Package ‘margEVT’

October 14, 2026

Type Package

Title Regularized Point Processes and Stochastic Marginalization for Extremes

Version 0.4.0

Description Implements a non-stationary extreme value analysis framework by coupling a covariate-driven non-homogeneous Poisson process (NHPP) with (adaptive) elastic-net regularization and analytical gradients. Provides methods for estimating conditional return levels and unconditional (marginalized) return levels via parametric stochastic integration over stable vector autoregressive VAR(p) covariate trajectories, or non-parametric annual-block resampling. Supports block-specific penalty controls, BIC selection along a regularization path, Karush-Kuhn-Tucker convergence checks, time-rescaling residual diagnostics, conditional parametric bootstrap inference, and walk-forward assessment.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/rodrigosqrt3/margEVT

BugReports https://github.com/rodrigosqrt3/margEVT/issues

RoxygenNote 7.3.3

Imports graphics, grDevices, parallel, stats, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vars

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Rodrigo Fonseca Villa [aut, cre] (ORCID: <https://orcid.org/0009-0005-2938-2270>)

Maintainer Rodrigo Fonseca Villa <rodrigo03.villa@gmail.com>

Repository CRAN

Date/Publication 2026-10-13 23:00:15 UTC

Contents

active_covariates . . . 2
backtest . . . 3
bic_nhpp . . . 5
bootstrap_coef . . . 6
bootstrap_rl . . . 7
build_cov_annual . . . 9
build_design_matrices . . . 10
coef.nhpp_fit . . . 11
fit_nhpp . . . 12
fit_var_generator . . . 14
is_nhpp_fit . . . 16
marginalize . . . 17
nhpp_fit-methods . . . 19
n_exceedances . . . 20
plot.nhpp_fit . . . 20
plot.nhpp_rl . . . 21
pp_grad . . . 22
pp_nllh . . . 23
predict_params . . . 24
print.nhpp_fit . . . 25
print.summary.nhpp_fit . . . 26
residuals.nhpp_fit . . . 26
rl_table . . . 27
seasonal_harmonics . . . 28
simulate_covariates . . . 29
summary.nhpp_fit . . . 30

Index **32**

`active_covariates` *Extract names of non-seasonal covariates from a fitted model*

Description

Returns the covariate names that appear in the model with a non-negligible coefficient - i.e. those that actually affect the parameters and need to be supplied during marginalization. Seasonal harmonic columns (`cos<k>`, `sin<k>`, `sen<k>`) are always excluded.

Usage

```
active_covariates(fit, free_cols = NULL, tol = NULL)
```

Arguments

`fit`	An `nhpp_fit` object.
`free_cols`	Character vector of column names to always exclude from the result (intercept and seasonality terms are always excluded). Default `NULL`.
`tol`	Numeric. Coefficients smaller than this in absolute value are treated as zero. If `NULL`, inherits `fit$active_tol`; legacy objects fall back to `1e-2`.

Value

Character vector of active covariate names, or `NULL` if the model has no active non-seasonal covariates.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(2000), z = rnorm(2000))
df$y <- 10 + 2 * df$x + rexp(2000, 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = c("x", "z"),
                penalty = "lasso", lambda = 5, verbose = FALSE)
active_covariates(fit)
```

---

backtest — *Walk-forward backtesting for NHPP extreme value models*

Description

Evaluates predictive performance of a fitted `nhpp_fit` model via walk-forward cross-validation. For each window, the model is refit on training data at the original penalty and lambda, return levels are computed, and exceedance rates are compared to nominal rates via binomial tests. A Kolmogorov-Smirnov comparison on tail-conditional PIT values is returned as an exploratory diagnostic; it is not a formal test of global predictive calibration. Validation years with fewer than `min_obs_year` observed values are excluded from binomial denominators.

Usage

```
backtest(
  fit,
  data,
  varname,
  TRs = c(2, 5, 10),
  n_obs = NULL,
  period = NULL,
  window_years = 5L,
  min_train_years = 10L,
  n_boot = 200L,
  interactions = list(),
```

```
  year_col = "year",
  min_obs_year = NULL,
  seed = 2024L,
  trend_col = NULL,
  fixed = list(),
  reselect_lambda = FALSE,
  cores = 1L,
  verbose = TRUE
)
```

Arguments

`fit`	An `nhpp_fit` object.
`data`	The data frame used to fit `fit`. Must contain columns `y`, `year` (or `year_col`), and all active covariates.
`varname`	Character. Name of the column in `data` containing the raw (non-declustered) series, used to compute observed annual maxima for validation.
`TRs`	Numeric vector of return periods to evaluate. Default `c(2, 5, 10)`.
`n_obs`	Integer observations per year. If `NULL`, uses the rounded `fit$obs_per_year` value.
`period`	Numeric seasonal period in observations. If `NULL`, inherits `fit$obs_per_year`.
`window_years`	Integer. Width of each validation window in years. Default `5L`.
`min_train_years`	Integer. Minimum number of training years before the first validation window. Default `10L`.
`n_boot`	Integer. Bootstrap years for covariate marginalization within each window. Default `200L`.
`interactions`	Named list of interactions passed to `build_cov_annual`. Default `list()`.
`year_col`	Character. Name of the year column. Default `"year"`.
`min_obs_year`	Integer. Minimum observations in a validation year for it to be included. Default `as.integer(n_obs * 0.8)`.
`seed`	Integer. Seed used for empirical annual-block sampling. Default `2024L`. Each validation window uses a distinct derived seed.
`trend_col`	Optional name of a linear time-trend covariate. Within each window it is recomputed as $(t - t_0)/(t_1 - t_0)$, where t_0 is the first year of `data` and t_1 the last training year, so that the trend equals 1 at the end of training. Return levels are evaluated at trend 1; the PIT of a validation year uses its extrapolated trend value. Default `NULL` (no trend).
`fixed`	Named list of scalar covariate values imposed on every covariate block (see `marginalize`). Default `list()`.
`reselect_lambda`	Logical. If `TRUE`, the penalty is re-selected by BIC in every training window; otherwise the original `lambda` is kept. Adaptive penalty weights are always recomputed within each training window, so the weights do not use validation data. Default `FALSE`.

cores	Positive integer number of worker processes over which the windows are distributed. Results do not depend on cores. Default 1L.
verbose	Logical. Print progress. Default TRUE.

Value

A list with three elements:

results	Data frame with one row per validation year, containing the observed annual maximum, PIT value, and exceedance indicators for each return period.
binom_tests	Data frame with nominal binomial comparisons for each TR. The not_rejected field is the recommended interpretation; calibrated is retained as a legacy alias.
ks_test	Data frame with the exploratory tail-conditional PIT KS comparison, including not_rejected and the legacy calibrated alias, or NULL if fewer than 5 validation years had events.

See Also

marginalize, fit_nhpp

Examples

```
set.seed(1)
n_years <- 30; n_obs <- 50
df <- data.frame(x = rnorm(n_years * n_obs),
                 year = rep(1991:2020, each = n_obs))
df$y <- 5 + df$x + rexp(nrow(df), 1 / 2)
fit <- fit_nhpp(df, threshold = 10, loc_vars = "x", penalty = "none",
                obs_per_year = n_obs, verbose = FALSE)

bt <- backtest(fit, df, varname = "y", TRs = c(2, 5), n_obs = n_obs,
               window_years = 5, min_train_years = 15, n_boot = 50,
               verbose = FALSE)
bt$binom_tests
```

---

bic_nhpp *Compute BIC for a fitted nhpp_fit model*

Description

Computes $-2\ell + k \log m$, where ℓ is the unpenalized log-likelihood at the operational estimator, k the number of active parameters, and m the number of exceedances. Equivalent to BIC(fit) for fits produced by margEVT 0.4.0 or later.

Usage

```
bic_nhpp(fit, tol = NULL)
```

Arguments

fit	An nhpp_fit object.
tol	Numeric. Coefficients smaller than this are counted as zero for the active parameter count. If NULL, inherits fit$active_tol; legacy objects fall back to 1e-2.

Value

Numeric scalar. BIC value.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
bic_nhpp(fit)
```

---

bootstrap_coef *Parametric bootstrap confidence intervals for model coefficients*

Description

Simulates R bootstrap samples, refits at the original penalty, lambda, and penalty weights, and returns empirical percentile intervals for each operational coefficient. See bootstrap_rl for the resampling scheme and the random-number streams.

Usage

```
bootstrap_coef(
  fit,
  data,
  R = 200L,
  level = 0.95,
  seed = 42L,
  cores = 1L,
  verbose = TRUE
)
```

Arguments

`fit`	An `nhpp_fit` object.
`data`	The data frame used to fit `fit`. Must contain column `y` and all covariates.
`R`	Integer. Number of bootstrap replicates. Default `200L`.
`level`	Numeric. Confidence level. Default `0.95`.
`seed`	Seed of the replicate streams, or `NULL` to derive it from the current random-number state. Default `42L`.
`cores`	Positive integer number of worker processes. Default `1L`.
`verbose`	Logical. Print progress. Default `TRUE`.

Value

A data frame with columns `parameter`, `estimate`, `CI_low`, `CI_high`, `n_ok`, with one row for every coefficient in the fitted model. Intervals require at least five successful replicates. The replicate matrix is attached as attribute `"replicates"`.

See Also

`bootstrap_rl`

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(1500))
df$y <- 10 + df$x + rexp(1500, 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = "x", penalty = "none",
                verbose = FALSE)

bootstrap_coef(fit, df, R = 50, verbose = FALSE)
```

`bootstrap_rl` *Parametric bootstrap confidence intervals for return levels*

Description

Simulates `R` bootstrap samples from the fitted model, refits at the original model design, penalty, lambda, penalty weights, and activity tolerance (starting from the original estimate), recomputes return levels via `marginalize`, and returns empirical percentile intervals. Exceedance count and times are held fixed, so the result is a nominal conditional summary rather than unconditional or post-selection inference.

Usage

```
bootstrap_rl(
  fit,
  data,
  TRs = c(2, 5, 10, 20, 50, 100),
  R = 200L,
  approach = "C",
  marginalize_args = list(),
  level = 0.95,
  seed = 42L,
  cores = 1L,
  verbose = TRUE
)
```

Arguments

fit	An nhpp_fit object.
data	The data frame used to fit fit. Must contain column y and all covariates.
TRs	Numeric vector of return periods. Default c(2,5,10,20,50,100).
R	Integer. Number of bootstrap replicates. Default 200L.
approach	Character. Which marginalization approach to use for each replicate: "A", "B", or "C". Default "C".
marginalize_args	Named list of additional arguments passed to marginalize (e.g. n_boot, seed, scenarios, fixed, interactions).
level	Numeric. Confidence level. Default 0.95.
seed	Seed of the replicate streams, or NULL to derive it from the current random-number state. Default 42L.
cores	Positive integer number of worker processes. Default 1L.
verbose	Logical. Print progress. Default TRUE.

Details

Each replicate uses its own L'Ecuyer-CMRG random-number stream derived from seed, so the output does not depend on cores. The caller's random-number state is left untouched when seed is supplied.

Value

A data frame with columns TR, RL_est (the return level from the original fitted model under the requested approach), CI_low, CI_high, n_ok (number of successful replicates). The R x length(TRs) matrix of replicate return levels is attached as attribute "replicates".

See Also

bootstrap_coef, marginalize

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(1500))
df$y <- 10 + df$x + rexp(1500, 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = "x", penalty = "none",
                verbose = FALSE)

bootstrap_rl(fit, df, TRs = c(10, 50), R = 50, approach = "A",
             verbose = FALSE)
```

---

build_cov_annual *Build a covariate data frame for marginalization*

Description

Constructs a data frame with n_obs rows containing all columns required by a fitted nhpp_fit model. The default seasonal columns (cos1, sen1, cos2, sen2) and any other harmonic column required by the model (cos<k>, sin<k>, sen<k>) are computed automatically with period (see seasonal_harmonics). Any other required column is filled from cov_vals, or set to zero with a warning if absent.

Usage

```
build_cov_annual(
  fit,
  cov_vals = list(),
  n_obs = NULL,
  period = NULL,
  interactions = list()
)
```

Arguments

fit	An nhpp_fit object.
cov_vals	Named list of covariate values. Each element is either a scalar (replicated to n_obs rows) or a vector of length n_obs.
n_obs	Integer number of rows (time points) to generate. If NULL, uses the rounded fit$obs_per_year value.
period	Numeric seasonal period in observations. If NULL, inherits fit$obs_per_year; legacy objects fall back to 365.25.
interactions	Named list of length-2 character vectors declaring interaction columns to compute. Each element is c("col_A", "col_B") and the interaction column is their product. The name of the list element becomes the column name. Example: list(AB = c("A", "B")) adds column AB = A * B.

Value

A data frame with `n_obs` rows and all columns required by `fit`.

Examples

```
set.seed(1)
n <- 3650
df <- cbind(data.frame(x = rnorm(n)), seasonal_harmonics(rep(1:365, 10)))
df$y <- 10 + df$x + 3 * df$cos1 + rexp(n, 1 / 3)
fit <- fit_nhpp(df, threshold = 20, loc_vars = c("x", "cos1", "sen1"),
                free_vars = c("cos1", "sen1"), penalty = "none",
                verbose = FALSE)
head(build_cov_annual(fit, list(x = 0.5), n_obs = 365))
```

`build_design_matrices` *Build design matrices for the NHPP point process model*

Description

Build design matrices for the NHPP point process model

Usage

```
build_design_matrices(
  df,
  loc_vars = NULL,
  scale_vars = NULL,
  shape_vars = NULL,
  free_vars = NULL
)
```

Arguments

`df`	A data frame containing all required columns.
`loc_vars`	Character vector of covariate names for the location (mu) parameter.
`scale_vars`	Character vector of covariate names for the scale (sigma) parameter.
`shape_vars`	Character vector of covariate names for the shape (xi) parameter.
`free_vars`	Character vector of column names that should never be penalized (e.g. seasonality terms like `c("cos1", "sen1")`). Defaults to `NULL`.

Value

A list with elements:

X_mu	Design matrix for location.
X_sigma	Design matrix for scale.
X_xi	Design matrix for shape.
idx_pen_mu	Integer indices of penalized columns in X_mu.
idx_pen_sigma	Integer indices of penalized columns in X_sigma.
idx_pen_xi	Integer indices of penalized columns in X_xi.

Examples

```
df <- cbind(data.frame(y = rexp(10), x = rnorm(10)),
            seasonal_harmonics(1:10))
dm <- build_design_matrices(df, loc_vars = c("x", "cos1", "sen1"),
                            scale_vars = "x", free_vars = c("cos1", "sen1"))
colnames(dm$X_mu)
dm$idx_pen_mu
```

---

coef.nhpp_fit *Extract coefficients of a fitted nhpp_fit model*

Description

Extract coefficients of a fitted nhpp_fit model

Usage

```
## S3 method for class 'nhpp_fit'
coef(object, operational = FALSE, ...)
```

Arguments

object	An nhpp_fit object.
operational	Logical. If TRUE, return the operational estimator, in which penalized coefficients below the activity tolerance are set to zero. Default FALSE (smooth optimizer solution).
...	Ignored.

Value

Named numeric vector of coefficients.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
coef(fit)
```

`fit_nhpp` *Fit a non-homogeneous point process model for extremes*

Description

Fits a peaks-over-threshold point process model via penalized maximum likelihood with quasi-Newton optimization and exact analytical gradients. The location, log-scale, and shape parameters are linear in the supplied covariates; seasonal harmonics and other `free_vars` are never penalized.

Usage

```
fit_nhpp(
  df,
  threshold,
  loc_vars = NULL,
  scale_vars = NULL,
  shape_vars = NULL,
  free_vars = NULL,
  penalty = c("none", "lasso", "ridge", "elnet"),
  alpha = 0.5,
  lambda = "bic",
  penalize_shape = TRUE,
  obs_per_year = 365.25,
  active_tol = 0.01,
  lambda_scaling = c("common", "gradient"),
  penalty_weights = "none",
  adaptive_gamma = 1,
  lambda_grid = NULL,
  init = NULL,
  grad_tol = 5e-04,
  maxit = 10000L,
  calc_hessian = FALSE,
  verbose = TRUE
)
```

Arguments

`df`	A data frame containing the response and all covariates. Must have a column named y (the observations).

threshold	Numeric scalar. The extreme value threshold u.
loc_vars	Character vector of covariate names for the location parameter. NULL for stationary location.
scale_vars	Character vector of covariate names for the log-scale parameter. NULL for stationary scale.
shape_vars	Character vector of covariate names for the shape parameter. NULL for stationary shape.
free_vars	Character vector of covariate names that are never penalized (e.g. seasonality terms). NULL by default.
penalty	Character. Penalty type: "none" (pure MLE), "lasso" (L1), "ridge" (L2), or "elnet" (elastic net, requires alpha).
alpha	Numeric in [0, 1]. Elastic-net mixing parameter. Only used when penalty = "elnet". 1 = LASSO, 0 = ridge. Scalar or named vector c(mu = , sigma = , xi =).
lambda	Numeric scalar or named vector c(mu=, sigma=, xi=), or "bic" to select automatically by BIC along a regularization path. Ignored when penalty = "none".
penalize_shape	Logical. Penalize shape parameter covariates? Default TRUE.
obs_per_year	Numeric. Observations per year. E.g. 365.25 for daily data, 365.25 / 7 for weekly. Default 365.25.
active_tol	Positive numeric tolerance used for BIC complexity counting, active-covariate extraction, and default coefficient reporting. Default 1e-2.
lambda_scaling	Character. Scaling used when lambda = "bic": "common" (default) applies the same scalar lambda to all parameter blocks; "gradient" calibrates the scale and shape penalties relative to the location-block score magnitude at the initial model. Ignored for numeric lambda.
penalty_weights	Coordinate weights multiplying lambda: "none" (default, unit weights), "adaptive" for the adaptive LASSO with weights $1/\lvert\tilde{\theta}_k\rvert^{\gamma}$ computed from an initial unpenalized fit (ridge fallback), or a non-negative numeric vector with one entry per parameter (names, if present, must match the parameter names). Weights of unpenalized coordinates are ignored.
adaptive_gamma	Positive exponent γ of the adaptive weights. Default 1.
lambda_grid	Optional decreasing numeric vector of base penalties used for BIC selection instead of the default two-phase grid.
init	Optional numeric starting vector, one entry per parameter.
grad_tol	Positive tolerance for the KKT violation divided by the number of exceedances. Default 5e-4.
maxit	Integer. Maximum optimizer iterations. Default 10000L.
calc_hessian	Logical. Compute the Hessian of the penalized objective at the solution? Default FALSE.
verbose	Logical. Print progress during BIC grid search and convergence warnings. Default TRUE.

Details

Convergence is assessed with the Karush-Kuhn-Tucker (KKT) conditions of the smoothed elastic-net objective: unpenalized and active coordinates must have a near-zero gradient, while penalized coordinates numerically at zero must satisfy the subgradient condition. The violation is scaled by the number of exceedances and compared with `grad_tol`.

When `lambda = "bic"`, the regularization path is evaluated on a coarse grid followed by a refined grid around the provisional optimum (or on a user-supplied `lambda_grid`). The returned model is the path solution at the selected penalty, so the reported fit is exactly the one scored by the BIC.

Value

An object of class `nhpp_fit`. Besides the smooth optimizer solution (`par`) and the operational activity-thresholded solution (`par_oper`), it stores the penalty weights, the scaled KKT violation (`kkt`), and, for `lambda = "bic"`, the evaluated regularization path (`path`).

See Also

`marginalize`, `bootstrap_coef`, `plot.nhpp_fit`

Examples

```
set.seed(42)
n <- 2000
df <- data.frame(x1 = rnorm(n), x2 = rnorm(n))
df$y <- 10 + 2 * df$x1 + rexp(n, 1 / 3)

# Unpenalized stationary-shape model
fit0 <- fit_nhpp(df, threshold = 18, loc_vars = c("x1", "x2"),
                 penalty = "none", verbose = FALSE)
coef(fit0)


# Adaptive LASSO with BIC-selected penalty
fit <- fit_nhpp(df, threshold = 18, loc_vars = c("x1", "x2"),
                penalty = "lasso", lambda = "bic",
                penalty_weights = "adaptive", verbose = FALSE)
summary(fit)
```

`fit_var_generator` *Fit a stationary VAR generator of covariate trajectories*

Description

Deseasonalizes each base variable by OLS on `n_harmonics` Fourier harmonic pairs, standardizes the residuals, and fits a stable vector autoregression VAR(p) without intercept by least squares, with the order selected by the Schwarz criterion on a common estimation sample (the convention of `vars::VARselect`). A univariate base variable gives a stable AR(p) model.

Usage

```
fit_var_generator(
  fit,
  data,
  vars = NULL,
  period = NULL,
  lag_max = NULL,
  n_harmonics = 2L,
  derive = NULL,
  derive_lead = 0L
)
```

Arguments

`fit`	An `nhpp_fit` object.
`data`	Data frame containing the base variables (and, optionally, seasonal harmonic columns).
`vars`	Character vector of base variables. Default `NULL`: the active covariates of `fit` present in `data`.
`period`	Numeric seasonal period in observations. If `NULL`, inherits `fit$obs_per_year`; legacy objects fall back to `365.25`.
`lag_max`	Integer. Maximum VAR order considered. Default `NULL`: `min(5, floor(n / (3 K)))`.
`n_harmonics`	Positive integer number of harmonic pairs used for deseasonalization. Default `2`.
`derive`	Optional function taking a data frame of simulated base variables (plus the harmonic columns) for one trajectory and returning a data frame with the same number of rows containing the model covariates.
`derive_lead`	Non-negative integer number of leading observations simulated before each year so that `derive` has the history it needs (e.g. the window of a lag or rolling mean). Default `0`.

Details

By default the base variables are the active covariates of `fit`. When model covariates are deterministic transformations of a smaller set of base variables (lags, rolling means, differences, interactions, standardizations), supply the base variables in `vars` and a `derive` function: `simulate_covariates` then simulates the base variables and recomputes the model covariates on each trajectory, preserving their deterministic relations.

Seasonal harmonics present in `data` (`cos<k>`, `sen<k>` or `sin<k>`) are reused; otherwise they are computed from the row index, which assumes contiguous rows starting at the first position of a year.

Value

An object of class `nhpp_var_generator`, including the coefficient matrix, innovation covariance, root moduli and spectral radius, or `NULL` (with a warning) if there are no variables to model. An unstable selected VAR causes an error.

Examples

```
set.seed(1)
n <- 3650
t <- rep(1:365, 10)
x <- as.numeric(stats::arima.sim(list(ar = 0.7), n)) + 2 * cos(2 * pi * t / 365.25)
df <- cbind(data.frame(x = x), seasonal_harmonics(t))
df$y <- 10 + df$x + rexp(n, 1 / 3)
fit <- fit_nhpp(df, threshold = 20, loc_vars = "x", penalty = "none",
                verbose = FALSE)
gen <- fit_var_generator(fit, df)
gen$p_opt
gen$spectral_radius

# Model covariates derived from a base variable: the lagged value of x
df$x_lag1 <- c(NA, head(df$x, -1))
gen_d <- fit_var_generator(
  fit, df[-1, ], vars = "x", derive_lead = 1,
  derive = function(b) data.frame(x = b$x, x_lag1 = c(NA, head(b$x, -1))))
head(simulate_covariates(gen_d, n_mc = 2, seed = 1)[[1]])
```

`is_nhpp_fit` *Check if an object is an nhpp_fit*

Description

Check if an object is an nhpp_fit

Usage

```
is_nhpp_fit(x)
```

Arguments

`x`	Any R object.

Value

Logical.

Examples

```
is_nhpp_fit(list())
```

`marginalize` *Compute unconditional return levels by marginalizing over covariates*

Description

Given a fitted `nhpp_fit` model, computes return levels by integrating out covariate uncertainty under one or more approaches:

Usage

```
marginalize(
  fit,
  data,
  TRs = c(2, 5, 10, 20, 50, 100),
  n_obs = NULL,
  period = NULL,
  approaches = c("A", "C"),
  scenarios = NULL,
  mc_sample = NULL,
  interactions = list(),
  fixed = list(),
  year_col = "year",
  n_boot = 500L,
  seed = 2024L,
  tol_root = 1e-04,
  z_hi = NULL
)
```

Arguments

`fit`	An `nhpp_fit` object.
`data`	The data frame used to fit `fit`. Required for approach C and for computing `z_hi`. Must contain a column `year` (or the name given in `year_col`) and all active covariates.
`TRs`	Numeric vector of return periods in years. Default `c(2, 5, 10, 20, 50, 100)`.
`n_obs`	Integer observations per year. If `NULL`, uses the rounded `fit$obs_per_year` value.
`period`	Numeric seasonal period in observations. If `NULL`, inherits `fit$obs_per_year`.
`approaches`	Character vector. Which approaches to run. Any subset of `c("A", "B", "C")`. Default `c("A", "C")`.
`scenarios`	Named list of covariate scenarios for approach A. Each element is itself a named list of scalar covariate values, passed to `build_cov_annual`. If `NULL`, a single scenario named `"default"` with all active covariates set to zero is used.
`mc_sample`	List of data frames for approach B. Each element is one simulated year of covariates (`n_obs` rows). Required if `"B" %in% approaches`.

interactions	Named list of interactions passed to build_cov_annual. Default list().
fixed	Named list of scalar covariate values imposed on every scenario and trajectory, overriding values from scenarios, mc_sample, and data (e.g. list(trend = 1) to evaluate return levels at the end of the record). Default list().
year_col	Name of the year column in data. Default "year".
n_boot	Integer. Number of resampled years for approach C. Default 500L.
seed	Integer seed for approach C (the caller's random-number stream is preserved). Default 2024L.
tol_root	Numeric. Root-finding tolerance. Default 1e-4.
z_hi	Numeric. Upper search bound for return levels. If NULL, set to 3 * max(data$y, na.rm = TRUE).

Details

A - conditional Covariates fixed at user-specified scenarios (e.g. quantiles of observed distributions). Fast; useful for sensitivity analysis.

B - parametric Covariates drawn from a user-supplied Monte Carlo sample of complete annual trajectories (e.g. from simulate_covariates).

C - empirical Annual-block resampling of observed covariate years from data, preserving within-year temporal structure.

For each approach and return period T, the return level solves $G(z) = 1 - 1/T$, where G is the annual non-exceedance probability averaged over trajectories. For approaches B and C the conditional Monte Carlo standard error of the root, $\hat{s}_T/(\sqrt{n}\,\hat{g}_T)$, is reported in se_mc, with $\hat{s}_T$ the standard deviation of the pathwise probabilities at the root and $\hat{g}_T$ the mean pathwise derivative with respect to the level. It measures only finite-ensemble integration error, conditional on the fitted model and trajectory law.

Value

A data frame of class nhpp_rl with columns approach, scenario, TR, RL, and se_mc (NA for approach A). See plot.nhpp_rl and rl_table.

Examples

```
set.seed(1)
n_years <- 30; n_obs <- 50
df <- data.frame(x = rnorm(n_years * n_obs),
                 year = rep(seq_len(n_years), each = n_obs))
df$y <- 10 + 2 * df$x + rexp(nrow(df), 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = "x", penalty = "none",
                obs_per_year = n_obs, verbose = FALSE)
rl <- marginalize(fit, df, TRs = c(5, 10, 50),
                  scenarios = list(low = list(x = -1), high = list(x = 1)),
                  n_boot = 200)
rl
rl_table(rl)
```

nhpp_fit-methods *Log-likelihood, number of observations, and prediction for nhpp_fit*

Description

logLik() returns the unpenalized log-likelihood at the operational estimator, with degrees of freedom equal to the number of active parameters and nobs equal to the number of exceedances, so that AIC() and BIC() work; BIC(fit) equals bic_nhpp(fit). predict() is a thin wrapper around predict_params.

Usage

```
## S3 method for class 'nhpp_fit'
logLik(object, ...)

## S3 method for class 'nhpp_fit'
nobs(object, ...)

## S3 method for class 'nhpp_fit'
predict(object, newdata = NULL, operational = TRUE, ...)
```

Arguments

object	An nhpp_fit object.
...	Ignored.
newdata	Optional data frame of covariates (see predict_params).
operational	Logical; see predict_params.

Value

logLik(): an object of class logLik. nobs(): the number of exceedances. predict(): a list with components mu, sigma, xi.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
logLik(fit)
c(AIC = AIC(fit), BIC = BIC(fit), nobs = nobs(fit))
str(predict(fit, newdata = data.frame(x = c(-1, 0, 1))))
```

n_exceedances — *Extract the number of exceedances from a fitted model*

Description

Extract the number of exceedances from a fitted model

Usage

```
n_exceedances(fit, y)
```

Arguments

fit	An nhpp_fit object.
y	Numeric vector of observations. If missing, the series stored in the fitted object is used.

Value

Integer. Number of observations exceeding the threshold.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)))
fit <- fit_nhpp(df, threshold = 4, penalty = "none", verbose = FALSE)
n_exceedances(fit)
```

plot.nhpp_fit — *Plot diagnostics for a fitted nhpp_fit object*

Description

Plot diagnostics for a fitted nhpp_fit object

Usage

```
## S3 method for class 'nhpp_fit'
plot(
  x,
  type = c("intensity", "fitted", "path", "qq"),
  residuals = c("interarrival", "magnitude"),
  ...
)
```

Arguments

x	An nhpp_fit object.
type	Character. "intensity" (default): cumulative fitted exceedance intensity $\hat{\Lambda}(0, t)$; "fitted": time-varying parameter profiles; "path": BIC and coefficient paths along the regularization path (fits with lambda = "bic"); "qq": exponential Q-Q plot of transformed residuals with exact pointwise 95% bands from the Beta distribution of exponential order statistics.
residuals	Residual type for type = "qq"; see residuals.nhpp_fit.
...	Additional graphical parameters passed to plot.

Value

Invisibly returns NULL, called for side effects.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
plot(fit, type = "intensity")
plot(fit, type = "qq")
```

---

plot.nhpp_rl *Plot return-level curves from marginalize()*

Description

Draws return level against return period (logarithmic axis), one curve per approach and scenario. For approaches with a Monte Carlo standard error, dashed lines show ± 2 standard errors.

Usage

```
## S3 method for class 'nhpp_rl'
plot(x, ...)
```

Arguments

x	An nhpp_rl object returned by marginalize.
...	Additional graphical parameters passed to plot.

Value

Invisibly returns NULL, called for side effects.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(1500), year = rep(1:30, each = 50))
df$y <- 10 + 2 * df$x + rexp(1500, 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = "x", penalty = "none",
                obs_per_year = 50, verbose = FALSE)
plot(marginalize(fit, df, TRs = c(2, 5, 10, 50, 100), n_boot = 200))
```

---

pp_grad *Analytical gradient of the penalized negative log-likelihood*

Description

Analytical gradient of the penalized negative log-likelihood

Usage

```
pp_grad(
  par,
  dm,
  y,
  threshold,
  lambda = 0.1,
  alpha = 0.5,
  pen_xi = TRUE,
  eps_l1 = 1e-10,
  obs_per_year = 365.25,
  penalty_weights = NULL
)
```

Arguments

`par`	Numeric vector of parameters: c(beta_mu, beta_sigma, beta_xi).
`dm`	Design matrix list from `build_design_matrices`.
`y`	Numeric vector of observations.
`threshold`	Numeric scalar. The extreme value threshold u.
`lambda`	Penalty strength. Either a scalar or a vector `c(mu=, sigma=, xi=)` (named, or unnamed in that order).
`alpha`	Elastic-net mixing parameter in [0,1]. 1 = LASSO, 0 = ridge. Either a scalar or a vector `c(mu=, sigma=, xi=)`.
`pen_xi`	Logical. Should the shape parameter covariates be penalized?
`eps_l1`	Small positive constant for smooth L1 approximation.
`obs_per_year`	Number of observations per year (e.g. 365.25 for daily).

`penalty_weights`
: Optional non-negative numeric vector of length `length(par)` multiplying `lambda` coordinate by coordinate (e.g. adaptive-LASSO weights). `NULL` (default) means unit weights.

Value

Numeric vector of length `length(par)`.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
dm <- build_design_matrices(df, loc_vars = "x")
pp_grad(c(6, 0.1, log(1.5), 0.1), dm, df$y, threshold = 4, lambda = 0)
```

---

pp_nllh — *Penalized negative log-likelihood for the NHPP model*

Description

Evaluates the negative log-likelihood of the non-homogeneous Poisson process (point-process) representation of threshold exceedances, plus a smoothed elastic-net penalty on the penalized coefficients.

Usage

```
pp_nllh(
  par,
  dm,
  y,
  threshold,
  lambda = 0.1,
  alpha = 0.5,
  pen_xi = TRUE,
  eps_l1 = 1e-10,
  obs_per_year = 365.25,
  penalty_weights = NULL
)
```

Arguments

`par`	Numeric vector of parameters: c(beta_mu, beta_sigma, beta_xi).
`dm`	Design matrix list from `build_design_matrices`.
`y`	Numeric vector of observations.
`threshold`	Numeric scalar. The extreme value threshold u.

lambda	Penalty strength. Either a scalar or a vector `c(mu=, sigma=, xi=)` (named, or unnamed in that order).
alpha	Elastic-net mixing parameter in [0,1]. 1 = LASSO, 0 = ridge. Either a scalar or a vector `c(mu=, sigma=, xi=)`.
pen_xi	Logical. Should the shape parameter covariates be penalized?
eps_l1	Small positive constant for smooth L1 approximation.
obs_per_year	Number of observations per year (e.g. 365.25 for daily).
penalty_weights	Optional non-negative numeric vector of length `length(par)` multiplying `lambda` coordinate by coordinate (e.g. adaptive-LASSO weights). `NULL` (default) means unit weights.

Value

Scalar. The penalized negative log-likelihood value. Parameter vectors outside the point-process support return `1e9`.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
dm <- build_design_matrices(df, loc_vars = "x")
par <- c(6, 0.1, log(1.5), 0.1)
pp_nllh(par, dm, df$y, threshold = 4, lambda = 0)
pp_nllh(par, dm, df$y, threshold = 4, lambda = 1)
```

---

predict_params *Compute time-varying GEV parameters from a fitted model*

Description

Given an `nhpp_fit` object and a data frame of covariate values, returns the implied location, scale, and shape parameters at each row.

Usage

```
predict_params(fit, newdata = NULL, operational = TRUE)
```

Arguments

fit	An object of class `nhpp_fit`.
newdata	A data frame with one row per time point. Must contain all columns that appear in the model's design matrices (except the intercept). If `NULL`, returns the fitted values stored in `fit$fitted`.
operational	Logical. If `TRUE` (default), coefficients below the fitted activity tolerance are set to zero before prediction. Set to `FALSE` to use the unthresholded smooth optimizer solution.

Value

A list with three numeric vectors of length `nrow(newdata)`:

`mu`	Location parameter.
`sigma`	Scale parameter (always positive).
`xi`	Shape parameter.

Examples

```
set.seed(1)
df <- data.frame(x = rnorm(2000))
df$y <- 10 + 2 * df$x + rexp(2000, 1 / 3)
fit <- fit_nhpp(df, threshold = 18, loc_vars = "x", penalty = "none",
                verbose = FALSE)
predict_params(fit, data.frame(x = c(-1, 0, 1)))
```

`print.nhpp_fit` *Print a fitted nhpp_fit model*

Description

Print a fitted nhpp_fit model

Usage

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

Arguments

`x`	An `nhpp_fit` object.
`...`	Ignored.

Value

Invisibly returns `x`.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
print(fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
               verbose = FALSE))
```

print.summary.nhpp_fit

Print a summary of a fitted nhpp_fit model

Description

Print a summary of a fitted nhpp_fit model

Usage

```
## S3 method for class 'summary.nhpp_fit'
print(x, digits = 5L, ...)
```

Arguments

`x`	A `summary.nhpp_fit` object.
`digits`	Number of significant digits for coefficients.
`...`	Ignored.

Value

Invisibly returns `x`.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)))
print(summary(fit_nhpp(df, threshold = 4, penalty = "none",
                       verbose = FALSE)))
```

residuals.nhpp_fit

Transformed residuals of a fitted nhpp_fit model

Description

Two residual types are available, both standard exponential under a correctly specified model:

`"interarrival"` Time-rescaled inter-exceedance increments $Z_k = n_y^{-1} \sum_{j:t_{k-1} \le j < t_k} \hat{\tau}_u(t_j)$, $k = 2, \ldots, m$, where $\hat{\tau}_u(t)$ is the fitted exceedance rate at the threshold (Ogata, 1988; Smith and Shively, 1995).

`"magnitude"` Exponential transforms of the threshold excesses under the implied generalized Pareto law, $\xi^{-1} \log\{1 + \xi(y-u)/\beta(t)\}$ with $\beta(t) = \sigma(t) + \xi(t)\{u - \mu(t)\}$.

Usage

```
## S3 method for class 'nhpp_fit'
residuals(
  object,
  type = c("interarrival", "magnitude"),
  operational = TRUE,
  ...
)
```

Arguments

`object`	An `nhpp_fit` object fitted with `margEVT` 0.4.0 or later (the observed series is stored in the object).
`type`	Character, `"interarrival"` or `"magnitude"`.
`operational`	Logical. Use the operational estimator (default).
`...`	Ignored.

Value

Numeric vector of residuals.

References

Ogata, Y. (1988). Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83(401), 9–27.

Smith, R. L. and Shively, T. S. (1995). Point process approach to modeling trends in tropospheric ozone based on exceedances of a high threshold. *Atmospheric Environment*, 29(23), 3489–3499.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
z <- residuals(fit)
c(mean = mean(z), var = var(z))
```

`rl_table` *Wide-format return level table from marginalize() output*

Description

Converts the long-format output of `marginalize` into a wide table with one row per scenario and one column per return period.

Usage

```
rl_table(marg_result)
```

Arguments

marg_result	Data frame returned by marginalize.

Value

A wide data frame.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)),
                 year = rep(1991:2020, each = 11))
fit <- fit_nhpp(df, threshold = 4, penalty = "none", obs_per_year = 11,
                verbose = FALSE)
rl_table(marginalize(fit, df, TRs = c(10, 50), approaches = "A"))
```

---

seasonal_harmonics *Seasonal Fourier harmonics*

Description

Builds the seasonal harmonic columns used by margEVT. The k-th pair is $\cos(2\pi kt/P)$ and $\sin(2\pi kt/P)$, where t is the position of the observation within the year and P is the seasonal period. Using this function to build the fitting data guarantees that build_cov_annual reconstructs the same seasonal cycle during marginalization, where positions run from 1 to n_obs within each simulated or resampled year.

Usage

```
seasonal_harmonics(
  t,
  period = 365.25,
  n_harmonics = 2L,
  sine = c("sen", "sin")
)
```

Arguments

t	Numeric vector of within-year positions (e.g. day of year).
period	Positive seasonal period. Default 365.25.
n_harmonics	Positive integer number of harmonic pairs. Default 2.
sine	Prefix of the sine columns: "sen" (default, the historical margEVT naming) or "sin".

Value

A data frame with columns cos1, sen1 (or sin1), ..., one row per element of t.

Examples

```
h <- seasonal_harmonics(1:365, period = 365.25)
head(h)
names(seasonal_harmonics(1:52, period = 52, n_harmonics = 3, sine = "sin"))
```

---

simulate_covariates *Simulate stationary covariate trajectories from a VAR generator*

Description

Simulates n_mc independent annual trajectories. All trajectories are propagated simultaneously, so the cost grows with the trajectory length rather than with n_mc times the length. Each trajectory is initialized from the stationary Gaussian law of the companion state (default) or from zero followed by burn_in transitions. Seasonality is re-added at positions 1, ..., n_obs; if the generator has a derive function, model covariates are recomputed from the simulated base variables, using derive_lead additional leading observations that are discarded afterwards.

Usage

```
simulate_covariates(
  generator,
  n_mc,
  n_obs = NULL,
  burn_in = 300L,
  initialization = c("stationary", "burnin"),
  seed = NULL
)
```

Arguments

generator	Object from fit_var_generator. If NULL, returns NULL (stationary/no-covariate model).
n_mc	Integer. Number of independently simulated annual trajectories.
n_obs	Integer observations per simulated year. If NULL, uses the rounded seasonal period stored in generator.
burn_in	Integer. Burn-in applied when initialization = "burnin". Default 300L.
initialization	Character. "stationary" (default) draws the initial companion state from the fitted stationary Gaussian law; "burnin" starts at zero and applies burn_in transitions.
seed	Integer seed or NULL. When supplied, the caller's random-number stream is preserved. Default NULL (use the current stream).

Value

A list of length n_mc, each element an n_obs-row data frame of covariates, directly usable as mc_sample in marginalize (approach B).

Examples

```
set.seed(1)
n <- 3650
t <- rep(1:365, 10)
df <- cbind(data.frame(x = as.numeric(stats::arima.sim(list(ar = 0.6), n))),
            seasonal_harmonics(t))
df$y <- 10 + df$x + rexp(n, 1 / 3)
fit <- fit_nhpp(df, threshold = 20, loc_vars = "x", penalty = "none",
                verbose = FALSE)
mc <- simulate_covariates(fit_var_generator(fit, df), n_mc = 3, seed = 1)
length(mc); dim(mc[[1]])
```

---

summary.nhpp_fit *Summarise a fitted nhpp_fit model*

Description

Builds a structured summary of a fitted nhpp_fit object, including the penalty used, the active and shrunk coefficients, information criteria, convergence diagnostics, and (when available) Hessian conditioning. Coefficient standard errors are deliberately not reported: after penalization and data-driven selection they lack a valid interpretation (see bootstrap_coef for conditional intervals).

Usage

```
## S3 method for class 'nhpp_fit'
summary(object, tol = NULL, ...)
```

Arguments

object	An nhpp_fit object.
tol	Numeric. Coefficients smaller than this in absolute value are treated as zero. If NULL, inherits object$active_tol; legacy objects fall back to 1e-2.
...	Ignored.

Value

An object of class summary.nhpp_fit, printed by print.summary.nhpp_fit.

Examples

```
set.seed(1)
df <- data.frame(y = c(rexp(300), runif(30, 5, 10)), x = rnorm(330))
fit <- fit_nhpp(df, threshold = 4, loc_vars = "x", penalty = "none",
                verbose = FALSE)
summary(fit)
```

Index

active_covariates, 2

backtest, 3
bic_nhpp, 5, *19*
bootstrap_coef, 6, *8*, *14*, *30*
bootstrap_rl, *6*, *7*, 7
build_cov_annual, *4*, 9, *17*, *18*, *28*
build_design_matrices, 10, *22*, *23*

coef.nhpp_fit, 11

fit_nhpp, *5*, 12
fit_var_generator, 14, *29*

is_nhpp_fit, 16

logLik.nhpp_fit *(*nhpp_fit-methods*)*, 19

marginalize, *4*, *5*, *7*, *8*, *14*, 17, *21*, *27*, *28*, *30*

n_exceedances, 20
nhpp_fit-methods, 19
nobs.nhpp_fit *(*nhpp_fit-methods*)*, 19

plot, *21*
plot.nhpp_fit, *14*, 20
plot.nhpp_rl, *18*, 21
pp_grad, 22
pp_nllh, 23
predict.nhpp_fit *(*nhpp_fit-methods*)*, 19
predict_params, *19*, 24
print.nhpp_fit, 25
print.summary.nhpp_fit, 26, *30*

residuals.nhpp_fit, *21*, 26
rl_table, *18*, 27

seasonal_harmonics, *9*, 28
simulate_covariates, *15*, *18*, 29
summary.nhpp_fit, 30