

Package ‘margEVT’

October 23, 2026

Type Package

Title Regularized Point Processes and Stochastic Marginalization for
Extremes

Version 0.5.0

Description Implements a non-stationary extreme value analysis framework by coupling a covariate-driven non-homogeneous Poisson process (NHPP) with (adaptive) elastic-net regularization, analytical gradients and a lower bound on the shape parameter. 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/rodrigossqrt3/margEVT>

BugReports <https://github.com/rodrigossqrt3/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 2026-10-23 20:10:02 UTC

Contents

active_covariates	2
annual_prob	3
backtest	4
bic_nhpp	6
bootstrap_coef	7
bootstrap_nhpp	8
bootstrap_rl	10
build_cov_annual	12
build_design_matrices	13
coef.nhpp_fit	14
fit_nhpp	15
fit_var_generator	18
is_nhpp_fit	19
marginalize	20
nhpp_fit-methods	22
n_exceedances	23
plot.nhpp_fit	24
plot.nhpp_rl	25
pp_grad	25
pp_nllh	27
predict_params	28
print.nhpp_fit	29
print.summary.nhpp_fit	30
refit_nhpp	30
residuals.nhpp_fit	31
rl_table	32
seasonal_harmonics	33
simulate_covariates	34
summary.nhpp_fit	35
Index	37

active_covariates	<i>Extract names of non-seasonal covariates from a fitted model</i>
-------------------	---

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

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

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)
```

<code>annual_prob</code>	<i>Annual non-exceedance probability under a covariate law</i>
--------------------------	--

Description

Computes $G(z) = P(M \leq z)$, the probability that the annual maximum does not exceed z , averaged over annual covariate blocks: the quantity inverted by [marginalize](#) and used for probability integral transforms of observed annual maxima.

Usage

```
annual_prob(fit, z, covariates, n_obs = NULL)
```

Arguments

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>z</code>	Numeric vector of levels.
<code>covariates</code>	Annual covariate blocks with every column of the model (e.g. from build_cov_annual): a list of data frames with <code>n_obs</code> rows each, or one data frame stacking them.
<code>n_obs</code>	Integer observations per year. If <code>NULL</code> , the rounded <code>fit\$obs_per_year</code> .

Value

Numeric vector of probabilities, one per element of `z`.

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",
               obs_per_year = 50, verbose = FALSE)
blocks <- lapply(c(-1, 0, 1), function(v) build_cov_annual(fit, list(x = v)))
annual_prob(fit, c(20, 25, 30), blocks)
```

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,
  approaches = "C",
  generator_args = list(),
  decluster = NULL,
  cores = 1L,
  verbose = TRUE
)
```

Arguments

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>data</code>	The data frame used to fit <code>fit</code> . Must contain columns <code>y</code> , <code>year</code> (or <code>year_col</code>), and all active covariates.
<code>varname</code>	Character. Name of the column in <code>data</code> containing the raw (non-declustered) series, used to compute observed annual maxima for validation.
<code>TRs</code>	Numeric vector of return periods to evaluate. Default <code>c(2, 5, 10)</code> .
<code>n_obs</code>	Integer observations per year. If <code>NULL</code> , uses the rounded <code>fit\$obs_per_year</code> value.
<code>period</code>	Numeric seasonal period in observations. If <code>NULL</code> , inherits <code>fit\$obs_per_year</code> .
<code>window_years</code>	Integer. Width of each validation window in years. Default 5L.
<code>min_train_years</code>	Integer. Minimum number of training years before the first validation window. Default 10L.
<code>n_boot</code>	Integer. Resampled years (approach C) or simulated trajectories (approach B) for covariate marginalization within each window. Default 200L.
<code>interactions</code>	Named list of interactions passed to <code>build_cov_annual</code> . Default <code>list()</code> .
<code>year_col</code>	Character. Name of the year column. Default "year".
<code>min_obs_year</code>	Integer. Minimum observations in a validation year for it to be included. Default <code>as.integer(n_obs * 0.8)</code> .
<code>seed</code>	Integer. Seed used for empirical annual-block sampling. Default 2024L. Each validation window uses a distinct derived seed.
<code>trend_col</code>	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 <code>NULL</code> (no trend).
<code>fixed</code>	Named list of scalar covariate values imposed on every covariate block (see <code>marginalize</code>). Default <code>list()</code> .
<code>reselect_lambda</code>	Logical. If <code>TRUE</code> , the penalty is re-selected by BIC in every training window; otherwise the original <code>lambda</code> is kept. Adaptive penalty weights are always recomputed within each training window, so the weights do not use validation data. Default <code>FALSE</code> .
<code>approaches</code>	Character vector, any of "B" and "C": how the covariates are integrated out in each window. "C" (default) resamples observed training years; "B" fits a VAR generator (<code>fit_var_generator</code>) to the training years only and uses <code>n_boot</code> simulated trajectories. Both are evaluated on the same window refit. The first element defines the legacy columns <code>F_ann</code> and <code>exc_T<TR></code> .
<code>generator_args</code>	Named list of further arguments for <code>fit_var_generator</code> in approach B (e.g. <code>vars</code> , <code>derive</code> , <code>derive_lead</code>). Default <code>list()</code> .
<code>decluster</code>	Optional function applied to the training-year values of <code>varname</code> (in row order) in each window; its result replaces <code>y</code> for the refit, so declustering uses training data only. Default <code>NULL</code> (use <code>data\$y</code> as supplied).

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 and, for each approach a, the PIT value $F_{\text{ann}}\langle a \rangle$ and exceedance indicators $\text{exc}_{\langle a \rangle_T\langle \text{TR} \rangle}$; F_{ann} and $\text{exc}_{T\langle \text{TR} \rangle}$ repeat those of the first approach.
binom_tests	Data frame with nominal binomial comparisons for each approach and TR. The <code>not_rejected</code> field is the recommended interpretation; <code>calibrated</code> is retained as a legacy alias.
ks_test	Data frame with the exploratory tail-conditional PIT KS comparison for each approach, including <code>not_rejected</code> and the legacy <code>calibrated</code> 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 $\text{BIC}(\text{fit})$ for fits produced by `margEVT` 0.4.0 or later.

Usage

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

Arguments

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

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, λ , 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,
  occurrence = c("fixed", "simulated"),
  cores = 1L,
  verbose = TRUE
)
```

Arguments

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>data</code>	The data frame used to fit <code>fit</code> . Must contain column <code>y</code> and all covariates.
<code>R</code>	Integer. Number of bootstrap replicates. Default 200L.
<code>level</code>	Numeric. Confidence level. Default 0.95.
<code>seed</code>	Seed of the replicate streams, or NULL to derive it from the current random-number state. Default 42L.
<code>occurrence</code>	"fixed" (default) keeps the observed number and timing of exceedances and resamples only their sizes (a bootstrap conditional on the occurrence pattern); "simulated" draws, in every period, an exceedance with probability $1 - \exp(-\tau_t/n_y)$ under the fitted model before drawing its size, so the uncertainty about the exceedance rate is also propagated.
<code>cores</code>	Positive integer number of worker processes. Default 1L.
<code>verbose</code>	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_nhpp

Parametric bootstrap for return levels and coefficients in one pass

Description

Runs the conditional parametric bootstrap of [bootstrap_rl](#) and [bootstrap_coef](#) with a single set of `R` refits, returning percentile intervals for the return levels (one marginalization approach) and for every operational coefficient. Use it instead of calling both functions, which would refit twice.

Usage

```
bootstrap_nhpp(
  fit,
  data,
  TRs = c(2, 5, 10, 20, 50, 100),
  R = 200L,
  approach = "C",
  marginalize_args = list(),
  level = 0.95,
  seed = 42L,
  occurrence = c("fixed", "simulated"),
  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	Return periods, or NULL for coefficient intervals only.
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.
occurrence	"fixed" (default) keeps the observed number and timing of exceedances and resamples only their sizes (a bootstrap conditional on the occurrence pattern); "simulated" draws, in every period, an exceedance with probability $1 - \exp(-\tau_t/n_y)$ under the fitted model before drawing its size, so the uncertainty about the exceedance rate is also propagated.
cores	Positive integer number of worker processes. Default 1L.
verbose	Logical. Print progress. Default TRUE.

Value

A list with r1 (data frame as in bootstrap_r1, or NULL), coef (data frame as in bootstrap_coef, with the share of replicates at zero in p_zero), and the replicate matrices r1_replicates and coef_replicates.

See Also

[bootstrap_r1](#), [bootstrap_coef](#)

Examples

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

b <- bootstrap_nhpp(fit, df, TRs = c(10, 50), R = 30, verbose = FALSE)
b$rl
b$coef
```

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,
  occurrence = c("fixed", "simulated"),
  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.
occurrence	"fixed" (default) keeps the observed number and timing of exceedances and resamples only their sizes (a bootstrap conditional on the occurrence pattern); "simulated" draws, in every period, an exceedance with probability $1 - \exp(-\tau_t/n_y)$ under the fitted model before drawing its size, so the uncertainty about the exceedance rate is also propagated.
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 $R \times \text{length}(\text{TRs})$ 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	<i>Build a covariate data frame for marginalization</i>
------------------	---

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

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

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 (μ) parameter.
scale_vars	Character vector of covariate names for the scale (σ) parameter.
shape_vars	Character vector of covariate names for the shape (ξ) 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,
  shape_lower = -0.5,
  maxit = 10000L,
  calc_hessian = FALSE,
  scaling = c("auto", "none"),
  verbose = TRUE
)
```

Arguments

<code>df</code>	A data frame containing the response and all covariates. Must have a column named <code>y</code> (the observations).
<code>threshold</code>	Numeric scalar. The extreme value threshold u .
<code>loc_vars</code>	Character vector of covariate names for the location parameter. <code>NULL</code> for stationary location.
<code>scale_vars</code>	Character vector of covariate names for the log-scale parameter. <code>NULL</code> 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 $1/ \hat{\theta}_k ^\gamma$ 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.
shape_lower	Lower bound imposed on the shape parameter ξ_t at every observation, or -Inf for no bound. The default -0.5 restricts estimation to the region in which the maximum likelihood estimator has its usual asymptotic properties (Smith, 1985); below -1 the point-process likelihood is unbounded. With a shape design made of an intercept and at most one covariate the bound is handled exactly (box-constrained optimization, KKT conditions with multipliers); with more shape covariates it acts as a barrier. Whether the bound is active at the solution is stored in shape_at_bound.
maxit	Integer. Maximum optimizer iterations. Default 10000L.
calc_hessian	Logical. Compute the Hessian of the penalized objective at the solution? Default FALSE.

scaling	Parameter scaling for the optimizer: "auto" (default) passes <code>optim()</code> the scales $1/\sqrt{ H_{kk} }$, with H_{kk} the diagonal curvature of the unpenalized negative log-likelihood at the starting values, computed once per call and used along the whole regularization path; "none" reproduces the unscaled optimization of earlier versions. Scaling changes the path of the optimizer, not the problem: every reported fit satisfies the same KKT conditions, and estimates differ only within the optimizer tolerance.
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

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>data</code>	Data frame containing the base variables (and, optionally, seasonal harmonic columns).
<code>vars</code>	Character vector of base variables. Default <code>NULL</code> : the active covariates of <code>fit</code> present in <code>data</code> .
<code>period</code>	Numeric seasonal period in observations. If <code>NULL</code> , inherits <code>fit\$obs_per_year</code> ; legacy objects fall back to 365.25.
<code>lag_max</code>	Integer. Maximum VAR order considered. Default <code>NULL</code> : <code>min(5, floor(n / (3 K)))</code> .
<code>n_harmonics</code>	Positive integer number of harmonic pairs used for deseasonalization. Default 2.
<code>derive</code>	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.
<code>derive_lead</code>	Non-negative integer number of leading observations simulated before each year so that <code>derive</code> 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

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>data</code>	The data frame used to fit <code>fit</code> . Required for approach C and for computing <code>z_hi</code> . Must contain a column <code>year</code> (or the name given in <code>year_col</code>) and all active covariates.
<code>TRs</code>	Numeric vector of return periods in years. Default <code>c(2, 5, 10, 20, 50, 100)</code> .

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 $3 * \max(\text{data\$y}, \text{na.rm} = \text{TRUE})$.

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_r1 with columns approach, scenario, TR, RL, and se_mc (NA for approach A). See [plot.nhpp_r1](#) 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)
r1 <- marginalize(fit, df, TRs = c(5, 10, 50),
                 scenarios = list(low = list(x = -1), high = list(x = 1)),
                 n_boot = 200)

r1
r1_table(r1)
```

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 <code>predict_params</code>).
operational	Logical; see <code>predict_params</code> .

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	<i>Extract the number of exceedances from a fitted model</i>
---------------	--

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 <code>lambda = "bic"</code>); "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	<i>Plot return-level curves from marginalize()</i>
--------------	--

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	<i>Analytical gradient of the penalized negative log-likelihood</i>
---------	---

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,
  shape_lower = -Inf
)
```

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.
shape_lower	Lower bound for the shape parameter at every period; parameter vectors with some ξ_t below it return 1e9. Default -Inf (no bound).

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,
  shape_lower = -Inf
)
```

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.
shape_lower	Lower bound for the shape parameter at every period; parameter vectors with some ξ_t below it return 1e9. Default -Inf (no bound).

Details

The integrated intensity is $\Lambda = n_y^{-1} \sum_t [1 + \xi_t(u - \mu_t)/\sigma_t]_+^{-1/\xi_t}$, so that, when $\xi_t < 0$ and the upper endpoint $\mu_t - \sigma_t/\xi_t$ lies below u , period t contributes zero (the intensity vanishes above the threshold).

Value

Scalar. The penalized negative log-likelihood value. Parameter vectors outside the point-process support (an exceedance beyond the upper endpoint, or a threshold below the lower endpoint) 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	<i>Print a fitted nhpp_fit model</i>
----------------	--------------------------------------

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)))
```

```
refit_nhpp
```

Refit an NHPP model on new data with the same specification

Description

Refits fit on data with the same design (location, scale and shape covariates, free columns), penalty, activity tolerance, KKT tolerance and shape bound. By default the penalty level and the penalty weights are those of fit; `lambda = "bic"` re-selects the penalty and `penalty_weights = "adaptive"` recomputes adaptive weights on data (as needed, e.g., in walk-forward validation).

Usage

```
refit_nhpp(
  fit,
  data,
  lambda = fit$lambda,
  penalty_weights = NULL,
  init = fit$par
)
```

Arguments

<code>fit</code>	An <code>nhpp_fit</code> object.
<code>data</code>	Data frame with column <code>y</code> and every covariate of <code>fit</code> .
<code>lambda</code>	Penalty level: the fitted one (default) or "bic".
<code>penalty_weights</code>	NULL keeps the weights of <code>fit</code> ; "adaptive" recomputes them; a numeric vector supplies new ones.
<code>init</code>	Starting values. Default: the estimate of <code>fit</code> .

Value

An `nhpp_fit` object, or NULL if the refit failed. Check `$converged` (KKT certification) before using it.

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)
df2 <- df; df2$y <- 10 + df2$x + rexp(1500, 1 / 3)
coef(refit_nhpp(fit, df2))
```

<code>residuals.nhpp_fit</code>	<i>Transformed residuals of a fitted nhpp_fit model</i>
---------------------------------	---

Description

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

"interarrival" Time-rescaled inter-exceedance increments $Z_k = n_y^{-1} \sum_{j:t_{k-1} \leq j < t_k} \hat{\tau}_u(t_j)$, $k = 2, \dots, m$, 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	<i>Seasonal Fourier harmonics</i>
--------------------	-----------------------------------

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

<code>t</code>	Numeric vector of within-year positions (e.g. day of year).
<code>period</code>	Positive seasonal period. Default 365.25.
<code>n_harmonics</code>	Positive integer number of harmonic pairs. Default 2.
<code>sine</code>	Prefix of the sine columns: "sen" (default, the historical <code>margEVT</code> 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	<i>Simulate stationary covariate trajectories from a VAR generator</i>
---------------------	--

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

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

<code>object</code>	An <code>nhpp_fit</code> object.
<code>tol</code>	Numeric. Coefficients smaller than this in absolute value are treated as zero. If <code>NULL</code> , inherits <code>object\$active_tol</code> ; legacy objects fall back to <code>1e-2</code> .
<code>...</code>	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](#)
annual_prob, [3](#)

backtest, [4](#)
bic_nhpp, [6](#), [22](#)
bootstrap_coef, [7](#), [8](#), [9](#), [11](#), [17](#), [35](#)
bootstrap_nhpp, [8](#)
bootstrap_rl, [7–9](#), [10](#)
build_cov_annual, [3](#), [5](#), [12](#), [21](#), [33](#)
build_design_matrices, [13](#), [26](#), [27](#)

coef.nhpp_fit, [14](#)

fit_nhpp, [6](#), [15](#)
fit_var_generator, [5](#), [18](#), [34](#)

is_nhpp_fit, [19](#)

logLik.nhpp_fit (nhpp_fit-methods), [22](#)

marginalize, [3](#), [5](#), [6](#), [9–11](#), [17](#), [20](#), [25](#), [32](#), [33](#),
[35](#)

n_exceedances, [23](#)
nhpp_fit-methods, [22](#)
nobs.nhpp_fit (nhpp_fit-methods), [22](#)

plot, [24](#), [25](#)
plot.nhpp_fit, [17](#), [24](#)
plot.nhpp_rl, [21](#), [25](#)
pp_grad, [25](#)
pp_nllh, [27](#)
predict.nhpp_fit (nhpp_fit-methods), [22](#)
predict_params, [22](#), [28](#)
print.nhpp_fit, [29](#)
print.summary.nhpp_fit, [30](#), [35](#)

refit_nhpp, [30](#)
residuals.nhpp_fit, [24](#), [31](#)
rl_table, [21](#), [32](#)

seasonal_harmonics, [12](#), [33](#)
simulate_covariates, [19](#), [21](#), [34](#)
summary.nhpp_fit, [35](#)