, rate = "speciation")
cp$speciation$entry_order
```

`diagnose_cem` *Diagnostics for a CEM fit*

Description

Computes convergence, importance-sampling quality, and population-diversity diagnostics from the output of `emphasis_cem` or `estimate_rates` (method `"cem"`).

Usage

```
diagnose_cem(
  x,
  plot = TRUE,
  lower_bound = NULL,
  upper_bound = NULL,
  true_pars = NULL
)
```

Arguments

x	An emphasis_fit object (from estimate_rates with method = "cem") or the raw list from emphasis_cem.
plot	Logical; if TRUE (default), produce diagnostic plots.
lower_bound, upper_bound	Optional numeric vectors of the same length as the parameter vector. When provided, dashed horizontal lines are drawn at the lower (orange) and upper (purple) bounds in each parameter-trace panel, making it easy to see whether the estimate has converged away from the boundary.
true_pars	Optional named or unnamed numeric vector of true parameter values. Useful in simulation studies where the generative parameters are known. When provided, a red dashed line is added to each parameter-trace panel at the true value. In real-data analyses this argument should be omitted (the true parameters are unknown).

Details

The function works on the details component of an emphasis_fit object or directly on the list returned by emphasis_cem. It requires that the fit was produced with the current version of the package (which stores history, final_pop, and best_IS in the output).

Value

A list of class "cem_diagnostics" (invisibly) with components convergence, population_spread, IS_quality, final_pop, and best_IS.

Convergence ($convergence)

A data frame with one row per iteration containing:

iteration Iteration index.

best_loglik IS log-likelihood of the best particle.

n_valid Number of particles with a finite fhat.

rej_lambda Trees rejected due to max_lambda exceedance.

rej_overruns Trees rejected due to max_missing exceedance.

Population spread ($population_spread)

Per-iteration summary of the fhat distribution across all valid particles: median, interquartile range, and total range. A narrowing range indicates population convergence.

IS quality ($IS_quality)

Importance-sampling diagnostics computed from the final evaluation of the globally best parameter vector (with max(sample_size, n_boot, 1) augmented trees):

ESS Effective sample size: $(\sum w_i)^2 / \sum w_i^2$.

`ESS_fraction` ESS as a fraction of `n_trees`.

`mean_lw`, `sd_lw` Mean and SD of IS log-weights.

`fhat` IS log-likelihood estimate.

`n_trees`, `n_rejected` Trees simulated and rejected.

Final population (`$final_pop`)

The full particle population at the last completed iteration (before resampling), with columns for parameters and `fhat` per particle. Useful for inspecting the distribution of surviving particles.

Raw IS data (`$best_IS`)

The raw IS output for the best particle: `logf`, `log_q`, `lw` (log-weights), and `trees` (augmented tree data frames in C++ format).

See Also

`emphasis_cem`, `estimate_rates`

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
# Exponential link (the default): the intercepts are log rates
lb <- c(-3, -5)
ub <- c(1, 0)
fit <- estimate_rates(phy, method = "cem", model = "cr",
  control = list(lower_bound = lb, upper_bound = ub,
                 max_iter = 5, num_particles = 20))
diag <- diagnose_cem(fit, lower_bound = lb, upper_bound = ub,
                     true_pars = log(c(0.5, 0.1)))
diag$convergence
```

---

diagnose_gam *Diagnostics for a GAM-based MLE fit*

Description

Produces diagnostic plots and returns summary statistics from a `estimate_rates` fit obtained with `method = "gam"`.

Usage

```
diagnose_gam(x, plot = TRUE, true_pars = NULL)
```

Arguments

x	An emphasis_fit object from estimate_rates with method = "gam".
plot	Logical; if TRUE (default), produce diagnostic plots.
true_pars	Optional numeric vector of true parameter values (for simulation studies).

Value

A list of class "gam_diagnostics" (invisibly) with components surface_summary, gam_summary, and mle.

Surface coverage ($surface_summary)

Summary of the IS log-likelihood grid: total grid points, number with finite fhat, fhat range, and mean trees per grid point.

GAM quality ($gam_summary)

Key statistics from the fitted GAM: deviance explained, GCV score, and effective degrees of freedom.

MLE ($mle)

The MLE parameter estimates and predicted log-likelihood at the optimum.

See Also

diagnose_cem, diagnose_mcem, estimate_rates

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
fit <- estimate_rates(phy, method = "gam", model = "cr",
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 grid_points = 8, sample_size = 20))
diagnose_gam(fit)
```

---

diagnose_mcem *Diagnostics for an MCEM fit*

Description

Computes convergence and importance-sampling quality diagnostics from the output of estimate_rates with method = "mcem".

Usage

```
diagnose_mcem(
  x,
  plot = TRUE,
  lower_bound = NULL,
  upper_bound = NULL,
  true_pars = NULL
)
```

Arguments

x	An emphasis_fit object (from estimate_rates with method = "mcem") or the raw list from .mcem_dynamic_fresh.
plot	Logical; if TRUE (default), produce diagnostic plots.
lower_bound, upper_bound	Optional numeric vectors for bound lines.
true_pars	Optional numeric vector of true parameter values (for simulation studies). Red dashed lines are drawn at these values.

Value

A list of class "mcem_diagnostics" (invisibly) with components convergence, stop_reason, final_fhat, IS_quality, and final_IS.

Convergence ($convergence)

A data frame with one row per EM iteration. The trace's last row, the final E-step at the returned parameters, is not an iteration and is not included; the table therefore has fit$iterations rows and its fhat of the final E-step is reported as $final_fhat. Columns:

iteration Iteration index.

fhat IS log-likelihood at the previous iterate (the E-step value the M-step of this iteration maximised).

delta_max Maximum relative parameter change from the previous iteration, each coordinate measured against its own magnitude.

rejected Draws rejected in the iteration's E-step: n_rejected (the sum over rejection channels) where the trace carries it, else the trace's own rejected column.

num_trees Number of augmented trees used.

time Wall-clock time for the iteration (ms).

IS quality ($IS_quality)

Importance-sampling diagnostics from the final EM iteration:

ESS Effective sample size.

ESS_fraction ESS as a fraction of num_trees.

mean_lw, sd_lw Mean and SD of IS log-weights.

`fhat` IS log-likelihood at the final parameters.

`loglik_var` Bootstrap variance of `fhat`.

See Also

`diagnose_cem`, `estimate_rates`

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
fit <- estimate_rates(phy, method = "mcem", model = "cr",
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 max_iter = 10, num_trees = 50))
diagnose_mcem(fit, lower_bound = c(-3, -5), upper_bound = c(1, 0))
```

---

emphasis_pipeline *Full estimation pipeline: auto_bounds -> GAM -> CEM -> MCEM*

Description

Runs a multi-stage estimation workflow with automatic bounds detection, exploratory fits (GAM and CEM), and optional MCEM refinement. Each stage is timed and logged. If MCEM fails, the best exploratory result is returned.

Usage

```
emphasis_pipeline(
  tree,
  model = "cr",
  link = "exponential",
  stages = c("bounds", "gam", "cem", "mcem"),
  control = list(),
  verbose = TRUE
)
```

Arguments

`tree`	A `phylo` object, branching-time vector, or `simulate_tree` result.
`model`	Model specification: `"cr"`, `"dd"`, `"d"`, `"nd"`, `"ed"`, `"ned"`, `"k"`, `"nk"`, a formula (e.g. `~ N + D`, `~ N + ED`, `~ N + K`), or a binary vector.
`link`	Link function: `"exponential"` (default), `rate = exp(eta)`, under which the intercepts `beta_0` and `gamma_0` are log rates; `"linear"`, `rate = max(0, eta)`, under which the parameters are rates and a covariate can drive a rate to zero, where it is clipped.

stages	Character vector of stages to run. Default c("bounds", "gam", "cem", "mcem"). Remove stages to skip them, e.g. c("bounds", "gam") for a quick exploratory fit.
control	Named list with optional sub-lists bounds, gam, cem, mcem to override defaults for each stage. Common parameters (num_threads, default 1, max_missing, max_time, rho) can be set at the top level and are inherited by all stages. rho, the sampling fraction in (0, 1], is inherited by the bounds, gam, cem and mcem stages alike, and a value nested under control$gam, control$cem or control$mcem overrides the inherited one for that stage.
verbose	Logical. If TRUE (default), print progress and a summary log to the console.

Value

A list of class "emphasis_pipeline" with:

pars Named parameter vector from the best stage.

loglik Log-likelihood from the best stage.

AIC AIC from the best stage.

best_stage Which stage produced the final result.

rho The sampling fraction the best stage ran at.

log Data frame with one row per stage: stage name, pars, loglik, AIC, elapsed seconds, status, rho.

mcem_trace If MCEM ran, a data frame of per-iteration fhat and parameter values. NULL otherwise.

bounds The auto_bounds result (bounds + survival GAM).

fits Named list of individual emphasis_fit objects.

Stages

1. **Bounds**: auto_bounds detects parameter bounds and trains a survival GAM, which the later stages condition on. The GAM is not accurate enough for that (experiment E43; see train_GAM). To condition on another survival probability, or not at all, leave this stage out and give control$lower_bound, control$upper_bound and, to condition, control$survival_gam in any form estimate_rates(cond =) accepts.
2. **GAM**: Fits a smooth likelihood surface over a grid and optimises it. Fast, one-shot.
3. **CEM**: Cross-entropy global search. Stochastic, no init needed. Explores the space differently than GAM.
4. **MCEM**: Monte Carlo EM starting from the best init found in stages 2–3. Local refinement for precise estimates.

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
# CEM stage only, with the search box given (log rates: the
# exponential link is the default)
res <- emphasis_pipeline(phy, model = "cr", stages = "cem",
```

```
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 cem = list(max_iter = 3, num_particles = 10)),
  verbose = FALSE)
res

# Bounds found automatically, then CEM
res2 <- emphasis_pipeline(phy, model = "cr", stages = c("bounds", "cem"),
  control = list(cem = list(max_iter = 3, num_particles = 10)),
  verbose = FALSE)
res2$log
```

`estimate_rates` *Estimate diversification rates from a phylogenetic tree*

Description

Unified entry point for parameter inference under the general diversification model. Accepts a simulated or empirical tree, extracts branching times (pruning to extant tips if needed), and dispatches to the chosen optimisation back-end.

Usage

```
estimate_rates(
  tree,
  method = c("mcem", "cem", "gam"),
  model = "cr",
  init_pars = NULL,
  control = list(),
  link = "exponential",
  cond = NULL
)
```

Arguments

`tree` One of:

- a `phylo` object (extant tips only recommended),
- the list returned by `simulate_tree` (`$tes` is used automatically, falling back to a pruned `$tas`), or
- a numeric vector of branching times (crown-age ordering).

The covariate `D` is a lineage's pendant age – measured from the last time that lineage became a tip – centred on the mean over the alive lineages, so it is a function of the topology and not of the branching times alone. A `phylo` (or a `simulate_tree` result) carries it; a bare numeric vector does not, and every observed lineage is then recorded as dating from the crown, which leaves `D = 0` at every observed branching event. Fit `"d"` or `"nd"` from a tree, not from branching times.

`method` Optimisation method:

`"mcem"` Monte Carlo Expectation-Maximisation (default). Iterates E- and M-steps until parameters stabilise (max relative change < `tol` for `patience` consecutive iterations) or `max_iter` is reached. Requires `init_pars`.

`"cem"` Monte Carlo Cross-Entropy Method. Initialises a particle population randomly within the bounds and evolves it until convergence (annealing exhausted, plateau, or `control$max_iter` reached). Does not need `init_pars`.

`"gam"` GAM-based MLE. Evaluates the IS log-likelihood on a grid spanning the bounds, fits a smooth GAM surface, and optimises it via L-BFGS-B. One-shot (non-iterative); no `init_pars` needed. Requires the **mgcv** package.

`model` Model specification. Accepts:

- a formula such as `~ N`, `~ N + D`, `~ N + ED` or `~ N + K`,
- a string shortcut (`"cr"`, `"dd"`, `"d"`, `"nd"`, `"ed"`, `"ned"`, `"edc"`, `"nedc"`, `"k"`, `"nk"`), or
- an integer vector `c(use_N, use_M, use_D, use_ED, use_K)` (`use_M` is internal only and should be 0; a shorter vector is an earlier layout and is padded; `use_ED = 3` selects the centred covariate).

`model = "all"` fits the nested ladder `cr`, `dd`, `nd`, `ned`, `nk` (the last two need a `phylo`), each started from the estimate of the model it nests, with each model's box from `auto_bounds` unless `control$bounds` names one per model, and returns the fits with their `compare_models` table and its `chosen` model, the fewest parameters within 2 AIC of the minimum (an `emphasis_fits` object); `control$models` restricts the ladder. Default `"cr"` (constant rate). `"nd"` (`~ N + D`) is diversity and age-imbalance; `"ned"` (`~ N + ED`) diversity and evolutionary distinctiveness, which needs a `phylo`. `"nedc"` (`~ N + EDc`) reads ED minus its mean over the lineages alive at the same time, a covariate with clade mean zero, so that `beta_ED` carries only the within-clade contrast and none of the clade-level level that `N` tracks. `"nk"` (`~ N + K`) is diversity and the number of speciation events on the path from the crown to the lineage, hidden splits counted; it needs a `phylo` and runs on the thinning sampler.

`init_pars` Starting parameter vector. Required for `"mcem"`; ignored for `"cem"`. If `NULL` with `"mcem"`, the midpoint of the bounds is used, with covariate slopes started at 0 (linear link) so that lambda stays positive during the early E-steps.

`control` Named list of tuning parameters. Must include `lower_bound` and `upper_bound`. Use `estimate_rates_control` to inspect other defaults. Key entries:

`lower_bound` Numeric vector of lower parameter bounds. Length must be `2 + 2*sum(model)`. Order: `c(beta_0, [beta_N], [beta_D], gamma_0, [gamma_N], [gamma_D])` (D = age-imbalance covariate; M is internal only).

`upper_bound` Numeric vector of upper parameter bounds (same order).

`verbose` Print progress messages. Default `FALSE`.

`link` Link function: `"exponential"` (default), `rate = exp(eta)`, under which the intercepts `beta_0` and `gamma_0` are log rates; `"linear"`, `rate = max(0, eta)`, under which the parameters are rates and a covariate can drive a rate to zero, where it is clipped.

cond — The survival probability to condition on, or NULL (default) for no conditioning. When provided, the log-likelihood is conditioned: $\ell_{\text{cond}} = \ell - \log P_\theta(S)$, where $P_\theta(S)$ is the probability that both crown lineages leave a sampled descendant at the present. "surface" (MCEM only) estimates $P_\theta(S)$ by direct simulation on local surfaces around the iterate, for any model; the section 'Conditioning on survival' describes it. The other forms give $P_\theta(S)$ directly: a function of the parameter data frame that returns it (the probability, not its log), a surface from survival_surface, a GAM from train_GAM, or a nnet object (see predict_survival). The GAM is not accurate enough to condition on.

Value

A list with class "emphasis_fit" containing:

pars Named numeric vector of parameter estimates.

loglik Final log-likelihood estimate. For MCEM this is the importance-sampling estimate at the returned parameters, and is NA when the E-step evaluating them failed.

loglik_var Bootstrap variance of loglik due to Monte Carlo noise. Populated automatically when control$num_trees >= 2 (B = 200 replicates); NA when num_trees = 1.

n_pars Number of free parameters (used for AIC): parameters with lower_bound == upper_bound are fixed and not counted.

AIC Akaike Information Criterion: -2 * loglik + 2 * n_pars.

method Method used ("mcem" or "cem").

model Resolved binary model vector.

rho The sampling fraction the fit was computed at, after the control merge.

stop_reason Why the MCEM run stopped ("converged", "max_iter", "time_budget", "e_step_failure", "m_step_failure"); NA for other methods.

iterations Number of completed MCEM iterations (0 when every E-step failed); NA for other methods.

n_failed Number of E- or M-step failures over the MCEM run; NA for other methods.

notes Named character vector of what the measurements and the run say about this fit: size, starved and turnover (emitted as a warning or messages when the fit is made), mean_field (the surrogate proposal's mean-field iteration was relaxed below the full step, ran under a damping other than "auto", or did not converge within its sweep budget: the setting, the sweeps taken, the relaxation and residual of the last E-step, and how many E-steps of the run relaxed or failed to converge) and clock (max_time ran out inside an E-step or an M-step). print() shows the last two.

details Full output from the back-end function (.mcem_dynamic_fresh or emphasis_cem); under cond = "surface", that of the last segment of the run kept.

conditioning Under cond = "surface" only: what the conditioning did. surfaces built, clades simulated and seconds spent simulating, over every run; logP and logP_se, the direct $\log P(S)$ at the estimate and its standard error, with logP_clades; loglik_uncond, the unconditioned log-likelihood at the estimate (loglik is it minus logP); start, the start of the run kept ("unconditioned", "nested", or "none" when no conditioned segment finished within max_time and the fit is the unconditioned one, conditioned at its estimate); restart, whether the run

from the nested start was made; `surface`, the last surface of the run kept; `runs`, the estimate, stop reason, iterations, segments, surfaces, trace and conditioned log-likelihood of each run; `settings`, `seed` and `seconds_total`.

Conditioning on survival

The survival emulator of `train_GAM` is not adequate for any lineage-level model. In experiment E43 (`https://franciscorichter.github.io/emphasis/experiments/E43.html`), GAMs trained on 2000, 5000 and 20 000 simulated clades for each of `d`, `nd`, `ed`, `edc`, `ned`, `nedc`, `k` and `nk` had a 90th-percentile error of $\log P(S)$ of 0.44 to 2.56, against the 0.05 its adequacy rule required, and the error did not fall with the training size.

Under `cr` and `dd`, condition on the exact survival probability. The package does not export a function for it: pass one to `cond`. The function receives a data frame with one row per parameter point and columns named as the parameters (`beta_0`, `gamma_0` under `cr`), and returns $P_\theta(S)$, one value per row. Under constant rates on the exponential link, with $\lambda = e^{\beta_0}$, $\mu = e^{\gamma_0}$, $d = \lambda - \mu$, crown age T and sampling fraction ρ, one crown lineage leaves a sampled descendant at the present with probability

$$p = \frac{\rho d}{\rho\lambda + (\lambda(1-\rho) - \mu)e^{-dT}}$$

($p = \rho/(1 + \rho\lambda T)$ when $\lambda = \mu$); the two crown lineages are independent, so $P_\theta(S) = p^2$. The last example below codes it. Under `dd` the exact value solves the master equation of the clade size, which the package does not provide. E43's Monte Carlo reference agreed with the exact values to a median error of 0.007 (`cr`) and 0.008 (`dd`) in $\log P$.

Under a lineage-level model (`d`, `nd`, `ed`, `edc`, `ned`, `nedc`, `k`, `nk`), condition with `cond = "surface"`. A direct Monte Carlo estimate of $P_\theta(S)$ (`survival_mc`) at a standard error of 0.01 in $\log P$ took E43 a median of 2 to 29 seconds of one core, depending on the model, and the M-step evaluates `cond` at every parameter value it tries, so the surface method estimates it on a local quadratic instead, in four steps:

1. The unconditioned fit, from `init_pars`, to the stopping rule.
2. A surface of $\log P(S)$ around its estimate (`survival_surface`): direct simulation at $2p + 1$ points (p the coefficients of a full quadratic in the parameters: 31 points for four parameters, 57 for six) in a box of half-width `surface_radius` (0.03) of the fit's box per coordinate, the same clades at every point, as many as give a standard error of `surface_se` (0.02) at the centre, a clade past the lineage cap counted as surviving; a quadratic by weighted least squares, the tangent plane outside the box.
3. MCEM continued from the unconditioned estimate (clamped 10^{-3} of the box span inside the box) with the surface as `cond`, in segments of `surface_segment` (25) iterations that carry the draw count forward; the surface is rebuilt at the iterate whenever the iterate leaves the inner half of its box, and the run ends when the stopping rule is met inside the inner half, or on `max_iter` iterations of the run or `max_time`.
4. When the unconditioned `gamma_0` is within `surface_edge` (10^{-3}) of the box span of an edge, step 3 is also run from `init_pars` (the nested start: the estimate of the nested model with the lineage-level coefficients at zero), and the run with the higher conditioned log-likelihood is kept.

The reported log-likelihood is the run's unconditioned value minus a direct $\log P(S)$ at its estimate (standard error `surface_final_se`, 0.01), and `loglik_var` adds that error's square. `max_time`

bounds steps 1 to 4 together; the direct estimate at the estimate then gets `max_time` again, as the final E-step does. What the conditioning did is in the fit's `conditioning`. In the pilot that chose the method for experiment E46 (100 tips, `cr` and `dd`, where the exact conditioned maximum is known), steps 1 to 3 reached a median deficit of 0.0011 nats against the exact conditioned maximum (worst 0.13), and without step 4 a second, lower maximum on the extinction edge was kept on 4 of 10 `cr` trees at turnover 0.10.

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
# Exponential link (the default): the intercepts are log rates
fit <- estimate_rates(phy, method = "mcem", model = "cr",
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 max_iter = 10, num_trees = 50))
fit

# Diversity-dependent model on the same tree
fit_dd <- estimate_rates(phy, method = "mcem", model = "dd",
  control = list(lower_bound = c(-3, -0.1, -5, -0.1),
                 upper_bound = c(1, 0.1, 0, 0.1),
                 max_iter = 10, num_trees = 50))
fit_dd$pars

# Conditioning a cr fit on its exact survival probability, P(S) = p^2
cond_cr <- function(T, rho = 1) function(pars) {
  lambda <- exp(pars$beta_0)
  mu     <- exp(pars$gamma_0)
  d      <- lambda - mu
  p <- ifelse(abs(d) < 1e-12, rho / (1 + rho * lambda * T),
              rho * d / (rho * lambda + (lambda * (1 - rho) - mu) * exp(-d * T)))
  p^2
}
crown_age <- max(ape::branching.times(phy))
fit_cond <- estimate_rates(phy, method = "mcem", model = "cr",
  cond = cond_cr(T = crown_age),
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 max_iter = 10, num_trees = 50))
fit_cond

# The same fit conditioned on local surfaces of a direct estimate, the
# route for the lineage-level models
fit_surf <- estimate_rates(phy, method = "mcem", model = "cr",
  cond = "surface",
  control = list(lower_bound = c(-3, -5), upper_bound = c(1, 0),
                 max_iter = 10, num_trees = 50, max_time = 20))
fit_surf$conditioning[c("surfaces", "clades", "logP", "logP_se", "start")]
```

lineage_table *The per-lineage design of an augmented tree*

Description

One row per lineage alive on each segment between consecutive events of an augmented tree: the segment's bounds, the covariates the rates read there (`N`, the clade mean pendant age `M` at the segment's end `t1`, the lineage's pendant start `ts`, so that `D = (t1 - ts) - M`, constant on the segment, and its `ED` at the segment start, growing at slope one within it), and whether the lineage's own event ends the segment (`event`: 0 none, 1 speciation, 2 extinction). These are the numbers the likelihood is built on; `covariate_path` runs a selection path over them.

Usage

```
lineage_table(tree)
```

Arguments

`tree`	An augmented-tree data frame with the topology (the frames `augment_trees` returns when given parent ids), or a `phylo` object, which is taken as fully observed.

Value

A data frame with columns `seg`, `lineage`, `id`, `event`, `t0`, `t1`, `N`, `M`, `ts`, `ed0`.

Examples

```
set.seed(1)
tr  <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
phy <- tr$tes
head(lineage_table(phy))
```

predict.emphasis_surface *Evaluate a survival surface*

Description

Evaluate a survival surface

Usage

```
## S3 method for class 'emphasis_surface'
predict(object, newdata, type = c("logP", "P", "gradient"), ...)
```

Arguments

object	An "emphasis_surface" from survival_surface or from a fit's conditioning$surface.
newdata	A parameter vector, or a matrix or data frame of them (one row a point; columns matched by name when named as the parameters, otherwise by position).
type	"logP" (default), "P", or "gradient", the gradient of $\log P$ in the parameters (outside the surface's box, the slope of its tangent plane; zero where the cap at $\log P = 0$ binds).
...	Ignored.

Value

A numeric vector (one value per point), or for "gradient" a matrix with one row per point.

---

predict_survival *Predict survival probability from a trained emulator*

Description

Evaluates the survival probability estimate_rates(cond =) conditions through, in any of its four forms: a function that takes the parameter data frame and returns probabilities, a surface from survival_surface, a gam from train_GAM, or a nnet object.

Usage

```
predict_survival(gam_fit, newpars)
```

Arguments

gam_fit	A gam object returned by train_GAM, a nnet object trained on the same columns, an "emphasis_surface" from survival_surface, or a function of a parameter data frame.
newpars	Numeric matrix or data frame of parameter values to predict at. Column names must match those used in train_GAM.

Details

The GAM is not accurate enough to condition on: in experiment E43 (https://franciscorichter.github.io/emphasis/experiments/E43.html) its 90th-percentile error of $\log P(S)$ was 0.44 to 2.56 for every lineage-level model, against 0.05 required. Under cr and dd pass the exact survival probability as a function, and under a lineage-level model use cond = "surface"; the section 'Conditioning on survival' of estimate_rates gives the cr formula and the alternatives.

Value

Numeric vector of predicted survival probabilities.

Examples

```
set.seed(1)
# Exponential link (the default): the intercepts are log rates
pars_mat <- cbind(beta_0  = log(runif(100, 0.3, 1.2)),
                  gamma_0 = log(runif(100, 0.05, 0.5)))
sims <- simulate_tree(pars = pars_mat, max_t = 4, model = "cr",
                      max_tries = 1)
gam_fit <- train_GAM(sims$simulations, pars_mat, model = "cr")
predict_survival(gam_fit, data.frame(beta_0  = log(c(0.5, 1.0)),
                                     gamma_0 = log(c(0.1, 0.3))))
```

---

print.cem_diagnostics *Print method for CEM diagnostics*

Description

Print method for CEM diagnostics

Usage

```
## S3 method for class 'cem_diagnostics'
print(x, ...)
```

Arguments

x	Output of diagnose_cem.
...	Additional arguments (ignored).

Value

x, invisibly.

---

print.emphasis_fit *Print method for emphasis_fit objects*

Description

Print method for emphasis_fit objects

Usage

```
## S3 method for class 'emphasis_fit'
print(x, ...)
```

Arguments

x	An emphasis_fit object returned by estimate_rates.
...	Additional arguments (ignored).

Value

x, invisibly.

---

print.emphasis_fits *Print method for the fits of* estimate_rates(model = "all")

Description

Print method for the fits of estimate_rates(model = "all")

Usage

```
## S3 method for class 'emphasis_fits'
print(x, ...)
```

Arguments

x	An emphasis_fits object.
...	Ignored.

Value

x, invisibly.

---

print.emphasis_pipeline *Print method for emphasis_pipeline objects*

Description

Print method for emphasis_pipeline objects

Usage

```
## S3 method for class 'emphasis_pipeline'
print(x, ...)
```

Arguments

x	An emphasis_pipeline result.
...	Additional arguments (ignored).

Value

x, invisibly.

`print.emphasis_surface` *Print a survival surface*

Description

Print a survival surface

Usage

```
## S3 method for class 'emphasis_surface'
print(x, ...)
```

Arguments

x	An "emphasis_surface".
...	Ignored.

Value

x, invisibly.

`print.gam_diagnostics` *Print method for GAM diagnostics*

Description

Print method for GAM diagnostics

Usage

```
## S3 method for class 'gam_diagnostics'
print(x, ...)
```

Arguments

x	Output of diagnose_gam.
...	Additional arguments (ignored).

Value

x, invisibly.

print.mcem_diagnostics *Print method for MCEM diagnostics*

Description

Print method for MCEM diagnostics

Usage

```
## S3 method for class 'mcem_diagnostics'
print(x, ...)
```

Arguments

x	Output of diagnose_mcem.
...	Additional arguments (ignored).

Value

x, invisibly.

simulate_tree *Simulate a phylogenetic tree under the general diversification model*

Description

Unified entry point for forward and conditional tree simulation. When tree = NULL (default) a complete tree is simulated forward from the crown; when tree is supplied the observed extant tree is augmented with stochastically drawn extinct lineages.

Usage

```
simulate_tree(
  tree = NULL,
  pars,
  max_t = 1,
  model = "cr",
  max_lin = 1e+06,
  max_tries = 1,
  useDDD = TRUE,
  n_trees = 1L,
  link = "exponential",
  max_missing = 10000,
  max_lambda = 500,
```

```
  maxN = NULL,
  num_threads = 1L,
  rho = 1,
  method = "bdi"
)
```

Arguments

tree	Optional observed extant tree for augmentation. Accepts NULL (forward sim), a simulate_tree result, a phylo object, or a numeric branching-time vector. A phylo also supplies the topology the pendant-age covariates M and D are computed from; with a bare branching-time vector every observed lineage is recorded as dating from the crown and D = 0 at every observed branching event.
pars	Numeric parameter vector or matrix. When a matrix each row produces one forward simulation.
max_t	Crown age (forward simulation only). Default 1.
model	Model specification (string, formula, or binary vector). Default "cr".
max_lin	Maximum lineages alive before declaring the tree too large; a clade that has had ten times as many lineages in all, alive and dead, is too large as well, which bounds the cost of a simulation that churns (high speciation and extinction over a long crown) under the cap. Default 1e6.
max_tries	Maximum number of forward simulations in all, the first included (forward simulation only): a clade that goes extinct is drawn again until one survives or max_tries draws have been made, so the default 1 is a single draw, unconditioned on survival. A positive integer; 0, which asked for a single draw when the argument counted the draws after the first, is read as 1 with a warning. A clade that overflows max_lin is not retried: the overflow is a property of the parameters, not of the draw, and every retry would run to the cap again. The draws are counted at the R level (attempts) to compute survival_prob. Default 1.
useDDD	Convert L-table to phylo via **DDD**. Default TRUE.
n_trees	Number of augmented trees to draw (augmentation only). Default 1L.
link	Link function: "exponential" (default), rate = exp(eta), under which the intercepts beta_0 and gamma_0 are log rates; "linear", rate = max(0, eta), under which the parameters are rates and a covariate can drive a rate to zero, where it is clipped.
max_missing	Maximum extinct lineages per augmented tree. Default 1e4.
max_lambda	Maximum speciation rate (thinning bound) for augmentation. Default 500.
maxN	Maximum total augmentation attempts. NULL (default) sets this to max(2000, 200 * n_trees).
num_threads	Threads for parallel augmentation. Default 1L.
rho	Sampling fraction in (0, 1]. Default 1 (complete sampling). With rho < 1 a random fraction of extant tips is dropped (forward simulation) or unsampled extant lineages are inserted (augmentation), matching incomplete taxon sampling.

`method` Augmentation method: `"bdi"` (default, exact BDI sampler) or `"thinning"` (C++ thinning proposal). The BDI sampler draws from the exact conditional distribution under constant rates (ESS = S, zero variance) and uses a self-consistent backward-forward iteration under diversity dependence. It covers N-only models (`"cr"`, `"dd"`) on the linear and exponential links; any other model or link falls back to thinning.

`method = "bdi"` **returns** `NULL` **for part of its draws.** The BDI sampler has no notion of the two crown lineages, so a lineage born before the first observed branching is recorded as a daughter of observed node 0, which is not yet born then. Such a draw cannot be turned into a `tas`: it is refused, and that element of `trees` (or `tas`, with `n_trees = 1`) is `NULL`. On seven trees measured for this release, 0 to 168 of 200 draws built; a tree whose first observed branching is late loses most of them. `log_q` is returned for every draw either way. It was previously silent and wrong – every such draw produced a `phylo` with an edge of negative length. The `"thinning"` method draws the parent from every lineage alive, the two crown lineages included, and is not affected.

Value

A list whose components depend on the mode: for a single forward simulation `tes`, `tas`, `L`, `status`, `survival_prob` and `attempts`; for a matrix `pars` a list with `simulations` and `survival_prob`; for an augmentation the augmented tree(s) and their proposal log-densities `log_q`. The sections below list every component.

Model specification

The model's covariates are diversity `N`, age-imbalance `D = E - M` (a lineage's isolation relative to the clade mean `M`, which is used only internally to centre `D`) and evolutionary distinctiveness `ED` (the fair-proportion score on the complete tree: the branches from the crown to the lineage, each divided by the alive lineages below it; its last term is the pendant age and its sum over lineages is Faith's phylogenetic diversity). Selected by named shortcuts or a formula:

`"cr"`	constant rate
`"dd"`	diversity dependence (N)
`"d"`	age-imbalance (D)
`"nd"`	N + D
`"ed"`	evolutionary distinctiveness (ED)
`"ned"`	N + ED
`"edc"`	centred ED (ED minus its mean over the lineages alive)
`"nedc"`	N + centred ED
`"k"`	ancestral splits (K: the speciation events on the path from the crown)
`"nk"`	N + K

Formulas `~ N`, `~ N + D`, `~ N + ED`, `~ N + EDc` and `~ N + K` are also accepted (`ep` is a legacy alias for `d`). An `ED` or `K` model needs the tree's topology (a `phylo`, not a branching-time vector).

Parameter vector

Compact layout; `beta_D` is the age-imbalance coefficient and `beta_ED` the distinctiveness coefficient (`M` is internal only, never part of the user vector):

```
c(beta_0,  [beta_N],  [beta_D],  [beta_ED],
  gamma_0, [gamma_N], [gamma_D], [gamma_ED])
```

Only active covariates are included; length = `2 + 2 * sum(model)`.

Reproducibility

Every call draws its seed for the C++ sampler from R's generator, so `set.seed()` fixes both the forward simulator and the thinning augmenter: two runs under one seed return the same tree, two runs under different seeds do not. The retries a forward simulation makes draw a seed each, so they are distinct trees rather than one tree redrawn. Under `parallel::mclapply` each child has its own R stream and therefore its own seeds. Augmentation with `num_threads > 1` is not reproducible: each worker draws from its own substream of the seed, but the sample keeps the first `n_trees` augmentations to finish.

Output – forward simulation (single)

`tes` Extant-only phylogeny (`phylo`), or `NULL`.

`tas` Full phylogeny with extinct lineages (`phylo`), or `NULL`.

`L` L-table (DDD format), or `NULL` if simulation failed.

`status` `"done"`, `"extinct"`, or `"too_large"`.

`survival_prob` Empirical survival probability: `1 / n_attempts` when the simulation succeeded, `0` otherwise.

`attempts` Number of forward simulations run, counting the retries after extinction; 1 when the first draw completed or overflowed.

Output – forward simulation (batch, matrix `pars`)

`simulations` List of individual results (one per row of `pars`), each with the fields above.

`survival_prob` Fraction of simulations with `status == "done"`.

Output – conditional simulation / augmentation

When `n_trees = 1`:

`tas` Augmented phylogeny (`phylo`) including extinct lineages, or `NULL` on failure or when `useDDD = FALSE`.

`log_q` Log sampling probability $\log q(z \mid \text{obs}, \theta)$ of the augmentation. `NA` on failure.

When `n_trees > 1`:

`trees` List of `tas` phylogenies (length up to `n_trees`; failed draws are `NULL`).

`log_q` Numeric vector of log sampling probabilities, one per successful draw.

Examples

```
# Forward simulation.  Under the exponential link (the default) the
# intercepts are log rates: speciation 0.5, extinction 0.1
set.seed(1)
tr <- simulate_tree(pars = log(c(0.5, 0.1)), max_t = 4, model = "cr")
tr$status
tr$survival_prob

# Batch: 20 constant-rate simulations
pm <- cbind(beta_0  = log(runif(20, 0.3, 0.8)),
            gamma_0 = log(runif(20, 0.05, 0.3)))
sims <- simulate_tree(pars = pm, max_t = 4, model = "cr")
sims$survival_prob

# Augmentation of the extant tree
aug <- simulate_tree(tree = tr, pars = log(c(0.5, 0.1)), model = "cr")
aug$log_q

# Many augmented trees
augN <- simulate_tree(tree = tr, pars = log(c(0.5, 0.1)), model = "cr",
                      n_trees = 20L)
length(augN$log_q)

# The linear link: the parameters are the rates themselves
tr_lin <- simulate_tree(pars = c(0.5, 0.1), max_t = 4, model = "cr",
                        link = "linear")
```

`survival_mc` *Probability that a clade survives, by direct simulation*

Description

Estimates $\log P_\theta(S)$, the log-probability that both crown lineages of a clade of crown age `crown_age` leave a sampled descendant at the present, from forward simulations under any model and link. This is the survival probability `estimate_rates(cond = )` divides the likelihood by.

Usage

```
survival_mc(
  pars,
  model = "cr",
  crown_age,
  rho = 1,
  link = "exponential",
  n_clades = 2000L,
  se = NULL,
  max_clades = 40000L,
  max_lin = 10000,
```

```
  seed = NULL,
  max_time = NULL
)
```

Arguments

pars	Parameter vector in the compact layout of model (as in simulate_tree).
model	Model name, formula or binary vector, as in simulate_tree. Default "cr".
crown_age	Crown age of the clade.
rho	Sampling fraction in (0, 1]. Default 1.
link	"exponential" (default) or "linear".
n_clades	Clades to draw; with se, the clades drawn first. Default 2000.
se	Target standard error of $\log P$, or NULL (default) to draw n_clades only. When set, the first n_clades give a provisional P and the clades are extended to the number that reaches se there, at most max_clades.
max_clades	Cap on the clades drawn under se. Default 40000.
max_lin	Lineage cap of one clade. Default 1e4.
seed	Positive integer, or NULL (default) to draw one from R's generator.
max_time	Seconds after which no further clade is drawn, or NULL (default) for no limit; the estimate then rests on the clades drawn.

Details

Clade i is one forward draw of the simulator seeded with seed + i, so calls with the same seed at different parameter values draw from the same random numbers and their difference has a much smaller error than either value. A clade that passes max_lin lineages is counted as surviving, as train_GAM counts it. Under rho < 1 a clade that reaches the present with n_1 and n_2 extant descendants of its two crown lineages contributes $(1-(1-\rho)^{n_1})(1-(1-\rho)^{n_2})$, the probability that sampling each extant tip with probability ρ keeps at least one on each side. The estimate is $\log((k+1/2)/(n+1))$, k the sum of the clades' contributions and n the clades drawn, and its standard error is the delta-method one, $s/(\sqrt{n}P)$ with s^2 the variance of the contributions (the binomial $\sqrt{(1-P)/(nP)}$ at $\rho = 1$).

Value

A named numeric vector: logP, se, P, survivors (the sum of the contributions), clades, seconds and seed.

See Also

survival_surface, estimate_rates

Examples

```
# cr at rho = 1 against the closed form p^2 of ?estimate_rates
s <- survival_mc(log(c(0.5, 0.2)), model = "cr", crown_age = 3,
                 n_clades = 2000, seed = 1)
l <- 0.5; m <- 0.2; d <- l - m; T <- 3
c(mc = s[["logP"]], se = s[["se"]],
  exact = 2 * log(d / (l - m * exp(-d * T))))
```

`survival_surface` *A local surface of the survival probability*

Description

Fits a quadratic surface of $\log P_\theta(S)$ (see `survival_mc`) in a box around `pars`, from direct simulation at $2p+1$ points, $p = (d+1)(d+2)/2$ the coefficients of a full quadratic in the d parameters (31 points for four parameters, 57 for six): the centre and a Latin hypercube of the box of half-width `radius`. Every point draws the same clades (common random numbers), as many as give a standard error of `se` at the centre, and the quadratic is fitted by weighted least squares, each point weighted by the inverse square of its standard error. Outside the box the surface continues as the tangent plane at the point where the ray from the centre leaves the box; it is capped at $\log P = 0$.

Usage

```
survival_surface(
  pars,
  model = "cr",
  crown_age,
  rho = 1,
  link = "exponential",
  radius = NULL,
  lower_bound = NULL,
  upper_bound = NULL,
  se = 0.02,
  n_clades = NULL,
  max_clades = 40000L,
  max_lin = 10000,
  seed = NULL,
  max_time = NULL
)
```

Arguments

`pars`	The centre, in the compact layout of `model`.
`model`	Model name, formula or binary vector, as in `simulate_tree`. Default `"cr"`.
`crown_age`	Crown age of the clade.
`rho`	Sampling fraction in `(0, 1]`. Default `1`.

`link` "exponential" (default) or "linear".

`radius` Half-width of the box per parameter (recycled), or NULL (default) for 0.03 of the span of lower_bound–upper_bound.

`lower_bound, upper_bound` Optional parameter box; design points are clamped into it.

`se` Standard error of $\log P$ at the centre the clade count is set for. Default 0.02.

`n_clades` Clades per design point, or NULL (default) to set it from se: 2000 clades (other seeds) give a provisional P at the centre, and the count is the one that reaches se there, between 500 and max_clades.

`max_clades` Cap on the clades drawn under se. Default 40000.

`max_lin` Lineage cap of one clade. Default 1e4.

`seed` Positive integer, or NULL (default) to draw one from R's generator.

`max_time` Seconds after which the construction stops with an error, or NULL (default) for no limit.

Details

The surface is the conditioning estimate_rates(cond = "surface") builds around its iterate, and it can be passed to cond itself, or to predict_survival. predict() returns $\log P$, P or the gradient of $\log P$ in the parameters.

Value

An object of class "emphasis_surface": centre, radius, the quadratic's coef (in the coordinates $z = (\theta - \text{centre})/\text{radius}$), the design (the points with their logP, se and P), n_clades per point, points, clades in all, seconds, and the model, link, crown age, rho, lineage cap and seed it was built for.

See Also

survival_mc, estimate_rates

Examples

```
S <- survival_surface(log(c(0.5, 0.2)), model = "cr", crown_age = 3,
                      radius = 0.1, n_clades = 1000, seed = 1)
predict(S, log(c(0.52, 0.2)))                  # log P
predict(S, log(c(0.52, 0.2)), type = "gradient")
```

`train_GAM` *Train a survival-probability GAM on forward simulation results*

Description

Fits a binomial GAM to estimate P(survival | parameters) from forward simulations run with `max_tries = 1` (one draw each, unconditioned on survival). Each simulation row has a binary outcome: 1 if both crown lineages reached the present (`status "done"`) or the clade passed the lineage cap (`"too_large"`), 0 if it went extinct. A clade stopped at the cap has survived so far, and the conditional likelihood divides by the probability of survival; counting it as extinct teaches the GAM a survival near zero wherever clades grow fast, which then rewards the fit for moving there (E36c of the paper).

Usage

```
train_GAM(
  simulations,
  pars_mat,
  model = "cr",
  spline_type = c("univariate", "bivariate"),
  verbose = TRUE
)
```

Arguments

`simulations`	List of `simulate_tree()` outputs (one per row of `pars_mat`), produced with `max_tries = 1`.
`pars_mat`	Numeric matrix whose rows are the parameter vectors used to generate `simulations`. Column names are used as predictor names; if absent, names are auto-generated from `model`.
`model`	String or binary integer vector (length 3 or 4) matching the model used in simulation. Used only when `pars_mat` has no column names. Default `"cr"`.
`spline_type`	`"univariate"` (default) fits independent smooth terms `s(x)` for each varying predictor. `"bivariate"` fits tensor-product smooths `te(x, y)` for every pair of varying predictors, which can capture interaction effects.
`verbose`	Report the number of simulations and the survival rate (default `TRUE`).

Details

The GAM is not accurate enough to condition a fit on. In experiment E43 (https://franciscorichter.github.io/emphasis/experiments/E43.html), GAMs trained on 2000, 5000 and 20 000 clades for each of the eight lineage-level models had a 90th-percentile error of $\log P(S)$ of 0.44 to 2.56, against the 0.05 its adequacy rule required, and the error did not fall with the training size. Condition `cr` and `dd` fits on the exact survival probability, and lineage-level fits with `estimate_rates(cond = "surface")`; the section 'Conditioning on survival' of `estimate_rates` shows how.

Value

A gam object (binomial family) representing $P(\text{survival} \mid \theta)$.

See Also

simulate_tree, predict_survival

Examples

```
set.seed(1)
# Exponential link (the default): the intercepts are log rates
pars_mat <- cbind(beta_0  = log(runif(100, 0.3, 1.2)),
                  gamma_0 = log(runif(100, 0.05, 0.5)))
sims <- simulate_tree(pars = pars_mat, max_t = 4, model = "cr",
                      max_tries = 1)
gam_uni <- train_GAM(sims$simulations, pars_mat, model = "cr")
predict_survival(gam_uni, data.frame(beta_0 = log(0.8), gamma_0 = log(0.2)))
```

Index

auto_bounds, 2, *13*, *15*

compare_models, 4, *15*
covariate_path, 6, *19*

diagnose_cem, 7, *10*, *12*, *21*
diagnose_gam, 9, *23*
diagnose_mcem, *10*, 10, *24*

emphasis_cem, *7–9*
emphasis_pipeline, 12
estimate_rates, *3–5*, *7–13*, 14, *20*, *22*, *29*, *31*, *32*
estimate_rates_control, *15*

lineage_table, 19

predict.emphasis_surface, 19
predict_survival, *16*, 20, *31*, *33*
print.cem_diagnostics, 21
print.emphasis_fit, 21
print.emphasis_fits, 22
print.emphasis_pipeline, 22
print.emphasis_surface, 23
print.gam_diagnostics, 23
print.mcem_diagnostics, 24

simulate_tree, *12*, *14*, 24, *29*, *30*, *33*
survival_mc, *17*, 28, *30*, *31*
survival_surface, *16*, *17*, *20*, *29*, 30

train_GAM, *3*, *13*, *16*, *17*, *20*, *29*, 32