Package ‘exdqlm’

June 5, 2026

Title Extended Dynamic Quantile Linear Models

Version 1.0.0

Author Raquel Barata [aut, cre],
Raquel Prado [ths],
Bruno Sanso [ths],
Antonio Aguirre [aut]

Maintainer Raquel Barata <raquel.a.barata@gmail.com>

Description Bayesian quantile-regression routines for dynamic state-space models and static regression under the extended asymmetric Laplace (exAL) error distribution. The dynamic state-space models are extended dynamic quantile linear models (exDQLMs). The package combines dynamic exDQLM inference via LDVB, MCMC, and legacy ISVB with static exAL regression via LDVB and MCMC, reduced AL/DQLM paths through fixed skewness, component builders for trend/seasonality/regression blocks, static shrinkage priors including ridge, regularized horseshoe, and 'rhs_ns', evidence lower bound diagnostics, optional C++ accelerators, and posterior predictive synthesis across separately fitted quantiles through 'quantileSynthesis()'. Dynamic exDQLM methods are described in Barata et al. (2020) <doi:10.1214/21-AOAS1497>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports stats, methods, graphics, coda, tictoc, magic, crch, truncnorm, LaplacesDemon, grDevices, Rcpp, matrixStats, nimble, numDeriv

Suggests rmarkdown, testthat (>= 3.0.0), ggplot2, pkgload, MASS

Config/testthat/edition 3

LinkingTo BH, Rcpp, RcppArmadillo, RcppEigen

SystemRequirements C++17; OpenMP (optional)

URL https://github.com/AntonioAPDL/exdqlm

BugReports https://github.com/AntonioAPDL/exdqlm/issues

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-06-04 23:30:02 UTC

Contents

exdqlm-package *exdqlm: Extended Dynamic Quantile Linear Models*

Description

Bayesian quantile-regression tools for dynamic state-space models and static regression under the extended asymmetric Laplace error distribution (exAL).

Details

The package centers on native dynamic quantile state-space modeling for univariate time series and also provides a static exAL regression workflow. Across these settings, `exdqlm` combines model construction helpers, multiple Bayesian inference engines, shrinkage priors for static coefficients, and post hoc synthesis of several fitted quantiles.

Main workflows

- Dynamic/state-space quantile modeling via `exdqlmLDVB()` and `exdqlmMCMC()`, with legacy `exdqlmISVB()` retained for backward compatibility and transfer-function extensions through `exdqlmTransferLDVB()`, `exdqlmTransferMCMC()`, and legacy `exdqlmTransferISVB()`.
- Static Bayesian exAL regression via `exalStaticLDVB()` and `exalStaticMCMC()`, with static fitted-quantile and coefficient summaries through `exalStaticDiagnostics()`.
- Modular state-space construction via `polytrendMod()`, `seasMod()`, and `regMod()`.
- Multi-quantile post-processing via `quantileSynthesis()` for post hoc posterior-predictive synthesis from separately fitted quantiles into a unified predictive distribution.

Distinctive features

- Dynamic Bayesian quantile state-space inference with LDVB as the main VB engine, MCMC for posterior simulation, and legacy ISVB retained for compatibility and historical comparisons.
- A unified package covering both dynamic exDQLM models and static exAL regression.
- Static shrinkage priors including ridge, regularized horseshoe (`"rhs"`), and `rhs_ns`.
- Reduced AL/DQLM paths through `dqlm.ind = TRUE` in both dynamic and static APIs.
- Standardized VB diagnostics traces via `fit$diagnostics$vb_trace` for ELBO, `sigma`, `gamma`, and convergence deltas across VB engines.
- Conservative automatic warmup defaults for the most failure-prone shared blocks: RHS-family `tau` scheduling plus exAL `(sigma, gamma)` warmup in VB and MCMC entry points, with explicit controls available only when users need to override the defaults.
- Optional C++ acceleration for selected state-space computations.

Release changes in 1.0.0

- Dynamic diagnostics report CRPS through a finite integrated quantile-score approximation over posterior predictive empirical quantiles, with user-configurable quantile levels and weights in `exdqlmDiagnostics()`.
- Held-out forecast diagnostics are available for forecast objects through `exdqlmForecastDiagnostics()`.
- Static diagnostics store fitted-quantile summaries and coefficient intervals, with `plot(..., type = "coefficients")` available for comparing static LDVB/MCMC coefficient summaries.
- Dynamic KL normality diagnostics are deterministic for fixed fitted objects and no longer depend on stochastic reference samples. The top-level diagnostic object exposes `KL` as the primary calibration diagnostic and keeps advanced KL sensitivity details under `kl.details`.

Runtime options

- `options(exdqlm.use_cpp_kf = TRUE|FALSE)` – C++ Kalman bridge (optional; default TRUE).
- `options(exdqlm.compute_elbo = TRUE|FALSE)` – Compute ELBO (optional; default TRUE).
- `options(exdqlm.tol_elbo = numeric)` – Positive ELBO convergence tolerance used when `exdqlm.compute_elbo = TRUE`; smaller values enforce stricter ELBO stabilization checks (default `1e-6`).
- `options(exdqlm.use_cpp_builders = TRUE|FALSE)` – C++ model builders (optional; default FALSE).
- `options(exdqlm.use_cpp_samplers = TRUE|FALSE)` – C++ samplers (optional; default FALSE).
- `options(exdqlm.use_cpp_postpred = TRUE|FALSE)` – C++ posterior predictive sampler (optional; default FALSE).
- `options(exdqlm.use_cpp_mcmc = TRUE|FALSE)` – MCMC backend routing (optional; default TRUE).
- `options(exdqlm.cpp_mcmc_mode = "strict"|"fast")` – strict keeps legacy R-kernel parity; fast enables C++ FFBS in MCMC (default "fast").
- `options(exdqlm.cpp_threads = numeric)` – Positive integer thread cap for eligible OpenMP-enabled C++ paths (`1L` forces single-thread; default `1L`).

Author(s)

Maintainer: Raquel Barata <raquel.a.barata@gmail.com>

Authors:

- Raquel Barata <raquel.a.barata@gmail.com>
- Antonio Aguirre

Other contributors:

- Raquel Prado [thesis advisor]
- Bruno Sanso [thesis advisor]

See Also

Useful links:

- https://github.com/AntonioAPDL/exdqlm
- Report bugs at https://github.com/AntonioAPDL/exdqlm/issues

---

+.exdqlm *Addition for* `exdqlm` *objects*

Description

Combines two state space blocks into a single state space model for an exDQLM.

Usage

```
## S3 method for class 'exdqlm'
m1 + m2
```

Arguments

m1	object of class "exdqlm" containing the first model to be combined.
m2	object of class "exdqlm" containing the second model to be combined.

Value

An object of class "exdqlm" containing the new combined state space model components:

- FF - Observational vector.
- GG - Evolution matrix.
- m0 - Prior mean of the state vector.
- C0 - Prior covariance of the state vector.

Examples

```
trend.comp = polytrendMod(2, rep(0, 2), 10*diag(2))
seas.comp = seasMod(365, c(1,2,4), C0 = 10*diag(6))
model = trend.comp + seas.comp
```

as.exdqlm *exdqlm objects*

Description

as.exdqlm attempts to turn a list into an exdqlm object. Works for time-invariant dlm objects created using the **dlm** package.

Usage

```
as.exdqlm(m)
```

Arguments

m a list containing named elements m0, C0, FF and GG.

Value

An object of class "exdqlm" containing the state space model components:

- FF - Observational vector.
- GG - Evolution matrix.
- m0 - Prior mean of the state vector.
- C0 - Prior covariance of the state vector.

BTflow *Monthly streamflow at the Big Trees gauge*

Description

Observed monthly mean streamflow, in cubic feet per second, at the Big Trees gauge on the San Lorenzo River in Santa Cruz County, California. The monthly values were obtained by averaging the daily USGS streamflow observations within each calendar month.

Usage

```
BTflow
```

Format

A monthly time series (ts) of length 471, spanning January 1987 through March 2026.

Source

U.S. Geological Survey, National Water Information System (USGS Water Data for the Nation), https://waterdata.usgs.gov/.

References

U.S. Geological Survey (2016). National Water Information System data available on the World Wide Web (USGS Water Data for the Nation). https://waterdata.usgs.gov/.

`climateIndices` *Monthly climate indices for streamflow examples*

Description

Monthly atmospheric and oceanic climate indices used as external predictors in the Big Trees streamflow examples. Values are stored on their original scales; examples that combine indices standardize the relevant columns within the analysis code.

Usage

```
climateIndices
```

Format

A data frame with 516 rows and 3 variables:

date First day of the calendar month.

noi Northern Oscillation Index.

amo Atlantic Multidecadal Oscillation index.

The data frame spans January 1980 through December 2022.

Source

Compiled from the NOAA Physical Sciences Laboratory monthly climate indices collection, https://psl.noaa.gov/data/climateindices/.

`compPlot` *Plot a component of an exDQLM*

Description

The function plots the dynamic MAP estimates and 95% credible intervals (CrIs) of a specified component of an exDQLM. Alternatively, if `just.theta=TRUE` the MAP estimates and 95% credible intervals (CrIs) of a single element of the dynamic state vector are plotted.

Usage

```
compPlot(
  m1,
  index,
  add = FALSE,
  col = "purple",
  just.theta = FALSE,
  cr.percent = 0.95,
  plot = TRUE,
  xlim = NULL,
  ylim = NULL,
  xlab = "time",
  ylab = NULL,
  lwd = 1.5,
  lwd.interval = 0.75,
  lty.interval = 2
)
```

Arguments

`m1`	An object of class "`exdqlmLDVB`", "`exdqlmMCMC`", or legacy "`exdqlmISVB`".
`index`	Vector of consecutive integers in $\{1, \ldots, q\}$ indicating the component or element of the state vector to be plotted.
`add`	Logical value indicating whether the dynamic component will be added to existing plot. Default is `FALSE`.
`col`	Character vector of length 1 giving color of the dynamic component to be plotted. Default is `purple`.
`just.theta`	Logical; if `TRUE`, the function plots the dynamic distribution of the `index` element of the state vector. If `just.theta=TRUE`, `index` must have length 1.
`cr.percent`	Numeric in `(0, 1)` indicating the probability mass for the credible intervals (e.g., `0.95`). Default `0.95`.
`plot`	Logical value indicating whether to draw the plot. If `FALSE`, the function only returns the plotted summaries. Default is `TRUE`.
`xlim`, `ylim`	Optional limits passed to the base plotting call when `plot = TRUE`.
`xlab`, `ylab`	Optional axis labels passed to the base plotting call when `plot = TRUE`.
`lwd`, `lwd.interval`	Line widths for the dynamic component and credible interval bounds, respectively.
`lty.interval`	Line type for the credible interval bounds.

Value

A list of the following is returned:

- `map.comp` - MAP estimate of the dynamic component (or element of the state vector).

- `lb.comp` - Lower bound of the 95% CrIs of the dynamic component (or element of the state vector).
- `ub.comp` - Upper bound of the 95% CrIs of the dynamic component (or element of the state vector).
- `x` - Time/index values used for plotting.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:80]
trend.comp = polytrendMod(2, rep(0, 2), 10*diag(2))
seas.comp = seasMod(365, c(1, 2), C0 = 10*diag(4))
model = trend.comp + seas.comp
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98, 1), dim.df = c(2, 4),
                   gam.init = -3.5, sig.init = 15,
                   n.samp = 20, tol = 0.2, verbose = FALSE)
# plot first harmonic component
compPlot(M0, index = c(3, 4), col = "blue")
c.summary = compPlot(M0, index = c(3, 4), plot = FALSE)
options(old)
```

---

dexal — *Density Function for the Extended Asymmetric Laplace (exAL) Distribution*

Description

Computes the PDF of the Extended Asymmetric Laplace (exAL) distribution. Vectorized over `x`.

Usage

```
dexal(x, p0 = 0.5, mu = 0, sigma = 1, gamma = 0, log = FALSE)
```

Arguments

`x`	Numeric vector of quantiles.
`p0`	Probability level used in the quantile parametrization. Scalar in (0, 1). Default `0.5`.
`mu`	Location parameter (scalar). Default `0`.
`sigma`	Scale parameter (scalar, strictly positive). Default `1`.
`gamma`	Skewness parameter controlling asymmetry (scalar). Must be within valid bounds implied by `p0`. Default `0`.
`log`	Logical scalar; if `TRUE` return log-density. Default `FALSE`.

Value

Numeric vector of densities (same length as x).

Examples

```
dexal(0)
dexal(1, p0 = 0.75, mu = 0, sigma = 2, gamma = 0.25)
dexal(seq(-3, 3, by = 0.1), p0 = 0.3, mu = 0, sigma = 1, gamma = -0.45)
```

`ELIanoms` *Daily time-series of ELI anomalies.*

Description

ELI anomalies on the daily time-scale from January 1, 1979 to December 31, 2019 with all February 29ths omitted.

Usage

```
ELIanoms
```

Format

A time series of length 14965.

Source

https://portal.nersc.gov/archive/home/projects/cascade/www/ELI

References

Patricola, C.M., O'Brien, J.P., Risser, M.D. et al. *Maximizing ENSO as a source of western US hydroclimate predictability.* Clim Dyn 54, 351–372 (2020). doi:10.1007/s00382019050048

`exalStaticDiagnostics` *exAL Diagnostics*

Description

Static diagnostics companion for `exalStaticLDVB()` and `exalStaticMCMC()`. The function summarizes fitted quantiles on a shared design matrix, reports mean check loss against observed responses when available, and can optionally compare the fitted quantile curve against a known reference quantile function. The returned diagnostic object also stores posterior summaries for the static regression coefficients, which can be plotted with `plot(..., type = "coefficients")`.

Usage

```
exalStaticDiagnostics(
  m1,
  m2 = NULL,
  X = NULL,
  y = NULL,
  ref = NULL,
  plot = TRUE,
  cols = c("red", "blue"),
  cr.percent = 0.95
)
```

Arguments

`m1`	An object of class `"exalStaticLDVB"` or `"exalStaticMCMC"`.
`m2`	Optional second fitted static model to compare against `m1`.
`X`	Optional design matrix. If omitted, the function uses `m1$X` when available.
`y`	Optional response vector. If omitted, the function uses `m1$y` when available.
`ref`	Optional reference quantile vector on the same rows as `X`.
`plot`	Logical; if `TRUE`, produce a compact static-diagnostics plot.
`cols`	Character vector of length 1 or 2 giving colors for plotted diagnostics.
`cr.percent`	Credible-interval mass used when summarizing fitted quantiles.

Details

Unlike `exdqlmDiagnostics`, which is built around one-step-ahead dynamic forecast diagnostics, `exalStaticDiagnostics()` is designed for the static regression setting. It reports fitted quantile summaries on a common design matrix, optional mean check loss against observed responses, optional reference-curve errors, coefficient posterior summaries, and compact comparison plots. The `ref` argument is a reference conditional quantile evaluated on the rows of `X`; it is distinct from the optional `beta.ref` argument of `plot.exalStaticDiagnostic`, which is used only to overlay known coefficient values in simulation examples.

Value

An object of class "exalStaticDiagnostic" containing fitted-quantile summaries, residual summaries (when y is provided), optional reference-curve error metrics, coefficient posterior summaries, and run-time metadata for m1 and m2 (if supplied).

Examples

```
set.seed(1)
x <- seq(-2, 2, length.out = 60)
X <- cbind(1, x)
y <- 0.5 * x + (1.2 + 0.35 * x) * stats::rnorm(length(x))
q_true <- 0.5 * x + (1.2 + 0.35 * x) * stats::qnorm(0.25)

fit_ldvb <- exalStaticLDVB(
  y = y, X = X, p0 = 0.25,
  max_iter = 60, tol = 1e-3,
  verbose = FALSE
)
fit_mcmc <- exalStaticMCMC(
  y = y, X = X, p0 = 0.25,
  n.burn = 60, n.mcmc = 60,
  mh.proposal = "slice",
  verbose = FALSE
)
out <- exalStaticDiagnostics(fit_ldvb, fit_mcmc, ref = q_true, plot = FALSE)
print(out)
plot(out, type = "coefficients")
```

---

exalStaticLDVB *Static exAL Regression - LDVB Approximation*

Description

The function applies a mean-field variational approximation to static Extended Asymmetric Laplace (exAL) regression. Closed-form block updates are combined with a Laplace-Delta approximation for the joint (σ, γ) block, yielding the package's static LDVB routine.

Usage

```
exalStaticLDVB(
  y,
  X,
  p0,
  max_iter = 1000,
  tol = 1e-04,
  b0 = NULL,
  V0 = NULL,
```

```
  beta_prior = c("ridge", "rhs", "rhs_ns"),
  beta_prior_controls = NULL,
  a_sigma = 1,
  b_sigma = 1,
  gamma_bounds = c(L.fn(p0), U.fn(p0)),
  log_prior_gamma = NULL,
  init = NULL,
  dqlm.ind = FALSE,
  al.ind = NULL,
  n.samp = 200,
  n_samp_xi = 200,
  ld_controls = NULL,
  vb_control = NULL,
  verbose = TRUE
)
```

Arguments

- `y` Numeric vector (length n).
- `X` Numeric matrix (n x p).
- `p0` Target quantile in (0,1).
- `max_iter` Integer; maximum LDVB iterations (default 1000).
- `tol` Numeric; convergence tolerance based on relative ELBO changes (default 1e-4).
- `b0`, `V0` Prior mean and covariance for $\beta \sim \mathcal{N}(b_0, V_0)$.
- `beta_prior` Coefficient prior type: `"ridge"` (default), `"rhs"` (regularized horseshoe), or `"rhs_ns"` (Nishimura-Suchard regularized horseshoe with a closed-form inverse-gamma hierarchy for static inference).
- `beta_prior_controls` Optional list of prior-specific controls. For RHS-family priors (that is, when `beta_prior` is `"rhs"` or `"rhs_ns"`), supported keys include: `tau0`, `nu`, `s` or `s2`, `shrink_intercept`, `intercept_prec`, `n_inner`, `eta_bounds`, `freeze_tau_iters`, `freeze_tau_warmup_iters`, `update_every`, `update_every_warmup`, `update_every_warmup_iters`, `force_tau_after_warmup`, `collapse_tau_ratio_tol`, `collapse_beta_max_abs_tol`, `collapse_invV_med_tol`, `collapse_beta_l2_tol`, `collapse_small_beta_frac_tol`, `small_beta_abs_tol`, `warn_on_collapse`, `var_floor`, `h_curv`, `verbose`, `init_lambda`, `init_log_lambda`, `init_tau`, `init_log_tau`, `init_c2`, and `init_log_c2`. For `beta_prior = "rhs_ns"`, optional slab controls `a_zeta`, `b_zeta`, and `zeta2_fixed` are also supported. In this mode, the local-global-slab block is represented by $(\lambda_j^2, \nu_j, \tau^2, \xi, \zeta^2)$, with closed-form inverse-gamma updates in VB and MCMC. When `beta_prior` is `"rhs"` or `"rhs_ns"`, `b0` and `V0` are retained only for backward-compatible ridge behavior and are ignored for the shrunk coefficients. If both `init_log_tau` and `init_tau` are omitted (or `NULL`), the RHS global scale initializes at `tau = 1` (`init_log_tau = 0`) instead of `tau0`. By default (`shrink_intercept = FALSE`), the intercept is excluded from horseshoe shrinkage and uses `intercept_prec`.
- `a_sigma`, `b_sigma` Prior for $\sigma \sim IG(a_\sigma, b_\sigma)$ with density $p(\sigma) \propto \sigma^{-(a_\sigma+1)} e^{-b_\sigma/\sigma}$.

`gamma_bounds` Two-vector (L, U) support for `gamma`. Defaults to `c(L.fn(p0), U.fn(p0))`.

`log_prior_gamma`
Function `g` -> `log pi(gamma=g)`. Default is a truncated Student-t prior centered at 0 on the admissible `gamma` support.

`init` Optional list with starting values: `beta`, `sigma`, `gamma`; if missing, reasonable defaults are used.

`dqlm.ind` Logical; if `TRUE`, fit the reduced AL model (`gamma = 0`). In that special case the nonconjugate block drops out and the remaining variational updates are available in closed form. This argument is retained for consistency with the dynamic exDQLM API; in static examples, `al.ind = TRUE` is the clearer spelling for this AL special case.

`al.ind` Optional static-model alias for `dqlm.ind`. Prefer this argument when requesting the static AL special case. When supplied, this flag maps directly to `dqlm.ind`. If both are given, they must agree.

`n.samp` Number of samples to draw from the approximated posterior distribution after convergence. Default is `n.samp = 200`.

`n_samp_xi` Integer; retained for backward compatibility in the Laplace-Delta block. Under the current delta-only implementation this value is ignored.

`ld_controls` Optional list of controls for the Laplace-Delta block. Supported keys include `optimizer_method` (`"lbfgsb"` or `"bfgs"`), `direct_commit`, `damping`, `xi_damping`, `optimizer_maxit`, `eig_floor`, `eig_cap`, `step_cap_eta`, `step_cap_ell`, `eta_lo`, `eta_hi`, `sigma_bounds`, `sigma_init_mode`, `gamma_init_mode`, `sigma_floor_abs`, `sigma_min_mult`, `sigma_max_mult`, `sigma_bound_ratio_min`, `gamma_init_pad_frac`, `logit_eps`, `init_cov_diag`, `reuse_seed`, `mode_grad_tol`, `mode_min_eig`, `auto_stabilize`, `cycle_window`, `cycle_lag1_max`, `cycle_lag2_min`, `cycle_gamma_min_amp`, `cycle_sigma_min_amp`, `cycle_s_min_amp`, `cycle_tau2_min_amp`, `stabilize_damping`, `stabilize_xi_damping`, `stabilize_step_cap_eta`, `stabilize_step_cap_ell`, and `store_trace`.

`vb_control` Optional normalized VB control list, usually from `exal_make_vb_control()`. When supplied, the core VB arguments and warmup blocks are read from `vb_control` first and then merged with the explicit function arguments. When omitted, the package applies its conservative default warmup profile for exAL `(sigma, gamma)` updates while retaining the built-in RHS-family `tau` warmup defaults.

`verbose` Logical; print progress.

Details

Mean-field factorization:

$$q(\beta)\,\prod_{i=1}^{n} q(v_i)\,q(s_i)\,q(\sigma,\gamma).$$

This factorization induces a blockwise coordinate-ascent variational inference scheme. The (σ,γ) block is the only nonconjugate component, so LDVB approximates it in transformed coordinates $\eta = \mathrm{logit}((\gamma - L)/(U - L))$ and $\ell = \log \sigma$. The `xi` expectations needed by the remaining block updates are then computed with a second-order Delta approximation. The `xi` expectations used in the updates can be computed either from a deterministic second-order Delta approximation in

(η, ℓ) or from a Gaussian Monte Carlo sample. The Laplace-Delta controls also allow bounded optimization in the transformed (η, ℓ) block to better mimic the stabilized RHS-family readout implementation. For RHS-family priors, the prior block uses tau warmup/freeze scheduling to avoid the early-collapse regime where global shrinkage drives all slope coefficients toward zero before the likelihood-side variational factors stabilize.

Value

An object of class "exalStaticLDVB" containing:

- qbeta: list with m, V.
- samp.beta: posterior sample from $q(\beta)$ with n.samp rows.
- qv: list with chi (length n), psi (scalar), E_v and E_inv_v (moments).
- qs: list with mu (length n), tau2 (length n), E_s, E_s2.
- qsiggam: list with eta_hat, ell_hat, Sigma (2x2), approximate means gamma_mean, sigma_mean, and the xi expectations.
- samp.sigma, samp.gamma: posterior samples from the variational approximation for the scale and skewness parameters. In the AL special case (dqlm.ind = TRUE), samp.gamma is a vector of zeros.
- converged, iter, run.time, and misc (including p0, bounds L,U, dimensions, and ELBO trace).
- beta_prior: prior metadata and, for RHS-family priors, posterior summaries of the shrinkage hyperparameters, including warmup/freeze-aware tau summaries and collapse diagnostics (collapse_flag, tau_near_zero, beta_collapse, and warning when triggered). For "rhs_ns", the state also tracks lambda2, nu, tau2, xi, and zeta2 with the corresponding inverse moments.
- diagnostics: ELBO and joint-convergence diagnostics including a standardized VB iteration trace at diagnostics$vb_trace (iteration-wise ELBO / sigma / gamma / convergence deltas), state/sigma/gamma/ELBO deltas, stopping reason, and Laplace-Delta block trace diagnostics, including replicated-xi controls, automatic stabilization / cycle-detection fields, and final local-mode quality checks. For RHS fits this also includes diagnostics$rhs with the resolved preflight configuration and collapse diagnostics.

Examples

```
set.seed(123)
n <- 60
X <- cbind(1, seq(-1, 1, length.out = n))
y <- as.numeric(X %*% c(0.2, -0.1) + rnorm(n, sd = 0.15))
fit <- exalStaticLDVB(y = y, X = X, p0 = 0.5, max_iter = 60, tol = 1e-3, verbose = FALSE)
fit$converged
head(fit$diagnostics$vb_trace)

fit_rhs <- exalStaticLDVB(
  y = y, X = X, p0 = 0.5,
  beta_prior = "rhs",
  beta_prior_controls = list(tau0 = 0.5, nu = 3, s2 = 1, shrink_intercept = FALSE),
  max_iter = 50, tol = 5e-3, verbose = FALSE
```

```
)
fit_rhs$beta_prior$type

fit_rhs_ns <- exalStaticLDVB(
  y = y, X = X, p0 = 0.5,
  beta_prior = "rhs_ns",
  beta_prior_controls = list(tau0 = 0.5, a_zeta = 1.5, b_zeta = 1, zeta2_fixed = 1),
  max_iter = 50, tol = 5e-3, verbose = FALSE
)
fit_rhs_ns$beta_prior$type

fit_al <- exalStaticLDVB(
  y = y, X = X, p0 = 0.5,
  al.ind = TRUE,
  max_iter = 50, tol = 5e-3, verbose = FALSE
)
fit_al$dqlm.ind
```

`exalStaticMCMC` *exAL (static) - MCMC algorithm*

Description

The function applies a Gibbs sampler for static Extended Asymmetric Laplace regression (exAL). We update β, v, s from their full conditionals, then update (σ, γ) either jointly on transformed coordinates $(\ell, \eta) = (\log \sigma, \text{logit}((\gamma - L)/(U - L)))$ (for `"rw"` and `"laplace_rw"`) or with univariate gamma kernels (`"slice"`, `"slice_eta"`, `"laplace_local"`). Optional multi-refresh and global-jump controls are available to improve exAL mixing in hard cases.

Usage

```
exalStaticMCMC(
  y,
  X,
  p0,
  b0 = NULL,
  V0 = NULL,
  beta_prior = c("ridge", "rhs", "rhs_ns"),
  beta_prior_controls = NULL,
  a_sigma = 1,
  b_sigma = 1,
  gamma_bounds = c(L.fn(p0), U.fn(p0)),
  log_prior_gamma = NULL,
  init = NULL,
  dqlm.ind = FALSE,
  al.ind = NULL,
  n.burn = 2000,
```

```
  n.mcmc = 1500,
  thin = 1,
  init.from.vb = FALSE,
  vb_init_controls = NULL,
  vb_init_fit = NULL,
  mcmc_control = NULL,
  sigmagam_controls = NULL,
  mh.proposal = c("slice", "laplace_rw", "rw", "slice_eta", "laplace_local"),
  mh.adapt = TRUE,
  mh.adapt.interval = 50L,
  mh.target.accept = c(0.2, 0.45),
  mh.scale.bounds = c(0.1, 10),
  mh.max_scale.step = 0.35,
  mh.min_burn_adapt = 50L,
  slice.width = 0.1,
  slice.max.steps = Inf,
  gamma.substeps = 1L,
  p.global.eta.jump = 0,
  global.eta.jump.scale = 1,
  trace.diagnostics = TRUE,
  trace.every = 1L,
  verbose = TRUE,
  progress_callback = NULL
)
```

Arguments

y Numeric vector of length n.

X Numeric matrix $n \times p$ (design).

p0 Quantile level in $(0, 1)$.

b0, V0 Prior mean and covariance for β (Normal). Defaults: $b_0 = \mathbf{0}_p$, $V_0 = 10^6 I_p$.

beta_prior Coefficient prior type: "ridge" (default), "rhs" (regularized horseshoe), or "rhs_ns" (Nishimura-Suchard regularized horseshoe with a closed-form inverse-gamma hierarchy for static inference).

beta_prior_controls
Optional list of prior-specific controls. For RHS-family priors (that is, when beta_prior is "rhs" or "rhs_ns"), supported keys include: tau0, nu, s or s2, shrink_intercept, intercept_prec, n_inner, eta_bounds, var_floor, h_curv, verbose, init_lambda, init_log_lambda, init_tau, init_log_tau, init_c2, init_log_c2, collapse_tau_ratio_tol, collapse_beta_max_abs_tol, collapse_invV_med_tol, collapse_beta_l2_tol, collapse_small_beta_frac_tol, small_beta_abs_tol, slice_width, and slice_max_steps. For beta_prior = "rhs_ns", optional slab controls a_zeta, b_zeta, and zeta2_fixed are also supported. In this mode, the local-global-slab block is represented by $(\lambda_j^2, \nu_j, \tau^2, \xi, \zeta^2)$ and updated with closed-form Gibbs steps. When beta_prior is "rhs" or "rhs_ns", b0 and V0 are ignored for the shrunk coefficients and retained only for backward-compatible ridge behavior. If both init_log_tau and init_tau

are omitted (or NULL), the RHS global scale initializes at tau = 1 (init_log_tau = 0) instead of tau0. By default (shrink_intercept = FALSE), the intercept is excluded from horseshoe shrinkage and uses intercept_prec.

a_sigma, b_sigma
: Hyperparameters for an inverse-gamma prior on sigma, with density proportional to sigma^{-(a_sigma+1)} exp(-b_sigma/sigma).

gamma_bounds
: Numeric length-2 vector (L, U) for gamma. Defaults to c(L.fn(p0), U.fn(p0)).

log_prior_gamma
: Function function(g) log pi(g) for gamma on (L, U). Default is a truncated Student-t prior centered at 0 on the admissible support.

init
: Optional list with starting values: beta, sigma, gamma, v (length n), s (length n), and for RHS-family priors optionally lambda, tau, and c2. For beta_prior = "rhs_ns", the squared-scale parameterization lambda2, tau2, zeta2, and optional auxiliaries nu, xi are also accepted. Missing pieces are filled sensibly.

dqlm.ind
: Logical; if TRUE, fit the reduced AL model (gamma = 0), corresponding to Bayesian linear quantile regression under the AL working likelihood. This removes the gamma- and s-blocks and leaves conjugate Gibbs updates for beta, sigma, and v. This argument is retained for consistency with the dynamic exDQLM API; in static examples, al.ind = TRUE is the clearer spelling for this AL special case.

al.ind
: Optional static-model alias for dqlm.ind. Prefer this argument when requesting the static AL special case. When supplied, this flag maps directly to dqlm.ind. If both are given, they must agree.

n.burn
: Number of burn-in iterations. Default 2000.

n.mcmc
: Number of kept MCMC iterations (after burn). Default 1500.

thin
: Integer; save every thin-th iteration after burn. We internally run n.burn + n.mcmc * thin iterations to return exactly n.mcmc saved draws.

init.from.vb
: Logical; if TRUE, run static VB first and use its posterior moments as MCMC initialization.

vb_init_controls
: Optional list controlling VB warm start. Supported keys: max_iter, tol, n_samp_xi, verbose, and ld_controls (passed through to exalStaticLDVB()).

vb_init_fit
: Optional precomputed static VB fit object used as the warm-start reference when init.from.vb = TRUE.

mcmc_control
: Optional normalized MCMC control list, usually from exal_make_mcmc_control(). When supplied, the core MCMC arguments and warmup blocks are read from mcmc_control first and then merged with the explicit function arguments. When omitted, the package applies its conservative default exAL (sigma, gamma) warmup profile and keeps the RHS-family tau warmup defaults active through the shared prior layer.

sigmagam_controls
: Optional list controlling warmup/freeze behavior for the nonconjugate (sigma, gamma) block. Prefer exal_make_mcmc_sigmagam_control() or the sigmagam block of exal_make_mcmc_control() for new code; this argument remains available as a direct advanced override.

mh.proposal
: Character string controlling the exAL nonconjugate update kernel. "slice" (default) uses an exact bounded univariate slice sampler on gamma (with sigma updated from its conditional), and "slice_eta" does the same on transformed eta. "laplace_rw" uses a Laplace-informed adaptive random-walk MH update on the transformed joint block $(\eta, \ell) = (\text{logit}((\gamma - L)/(U - L)), \log\sigma)$. "rw" uses the same exact joint update with identity base covariance. "laplace_local" reproduces the prior approximate local-Gaussian gamma draw retained for compatibility and not recommended for routine use. Only "laplace_local" is approximate.

mh.adapt
: Logical; adapt the random-walk proposal scale during burn-in for "rw" and "laplace_rw". Ignored for "laplace_local", "slice", and "slice_eta".

mh.adapt.interval
: Integer adaptation window for RW-based kernels.

mh.target.accept
: Numeric length-2 target acceptance band.

mh.scale.bounds
: Numeric length-2 lower/upper bounds for RW proposal scale multiplier.

mh.max_scale.step
: Numeric multiplicative adaptation cap in (0,1).

mh.min_burn_adapt
: Integer minimum burn-in before adaptation starts.

slice.width
: Positive numeric width for bounded slice updates when mh.proposal = "slice" or "slice_eta".

slice.max.steps
: Positive integer or Inf; maximum stepping-out expansions for the slice sampler.

gamma.substeps
: Positive integer; number of consecutive gamma-kernel refreshes per outer MCMC iteration. Default 1.

p.global.eta.jump
: Numeric in [0,1]; per-substep probability of proposing a global independence-MH move on eta (the logit transform of gamma) using a Laplace proposal with MH correction. Default 0.

global.eta.jump.scale
: Positive numeric scale multiplier applied to the Laplace proposal SD used in global eta jumps.

trace.diagnostics
: Logical; if TRUE, retain per-iteration gamma/s diagnostics under mh.diagnostics$trace. Set FALSE for lighter-weight runs.

trace.every
: Positive integer; when trace.diagnostics=TRUE, record one diagnostics row every trace.every iterations.

verbose
: Print progress every progress_every iterations.

progress_callback
: Optional callback invoked with a named list at MCMC start, at each progress checkpoint, and on completion. Intended for workflow-level per-case progress logging.

Value

An object of class "exalStaticMCMC" containing:

- `run.time` - total wall time in seconds.
- `X`, `p0`, bounds - design, quantile, and (L, U).
- `samp.beta` - posterior sample of `beta` as `coda::mcmc` (n.mcmc x p).
- `samp.sigma` - posterior sample of `sigma` as `coda::mcmc`.
- `samp.gamma` - posterior sample of `gamma` as `coda::mcmc`.
- `samp.v` - latent `v` draws as `coda::mcmc` (`n.mcmc` x `n`).
- `samp.s` - latent `s` draws as `coda::mcmc` (`n.mcmc` x `n`).
- `samp.lambda`, `samp.tau`, `samp.c2` - RHS latent draws when an RHS-family prior is used; otherwise `NULL`.
- `beta_prior` - prior metadata and, for RHS-family priors, posterior summaries of the shrinkage block. For `"rhs_ns"`, the state tracks `lambda2`, `nu`, `tau2`, `xi`, and `zeta2`.
- `mh.diagnostics` - proposal kernel diagnostics for the exAL gamma update, including whether the saved kernel is exact/signoff-ready.
- `rhs.diagnostics` - RHS latent summaries and optional trace metadata when an RHS-family prior is used, including the resolved preflight configuration used at fit start.
- `last` - last state of the chain (useful for restarts).

Examples

```
set.seed(123)
n <- 60; p <- 3
X <- cbind(1, rnorm(n), rnorm(n))
beta0 <- c(0.5, -1, 0.8); sigma0 <- 1.2
y <- as.numeric(X %*% beta0 + rnorm(n, 0, sigma0))
fit <- exalStaticMCMC(
  y, X, p0 = 0.5, n.burn = 60, n.mcmc = 60, thin = 1, verbose = FALSE
)
summary(fit$samp.beta)

fit_rhs <- exalStaticMCMC(
  y, X, p0 = 0.5,
  beta_prior = "rhs",
  beta_prior_controls = list(tau0 = 0.5, nu = 3, s2 = 1, shrink_intercept = FALSE),
  n.burn = 50, n.mcmc = 50, thin = 1, mh.proposal = "slice", verbose = FALSE
)
fit_rhs$beta_prior$type

fit_rhs_ns <- exalStaticMCMC(
  y, X, p0 = 0.5,
  beta_prior = "rhs_ns",
  beta_prior_controls = list(tau0 = 0.5, a_zeta = 1.5, b_zeta = 1, zeta2_fixed = 1),
  n.burn = 40, n.mcmc = 40, thin = 1, mh.proposal = "slice", verbose = FALSE
)
fit_rhs_ns$beta_prior$type
```

```
fit_al <- exalStaticMCMC(
  y, X, p0 = 0.5,
  al.ind = TRUE,
  n.burn = 50, n.mcmc = 50, thin = 1, verbose = FALSE
)
fit_al$dqlm.ind
```

---

exal_make_mcmc_control *Build advanced MCMC control*

Description

Returns a readable `mcmc_control` list for `exalStaticMCMC()` and `exdqlmMCMC()`. This keeps the warmup surface explicit instead of relying on ad hoc nested lists.

Usage

```
exal_make_mcmc_control(
  n_burn = 2000L,
  n_mcmc = 1500L,
  thin = 1L,
  verbose = FALSE,
  progress_every = NULL,
  init_from_vb = TRUE,
  vb_warm_start_control = NULL,
  sigmagam = NULL,
  theta = NULL,
  latent_state = NULL,
  dqlm_sigma = NULL,
  control = NULL
)
```

Arguments

`n_burn, n_mcmc, thin, verbose`	Core MCMC controls.
`progress_every`	Optional progress cadence for callers that support it.
`init_from_vb`	Logical; initialize from a VB warm start.
`vb_warm_start_control`	Optional VB warm-start control list, often from `exal_make_vb_control()`.
`sigmagam`	Optional list, usually from `exal_make_mcmc_sigmagam_control()`.
`theta`	Optional list, usually from `exal_make_mcmc_theta_control()`.
`latent_state`	Optional list, usually from `exal_make_mcmc_latent_state_control()`.
`dqlm_sigma`	Optional list, usually from `exal_make_mcmc_dqlm_sigma_control()`.
`control`	Optional existing control list to update.

Value

A normalized list suitable for `mcmc_control`.

`exal_make_mcmc_dqlm_sigma_control`
Build DQLM sigma-only MCMC warmup control

Description

Returns a normalized `mcmc_control$dqlm_sigma` block for `exdqlmMCMC()` in the reduced AL / DQLM branch.

Usage

```
exal_make_mcmc_dqlm_sigma_control(
  freeze_burnin_iters = 0L,
  freeze_only_during_burn = TRUE,
  force_after_warmup = TRUE,
  trace = TRUE
)
```

Arguments

`freeze_burnin_iters`
: Non-negative integer; number of burn-in iterations to hold the sigma-only block fixed.

`freeze_only_during_burn`
: Logical; if `TRUE`, hard freeze only applies during burn-in.

`force_after_warmup`
: Logical; force one immediate post-warmup update.

`trace`
: Logical; record diagnostics traces.

Value

A normalized list suitable for `mcmc_control$dqlm_sigma`.

exal_make_mcmc_latent_state_control

Build dynamic MCMC latent-state warmup control

Description

Returns a normalized `mcmc_control$latent_state` block for `exdqlmMCMC()`.

Usage

```
exal_make_mcmc_latent_state_control(
  mode = c("u_only", "u_st_pair"),
  freeze_burnin_iters = 0L,
  freeze_only_during_burn = TRUE,
  force_after_warmup = TRUE,
  min_postwarmup_updates = 0L,
  trace = TRUE
)
```

Arguments

`mode`	One of "`u_only`" or "`u_st_pair`".
`freeze_burnin_iters`	Non-negative integer; number of burn-in iterations to hold the latent-state block fixed.
`freeze_only_during_burn`	Logical; if `TRUE`, hard freeze only applies during burn-in.
`force_after_warmup`	Logical; force one immediate post-warmup update.
`min_postwarmup_updates`	Non-negative integer; minimum number of post-warmup updates required before chain-health style gates can fire.
`trace`	Logical; record diagnostics traces.

Value

A normalized list suitable for `mcmc_control$latent_state`.

exal_make_mcmc_sigmagam_control

Build MCMC sigmagam warmup control

Description

Returns a normalized `mcmc_control$sigmagam` block for `exalStaticMCMC()` and `exdqlmMCMC()`.

Usage

```
exal_make_mcmc_sigmagam_control(
  freeze_burnin_iters = NULL,
  freeze_only_during_burn = NULL,
  force_after_warmup = NULL,
  delay_adapt_until_after_warmup = NULL,
  delay_laplace_refresh_until_after_warmup = NULL
)
```

Arguments

`freeze_burnin_iters`
: Non-negative integer; number of burn-in iterations to hold the `(sigma, gamma)` block fixed.

`freeze_only_during_burn`
: Logical; if `TRUE`, hard freeze only applies during burn-in.

`force_after_warmup`
: Logical; force one post-warmup update.

`delay_adapt_until_after_warmup`
: Logical; keep proposal adaptation off until warmup ends.

`delay_laplace_refresh_until_after_warmup`
: Logical; keep Laplace refresh off until warmup ends.

Value

A normalized list suitable for `mcmc_control$sigmagam`.

When called with no arguments, this returns the package's conservative default exAL `(sigma, gamma)` MCMC warmup profile.

exal_make_mcmc_theta_control

Build MCMC theta warmup control

Description

Returns a normalized `mcmc_control$theta` block for `exdqlmMCMC()`.

Usage

```
exal_make_mcmc_theta_control(
  enabled = FALSE,
  freeze_burnin_iters = 0L,
  freeze_only_during_burn = TRUE,
  sparse_update_every = 1L,
  sparse_update_until_iter = 0L,
  force_first_postwarmup_update = TRUE,
  trace = TRUE
)
```

Arguments

- `enabled` Logical; explicit on/off switch.
- `freeze_burnin_iters` Non-negative integer; number of burn-in iterations to hold the theta block fixed.
- `freeze_only_during_burn` Logical; if `TRUE`, hard freeze only applies during burn-in.
- `sparse_update_every` Positive integer; sparse-update period during the warmup window.
- `sparse_update_until_iter` Non-negative integer; last iteration where the sparse schedule is active.
- `force_first_postwarmup_update` Logical; force one update immediately after the hard freeze / sparse schedule ends.
- `trace` Logical; record diagnostics traces.

Value

A normalized list suitable for `mcmc_control$theta`.

exal_make_vb_control *Build advanced VB control*

Description

Returns a readable vb_control list for exalStaticLDVB() and exdqlmLDVB(). This keeps the warmup surface explicit instead of relying on ad hoc nested lists.

Usage

```
exal_make_vb_control(
  max_iter = 150L,
  tol = 1e-04,
  n_samp_xi = 200L,
  verbose = FALSE,
  sigmagam = NULL,
  sts = NULL,
  control = NULL
)
```

Arguments

max_iter, tol, n_samp_xi, verbose	Core VB controls.
sigmagam	Optional list, usually from exal_make_vb_sigmagam_control().
sts	Optional list, usually from exal_make_vb_sts_control(), for the dynamic latent s_t block in exdqlmLDVB().
control	Optional existing control list to update.

Value

A normalized list suitable for vb_control.

exal_make_vb_sigmagam_control *Build VB sigmagam warmup control*

Description

Returns a normalized sigmagam block for vb_control lists used by exalStaticLDVB(), exdqlmLDVB(), and VB warm-start paths in exalStaticMCMC() and exdqlmMCMC().

Usage

```
exal_make_vb_sigmagam_control(
  freeze_warmup_iters = NULL,
  force_after_warmup = NULL,
  postwarmup_damping = NULL,
  postwarmup_damping_iters = NULL,
  min_postwarmup_updates = NULL
)
```

Arguments

`freeze_warmup_iters`
: Non-negative integer; number of early VB iterations during which the `(sigma, gamma)` block is held fixed.

`force_after_warmup`
: Logical; force one immediate post-warmup update.

`postwarmup_damping`
: Numeric in `(0, 1]`; damping applied after warmup.

`postwarmup_damping_iters`
: Non-negative integer; number of damped post-warmup iterations.

`min_postwarmup_updates`
: Non-negative integer; minimum number of post-warmup updates required before convergence-style gates can fire.

Value

A normalized list suitable for `vb_control$sigmagam`.

When called with no arguments, this returns the package's conservative default exAL `(sigma, gamma)` warmup profile.

---

exal_make_vb_sts_control

Build dynamic VB latent-state warmup control

Description

Returns a normalized `sts` block for `vb_control` lists used by `exdqlmLDVB()`.

Usage

```
exal_make_vb_sts_control(
  freeze_warmup_iters = 0L,
  force_after_warmup = TRUE,
  min_postwarmup_updates = 0L
)
```

Arguments

`freeze_warmup_iters`
: Non-negative integer; number of early VB iterations during which the latent `s_t` block is held fixed.

`force_after_warmup`
: Logical; force one immediate post-warmup update.

`min_postwarmup_updates`
: Non-negative integer; minimum number of post-warmup updates required before convergence-style gates can fire.

Value

A normalized list suitable for `vb_control$sts`.

`exdqlmDiagnostics` *exDQLM Diagnostics*

Description

The function computes the following for the model(s) provided: the posterior predictive loss criterion based off the check loss, the CRPS approximated as a finite integrated quantile score over posterior predictive empirical quantiles, the one-step-ahead distribution sequence, and deterministic semiclosed KL normality diagnostics for the MAP standardized forecast errors. The function also plots the following: the qq-plot and ACF plot corresponding to the one-step-ahead distribution sequence, and a time series plot of the MAP standard forecast errors.

Usage

```
exdqlmDiagnostics(
  m1,
  m2 = NULL,
  plot = TRUE,
  cols = c("red", "blue"),
  ref = NULL,
  crps_probs = seq(0.01, 0.99, by = 0.01),
  crps_weights = NULL,
  kl_k = NULL
)
```

Arguments

`m1`
: An object of class "`exdqlmLDVB`", "`exdqlmMCMC`", or legacy "`exdqlmISVB`".

`m2`
: An optional additional object of class "`exdqlmLDVB`", "`exdqlmMCMC`", or legacy "`exdqlmISVB`" to compare with `m1`.

plot — Logical value indicating whether the following will be plotted for `m1` and `m2` (if provided): a qq-plot and ACF plot of the MAP one-step-ahead distribution sequence, and a time series plot of the standardized forecast errors. Default is `TRUE`.

cols — Character vector of length 1 or 2 giving color(s) used to plot diagnostics. Default `c("red","blue")`.

ref — Optional finite reference sample of size `length(m1$y)` from a standard normal distribution. Used for the reversed KL diagnostic. When `NULL`, a deterministic standard-normal quantile grid is used.

crps_probs — Numeric vector of quantile levels used to approximate CRPS through the integrated quantile-score identity. Values must be strictly between 0 and 1. Default is `seq(0.01, 0.99, by = 0.01)`.

crps_weights — Optional non-negative numeric weights for `crps_probs`. When `NULL`, equal weights are used. When provided, weights are normalized to sum to 1.

kl_k — Optional positive integer vector of nearest-neighbor values used for the KL entropy and cross-entropy estimates. When `NULL`, the default grid `c(3, 5, 10, 20, 30)` is filtered to values supported by the finite standardized-error sample size, falling back to `1` for very small samples.

Details

The primary KL summary is computed from the MAP standardized one-step-ahead forecast errors `map.standard.forecast.errors`. The reported `KL` value is the user-facing calibration diagnostic and estimates $KL(P_e||N(0,1))$, where P_e is the continuous diagnostic-error law represented by the standardized errors. It uses the semiclosed identity $KL(P_e||N(0,1)) = CE(P_e, N(0,1)) - H(P_e)$, with the normal cross-entropy term evaluated analytically and the entropy estimated by a one-dimensional k-nearest-neighbor estimator. The reported `KL.flip` estimates the reversed diagnostic $KL(N(0,1)||P_e)$ using kNN cross-entropy. The reversed direction is more sensitive and should be read as a secondary sensitivity diagnostic, not as a replacement for `KL`. Advanced by-`k` sensitivity tables and Gaussian plug-in checks are stored under `kl.details` so the top-level diagnostic object exposes a single primary KL value. Negative finite-sample estimates are not clamped; they indicate estimator bias or instability for the current sample.

Value

An object of class "`exdqlmDiagnostic`" containing the following:

- `m1.uts` - The one-step-ahead distribution sequence of `m1`.
- `m1.KL` - The forward KL normality diagnostic `KL(P_error || N(0,1))` for the MAP standardized forecast errors.
- `m1.KL.flip` - The reversed ("flipped") KL diagnostic `KL(N(0,1) || P_error)` for the MAP standardized forecast errors; this is a secondary sensitivity diagnostic.
- `m1.CRPS` - The mean CRPS approximated by a finite integrated quantile score over posterior predictive empirical quantiles.
- `m1.pplc` - The posterior predictive loss criterion of `m1` based off the check loss function.
- `m1.qq` - The ordered pairs of the qq-plot comparing `m1.uts` with a standard normal distribution.

- `m1.acf` - The autocorrelations of `m1.uts` by lag.
- `m1.rt` - Run-time of the original model `m1` in seconds.
- `m1.msfe` - MAP standardized one-step-ahead forecast errors from the original model `m1`.
- `y` - The original time-series used to fit `m1`.
- `crps.method` - The CRPS approximation method.
- `crps.probs` - The quantile levels used for the CRPS approximation.
- `crps.weights` - The normalized weights used for the CRPS approximation.
- `kl.method`, `kl.k`, `kl.aggregate`, and `kl.reference` - KL estimator metadata.
- `kl.n_finite`, `kl.n_ref`, and `kl.zero_distance_count` - KL diagnostic sample-size and distance-floor metadata.
- `kl.details` - Advanced KL estimator details by model. For each model this includes primary/flipped definitions, by-k sensitivity tables, a Gaussian plug-in check, and estimator metadata.

If `m2` is provided, analogous results for `m2` are also included in the list.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.95), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.samp = 20, tol = 0.2, verbose = FALSE)
M0.diags = exdqlmDiagnostics(M0, plot = FALSE)
options(old)
```

`exdqlmForecast` *k-step-ahead quantile forecasts*

Description

Computes filtered and k-step-ahead forecast quantiles from a fitted dynamic quantile model and optionally adds them to an existing plot.

Usage

```
exdqlmForecast(
  start.t,
  k,
  m1,
  fFF = NULL,
  fGG = NULL,
```

```
  plot = TRUE,
  add = FALSE,
  cols = c("purple", "magenta"),
  cr.percent = 0.95,
  return.draws = FALSE,
  n.samp = NULL,
  seed = NULL
)
```

Arguments

`start.t`	Integer index at which forecasts start (must be within the span of the fitted model in `m1`).
`k`	Integer number of steps ahead to forecast.
`m1`	A fitted exDQLM model object, returned by `exdqlmLDVB()`, `exdqlmMCMC()`, or legacy `exdqlmISVB()`.
`fFF`	Optional state vector(s) for the forecast steps. A numeric matrix with q rows and either 1 column (non–time-varying) or `k` columns (time-varying). Its dimension must match the fitted model in `m1`.
`fGG`	Optional evolution matrix/matrices for the forecast steps. Either a numeric $q \times q$ matrix (non–time-varying) or a $q \times q \times k$ array (time-varying). Its dimensions must match the fitted model in `m1`.
`plot`	Logical value indicating whether to plot filtered and forecast quantiles with equal–tailed credible intervals. Default is `TRUE`.
`add`	Logical value indicating whether to add the forecasted quantiles to the current plot. Default is `FALSE`.
`cols`	Character vector of length 2 giving the colors for filtered and forecasted quantiles respectively. Default `c("purple","magenta")`.
`cr.percent`	Numeric in `(0, 1)` indicating the probability mass for the credible intervals (e.g., `0.95`). Default `0.95`.
`return.draws`	Logical; if `TRUE`, the function also returns a matrix of posterior predictive forecast draws in `samp.fore`. Default is `FALSE`.
`n.samp`	Optional positive integer specifying how many forecast draws to return when `return.draws = TRUE`. If omitted, all available posterior (σ, γ) draws from `m1` are used.
`seed`	Optional integer random seed used only for forecast-draw generation when `return.draws = TRUE`. If provided, the previous R RNG state is restored on exit.

Value

An object of class "`exdqlmForecast`" containing the following:

- `start.t` Integer index at which forecasts start (within the span of the fitted model in `m1`).
- `k` Integer number of steps ahead forecasted.
- `m1` The fitted exDQLM model object used to initialize the forecast.

- `cr.percent` The probability mass for the credible intervals (e.g., `0.95`).
- `fa` Forecast state mean vectors ($q \times k$ matrix).
- `fR` Forecast state covariance matrices ($q \times q \times k$ array).
- `ff` Forecast quantile means (length-k numeric).
- `fQ` Forecast quantile variances (length-k numeric).
- `samp.fore` Optional posterior predictive forecast draws (k x `n.samp`) returned when `return.draws = TRUE`.

Examples

```
# Toy example
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 20L)
y = scIVTmag[1:100]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15, n.samp = 30,
                verbose = FALSE)
exdqlmForecast(start.t = 90, k = 10, m1 = M0)
M0.forecast = exdqlmForecast(start.t = 90, k = 10, m1 = M0,
                             return.draws = TRUE, n.samp = 50, seed = 123)
dim(M0.forecast$samp.fore)
options(old)
```

`exdqlmForecastDiagnostics` *Held-out forecast diagnostics for exDQLM forecasts*

Description

Computes held-out forecast scores from one or two `exdqlmForecast` objects returned by `exdqlmForecast()`. Unlike `exdqlmDiagnostics()`, which summarizes fitted one-step-ahead forecast errors and their KL normality diagnostics, this function evaluates posterior predictive forecast draws against observations reserved outside the fitted sample.

Usage

```
exdqlmForecastDiagnostics(
  m1,
  m2 = NULL,
  y,
  p0 = NULL,
  crps_probs = seq(0.01, 0.99, by = 0.01),
  crps_weights = NULL
)
```

Arguments

m1	An object of class "exdqlmForecast", returned by exdqlmForecast() with return.draws = TRUE.
m2	An optional second object of class "exdqlmForecast" to compare with m1.
y	Numeric vector or time series of held-out observations. Its length must equal the forecast horizon.
p0	Optional quantile level used for the check-loss calculation. When NULL, the value is taken from m1$m1$p0. If m2 is supplied, its fitted quantile level must agree with the resolved value.
crps_probs	Numeric vector of quantile levels used to approximate CRPS through the integrated quantile-score identity. Values must be strictly between 0 and 1. Default is seq(0.01, 0.99, by = 0.01).
crps_weights	Optional non-negative numeric weights for crps_probs. When NULL, equal weights are used. When provided, weights are normalized to sum to 1.

Details

The check loss is computed at the target quantile level p0 using the forecast quantile means ff stored in each forecast object. CRPS is computed from samp.fore using the same finite integrated quantile-score approximation used by exdqlmDiagnostics(). This function does not compute KL diagnostics because KL in **exdqlm** is defined for fitted one-step-ahead MAP standardized forecast errors, not for arbitrary held-out forecast draws.

Value

An object of class "exdqlmForecastDiagnostic" containing:

- y - Held-out observations used for scoring.
- p0 - Quantile level used for check loss.
- horizon - Forecast horizon.
- m1.check_loss - Mean target-quantile check loss for m1.
- m1.CRPS - Mean CRPS approximation for m1.
- m1.pointwise - Pointwise held-out scores for m1.
- crps.method, crps.probs, and crps.weights - CRPS approximation metadata.

If m2 is supplied, analogous m2.* fields are included.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:65]
y_train = y[1:60]
y_holdout = y[61:65]
model = polytrendMod(1, stats::quantile(y_train, 0.85), 10)
M0 = exdqlmLDVB(y_train, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
```

```
                   n.samp = 20, tol = 0.2, verbose = FALSE)
fFF = model$FF[, 1, drop = FALSE]
fGG = model$GG
M0.forecast = exdqlmForecast(start.t = 60, k = 5, m1 = M0,
                             fFF = fFF, fGG = fGG,
                             return.draws = TRUE, n.samp = 20, seed = 123,
                             plot = FALSE)
exdqlmForecastDiagnostics(M0.forecast, y = y_holdout)
options(old)
```

`exdqlmISVB` *exDQLM - legacy ISVB algorithm*

Description

The function applies an Importance Sampling Variational Bayes (ISVB) algorithm to estimate the posterior of an exDQLM. This legacy VB engine is retained for backward compatibility and historical comparisons; for standard exDQLM VB fits, `exdqlmLDVB()` is the main default technique.

Usage

```
exdqlmISVB(
  y,
  p0,
  model,
  df,
  dim.df,
  fix.gamma = FALSE,
  gam.init = NA,
  fix.sigma = FALSE,
  sig.init = NA,
  dqlm.ind = FALSE,
  exps0,
  tol = 0.1,
  n.IS = 500,
  n.samp = 200,
  PriorSigma = NULL,
  PriorGamma = NULL,
  verbose = TRUE,
  debug_shapes = FALSE,
  debug_every = 5
)
```

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.

model	List of the state-space model including GG, FF, prior parameters m0 and C0.
df	Discount factors for each block.
dim.df	Dimension of each block of discount factors.
fix.gamma	Logical value indicating whether to fix gamma at gam.init. Default is FALSE.
gam.init	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if fix.gamma=TRUE.
fix.sigma	Logical value indicating whether to fix sigma at sig.init. Default is FALSE.
sig.init	Initial value for sigma (scale parameter), or value at which sigma will be fixed if fix.sigma=TRUE.
dqlm.ind	Logical value indicating whether to fix gamma at 0, reducing the exDQLM to the special case of the DQLM. Default is FALSE.
exps0	Initial value for dynamic quantile. If exps0 is not specified, it is set to the DLM estimate of the p0 quantile.
tol	Tolerance for convergence of dynamic quantile estimates. Default is tol=0.1.
n.IS	Number of particles for the importance sampling of joint variational distribution of sigma and gamma. Default is n.IS=500.
n.samp	Number of samples to draw from the approximated posterior distribution. Default is n.samp=200.
PriorSigma	List of parameters for inverse gamma prior on sigma; shape a_sig and scale b_sig. Default is an inverse gamma with mean 1 (or sig.init if provided) and variance 10.
PriorGamma	List of parameters for truncated student-t prior on gamma; center m_gam, scale s_gam and degrees of freedom df_gam. Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
verbose	Logical value indicating whether progress should be displayed.
debug_shapes	Logical; if TRUE, print KF input/output shapes every debug_every iterations.
debug_every	Integer; frequency (in iterations) for shape prints when debug_shapes=TRUE.

Details

Advanced options (set via options()):

- exdqlm.use_cpp_kf: use the C++ Kalman filter bridge (default TRUE).
- exdqlm.compute_elbo: compute ELBO every iteration (default TRUE).
- exdqlm.tol_elbo: ELBO convergence tolerance (default 1e-6).
- exdqlm.tol_sigma: sigma-delta convergence tolerance (default: tol).
- exdqlm.tol_gamma: gamma-delta convergence tolerance (default: tol).
- exdqlm.vb.min_iter: minimum iterations before convergence can trigger (default 10).
- exdqlm.vb.patience: number of consecutive joint-converged iterations required (default 3).
- exdqlm.use_cpp_samplers: use C++ samplers for s_t, u_t, theta (default FALSE). The GIG-based u_t sampler always uses the package C++ Devroye implementation; when FALSE, the remaining samplers fall back to R implementations.
- exdqlm.use_cpp_postpred: use C++ posterior predictive sampler (default FALSE).

Value

An object of class "exdqlmISVB" containing the following:

- y - Time-series data used to fit the model.
- run.time - Algorithm run time in seconds.
- iter - Number of iterations until convergence was reached.
- dqlm.ind - Logical value indicating whether gamma was fixed at 0, reducing the exDQLM to the special case of the DQLM.
- model - List of the state-space model including GG, FF, prior parameters m0 and C0.
- p0 - The quantile which was estimated.
- df - Discount factors used for each block.
- dim.df - Dimension used for each block of discount factors.
- sig.init - Initial value for sigma, or value at which sigma was fixed if fix.sigma=TRUE.
- seq.sigma - Sequence of sigma estimated by the algorithm until convergence.
- samp.theta - Posterior sample of the state vector variational distribution.
- samp.post.pred - Sample of the posterior predictive distributions.
- map.standard.forecast.errors - MAP standardized one-step-ahead forecast errors.
- samp.sigma - Posterior sample of scale parameter sigma variational distribution.
- samp.vts - Posterior sample of latent parameters, v_t, variational distributions.
- theta.out - List containing the variational distribution of the state vector including filtered distribution parameters (fm and fC) and smoothed distribution parameters (sm and sC).
- vts.out - List containing the variational distributions of latent parameters v_t.
- fix.sigma Logical value indicating whether sigma was fixed at sig.init.
- diagnostics - List containing ELBO trace, standardized VB iteration trace diagnostics$vb_trace (iteration-wise ELBO / sigma / gamma / convergence deltas), and convergence diagnostics (joint stopping status, deltas for state/sigma/gamma/ELBO, and criteria used).

If dqlm.ind=FALSE, the object also contains:

- gam.init - Initial value for gamma, or value at which gamma was fixed if fix.gamma=TRUE.
- seq.gamma - Sequence of gamma estimated by the algorithm until convergence.
- samp.gamma - Posterior sample of skewness parameter gamma variational distribution.
- samp.sts - Posterior sample of latent parameters, s_t, variational distributions.
- gammasig.out - List containing the IS estimate of the variational distribution of sigma and gamma.
- sts.out - List containing the variational distributions of latent parameters s_t.
- fix.gamma Logical value indicating whether gamma was fixed at gam.init.

Or if dqlm.ind=TRUE, the object also contains:

- sig.out - As above but for the DQLM case (gamma = 0); list containing the IS estimate of the variational distribution of sigma.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 20L)
y = scIVTmag[1:120]
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
seas.comp = seasMod(365, c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
# Legacy ISVB fit retained for backward-compatible comparisons
M0 = exdqlmISVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                gam.init = -3.5, sig.init = 15, tol = 0.2,
                n.IS = 20, n.samp = 20, verbose = FALSE)
head(M0$diagnostics$vb_trace)

M0_al = exdqlmISVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                   dqlm.ind = TRUE, sig.init = 15, tol = 0.2,
                   n.IS = 20, n.samp = 20, verbose = FALSE)
tail(M0_al$diagnostics$vb_trace$elbo, 2)
options(old)
```

`exdqlmLDVB` *exDQLM - LDVB algorithm (Laplace-Delta)*

Description

The function applies a Laplace-Delta Variational Bayes (LDVB) algorithm to estimate the posterior of an exDQLM.

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.
`model`	List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
`df`	Discount factors for each block.
`dim.df`	Dimension of each block of discount factors.
`fix.gamma`	Logical value indicating whether to fix gamma at `gam.init`. Default is `FALSE`.
`gam.init`	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if `fix.gamma=TRUE`.
`fix.sigma`	Logical value indicating whether to fix sigma at `sig.init`. Default is `FALSE`.
`sig.init`	Initial value for sigma (scale parameter), or value at which sigma will be fixed if `fix.sigma=TRUE`.
`dqlm.ind`	Logical value indicating whether to fix gamma at `0`, reducing the exDQLM to the special case of the DQLM. Default is `FALSE`.

`exps0`	Initial value for dynamic quantile. If `exps0` is not specified, it is set to the DLM estimate of the `p0` quantile.
`tol`	Tolerance for convergence of dynamic quantile estimates. Default is `tol=0.1`.
`n.samp`	Number of samples to draw from the approximated posterior distribution. Default is `n.samp=200`.
`PriorSigma`	List of parameters for inverse gamma prior on sigma; shape `a_sig` and scale `b_sig`. Default is an inverse gamma with mean 1 (or `sig.init` if provided) and variance 10.
`PriorGamma`	List of parameters for truncated student-t prior on gamma; center `m_gam`, scale `s_gam` and degrees of freedom `df_gam`. Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
`vb_control`	Optional normalized VB control list, usually from `exal_make_vb_control()`. When supplied, the core VB arguments and warmup blocks are read from `vb_control` first and then merged with the explicit function arguments. When omitted, exAL-style VB fits use the package's conservative default `(sigma, gamma)` warmup profile automatically; explicit controls remain the advanced override path.
`verbose`	Logical value indicating whether progress should be displayed.
`debug_shapes`	Logical; if TRUE, print KF input/output shapes every `debug_every` iterations.
`debug_every`	Integer; frequency (in iterations) for shape prints when `debug_shapes=TRUE`.

Details

Advanced options (set via `options()`):

- `exdqlm.use_cpp_kf`: use the C++ Kalman filter bridge (default TRUE).
- `exdqlm.compute_elbo`: compute ELBO every iteration (default TRUE).
- `exdqlm.tol_elbo`: ELBO convergence tolerance (default 1e-6).
- `exdqlm.tol_sigma`: sigma-delta convergence tolerance (default: `tol`).
- `exdqlm.tol_gamma`: gamma-delta convergence tolerance (default: `tol`).
- `exdqlm.vb.min_iter`: minimum iterations before convergence can trigger (default 10).
- `exdqlm.vb.patience`: number of consecutive joint-converged iterations required (default 3).
- `exdqlm.use_cpp_samplers`: use C++ samplers for s_t, u_t, theta (default FALSE). The GIG-based u_t sampler always uses the package C++ Devroye implementation; when FALSE, the remaining samplers fall back to R implementations.
- `exdqlm.use_cpp_postpred`: use C++ posterior predictive sampler (default FALSE).
- `exdqlm.dynamic.ldvb.sts`: optional warmup/freeze controls for the exDQLM latent s_t VB block. Supported fields are `freeze_warmup_iters`, `force_after_warmup`, and `min_postwarmup_updates`.

Value

An object of class "`exdqlmLDVB`" containing the following:

- `y` - Time-series data used to fit the model.

- run.time - Algorithm run time in seconds.
- iter - Number of iterations until convergence was reached.
- dqlm.ind - Logical value indicating whether gamma was fixed at 0, reducing the exDQLM to the special case of the DQLM.
- model - List of the state-space model including GG, FF, prior parameters m0 and C0.
- p0 - The quantile which was estimated.
- df - Discount factors used for each block.
- dim.df - Dimension used for each block of discount factors.
- sig.init - Initial value for sigma, or value at which sigma was fixed if fix.sigma=TRUE.
- seq.sigma - Sequence of sigma estimated by the algorithm until convergence.
- samp.theta - Posterior sample of the state vector variational distribution.
- samp.post.pred - Sample of the posterior predictive distributions.
- map.standard.forecast.errors - MAP standardized one-step-ahead forecast errors.
- samp.sigma - Posterior sample of scale parameter sigma variational distribution.
- samp.vts - Posterior sample of latent parameters, v_t, variational distributions.
- theta.out - List containing the variational distribution of the state vector including filtered distribution parameters (fm and fC) and smoothed distribution parameters (sm and sC).
- vts.out - List containing the variational distributions of latent parameters v_t.
- fix.sigma Logical value indicating whether sigma was fixed at sig.init.
- diagnostics - List containing ELBO trace, standardized VB iteration trace diagnostics$vb_trace (iteration-wise ELBO / sigma / gamma / convergence deltas), and convergence diagnostics (joint stopping status, deltas for state/sigma/gamma/ELBO, and criteria used).

If dqlm.ind=FALSE, the list also contains:

- gam.init - Initial value for gamma, or value at which gamma was fixed if fix.gamma=TRUE.
- seq.gamma - Sequence of gamma estimated by the algorithm until convergence.
- samp.gamma - Posterior sample of skewness parameter gamma variational distribution.
- samp.sts - Posterior sample of latent parameters, s_t, variational distributions.
- gammasig.out - List containing the LD (Laplace-Delta) approximation for the variational distribution of sigma and gamma (means, transformed Hessian, and ELBO components).
- sts.out - List containing the variational distributions of latent parameters s_t.
- fix.gamma Logical value indicating whether gamma was fixed at gam.init.

Or if dqlm.ind=TRUE, the list also contains:

- sig.out - As above but for the DQLM case (gamma = 0), the LD approximation for sigma.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 20L)
y = scIVTmag[1:80]
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
seas.comp = seasMod(365, c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                gam.init = -3.5, sig.init = 15, tol = 0.2,
                n.samp = 20, verbose = FALSE)

M0_al = exdqlmLDVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                   dqlm.ind = TRUE, sig.init = 15, tol = 0.2,
                   n.samp = 20, verbose = FALSE)
options(old)
```

---

exdqlmMCMC *exDQLM - MCMC algorithm*

Description

The function applies a Markov chain Monte Carlo (MCMC) algorithm to sample the posterior of an exDQLM.

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.
`model`	List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
`df`	Discount factors for each block.
`dim.df`	Dimension of each block of discount factors.
`fix.gamma`	Logical value indicating whether to fix gamma at `gam.init`. Default is `FALSE`.
`gam.init`	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if `fix.gamma = TRUE`.
`fix.sigma`	Logical value indicating whether to fix sigma at `sig.init`. Default is `FALSE`.
`sig.init`	Initial value for sigma (scale parameter), or value at which sigma will be fixed if `fix.sigma = TRUE`.
`dqlm.ind`	Logical value indicating whether to fix gamma at `0`, reducing the exDQLM to the AL/DQLM special case. Default is `FALSE`.
`Sig.mh`	Covariance matrix used in the random walk MH step to jointly sample sigma and gamma.
`joint.sample`	Logical value indicating whether or not to recompute `Sig.mh` based off the initial burn-in samples of gamma and sigma. Default is `FALSE`.

n.burn — Number of MCMC iterations to burn. Default is n.burn = 2000.

n.mcmc — Number of MCMC iterations to sample. Default is n.mcmc = 1500.

init.from.isvb — Logical value indicating whether to use the legacy ISVB warm start when init.from.vb = TRUE. Default is FALSE, which favors LDVB as the default VB warm start. This flag only chooses the warm-start source; it does not change the subsequent MCMC proposal kernel.

init.from.vb — Optional logical. If TRUE, run a VB pre-initialization step (LDVB by default, or ISVB when init.from.isvb = TRUE) and initialize MCMC from converged VB moments. Default is TRUE. If explicitly set to NULL, it falls back to init.from.isvb behavior for backward compatibility.

vb_init_controls — Optional list controlling VB warm start. Supported keys: method ("isvb" or "ldvb"), tol, n.IS, n.samp, max_iter, verbose.

vb_init_fit — Optional precomputed VB fit object. If supplied, warm start uses this object directly and does not rerun VB internally.

mcmc_control — Optional normalized MCMC control list, usually from exal_make_mcmc_control(). When supplied, the core MCMC arguments and warmup blocks are read from mcmc_control first and then merged with the explicit function arguments. When omitted, exAL-style dynamic MCMC uses the package's conservative default (sigma, gamma) warmup profile automatically; explicit controls remain the advanced override path.

sigmagam_controls — Optional list controlling warmup/freeze for the exDQLM sigma/gamma block during MCMC.

latent_state_controls — Optional list controlling early latent-state warmup/freeze in dynamic MCMC. Supported keys include freeze_burnin_iters, freeze_only_during_burn, force_after_warmup, and mode ("u_only" or "u_st_pair").

theta_state_controls — Optional list controlling early theta-state warmup/freeze in dynamic MCMC. Supported keys include freeze_burnin_iters, freeze_only_during_burn, and force_after_warmup.

dqlm_sigma_controls — Optional list controlling sigma-only warmup/freeze in the DQLM branch. Supported keys mirror sigmagam_controls.

mh.proposal — Character; proposal kernel for the exDQLM scale/skew block. "slice" (default) uses an exact sigma GIG update plus a bounded univariate slice sampler directly on gamma; "laplace_rw" uses a Laplace-informed covariance then RW; and "rw" uses joint random-walk MH on (log sigma, logit gamma). This choice is separate from the VB warm-start method.

mh.adapt — Logical; adapt MH proposal scale during burn-in.

mh.adapt.interval — Integer; adaptation interval (iterations).

mh.target.accept — Numeric length-2 vector with lower/upper target acceptance rates.

mh.scale.bounds
: Numeric length-2 vector with min/max global scaling for MH covariance.

mh.max_scale.step
: Numeric in (0,1); maximum fractional scale change per adaptation step.

mh.min_burn_adapt
: Minimum burn-in iterations required to enable adaptation.

slice.width
: Positive numeric width for the bounded slice sampler when mh.proposal = "slice". Default 0.1 for parity with bqrgal.

slice.max.steps
: Positive integer or Inf; maximum stepping-out expansions for the slice sampler.

trace.diagnostics
: Logical; if TRUE, retain per-iteration sigma/gamma/s/u diagnostics under mh.diagnostics$trace. Set FALSE for lighter-weight runs.

trace.every
: Positive integer; when trace.diagnostics = TRUE, record one diagnostics row every trace.every iterations.

verbose.every
: Positive integer controlling how often console progress is printed when verbose = TRUE. Default 500, independent of trace.every.

progress_callback
: Optional callback invoked with a named list at MCMC start, at each progress checkpoint, and on completion. Intended for workflow-level progress logging.

PriorSigma
: List of parameters for inverse gamma prior on sigma; shape a_sig and scale b_sig. Default is an inverse gamma with mean 1, or sig.init when supplied, and variance 10.

PriorGamma
: List of parameters for truncated Student-t prior on gamma; center m_gam, scale s_gam, and degrees of freedom df_gam. Default is a standard Student-t with 1 degree of freedom, truncated to the support of gamma.

verbose
: Logical value indicating whether progress should be displayed.

Value

An object of class "exdqlmMCMC" containing the following:

- y - Time-series data used to fit the model.
- run.time - Algorithm run time in seconds.
- dqlm.ind - Logical value indicating whether gamma was fixed at 0, reducing the exDQLM to the special case of the DQLM.
- model - List of the state-space model including GG, FF, prior parameters m0 and C0.
- p0 - The quantile which was estimated.
- df - Discount factors used for each block.
- dim.df - Dimension used for each block of discount factors.
- samp.theta - Posterior sample of the state vector.
- samp.post.pred - Sample of the posterior predictive distributions.
- map.standard.forecast.errors - MAP standardized one-step-ahead forecast errors.

- `samp.sigma` - Posterior sample of scale parameter sigma.
- `samp.vts` - Posterior sample of latent parameters, v_t.
- `theta.out` - List containing the distributions of the state vector including filtered distribution parameters (`fm` and `fC`) and smoothed distribution parameters (`sm` and `sC`).
- `n.burn` Number of MCMC iterations that were burned.
- `n.mcmc` Number of MCMC iterations that were sampled.

If `dqlm.ind=FALSE`, the object also contains the following:

- `samp.gamma` - Posterior sample of skewness parameter gamma.
- `samp.sts` - Posterior sample of latent parameters, s_t.
- `init.log.sigma` - Burned samples of log sigma from the random walk MH joint sampling of sigma and gamma.
- `init.logit.gamma` - Burned samples of logit gamma from the random walk MH joint sampling of sigma and gamma.
- `accept.rate` - Acceptance rate of the MH step.
- `accept.rate.burn` - MH acceptance rate during burn-in.
- `accept.rate.keep` - MH acceptance rate in kept MCMC samples.
- `Sig.mh` - Covariance matrix used in MH step to jointly sample sigma and gamma.
- `mh.diagnostics` - MH tuning diagnostics (proposal mode, scaling path, adaptation summary).
- `diagnostics` - ESS and chain-ready summaries for sigma/gamma.

Examples

```
data("scIVTmag", package = "exdqlm")
y = scIVTmag[1:80]
trend.comp = polytrendMod(order = 1, m0 = stats::quantile(y, 0.85), C0 = 10)
seas.comp = seasMod(p = 365, h = c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
M2 = exdqlmMCMC(y, p0=0.85, model, df = c(1,1), dim.df = c(1,4),
                gam.init = -3.5, sig.init = 15,
                n.burn = 40, n.mcmc = 40,
                init.from.vb = FALSE, verbose = FALSE)

M2_al = exdqlmMCMC(y, p0=0.85, model, df = c(1,1), dim.df = c(1,4),
                   dqlm.ind = TRUE, sig.init = 15,
                   n.burn = 30, n.mcmc = 30,
                   init.from.vb = FALSE, verbose = FALSE)
```

exdqlmPlot *Plot exDQLM*

Description

The function plots the MAP estimates and 95% credible intervals (CrIs) of the dynamic quantile of an exDQLM.

Usage

```
exdqlmPlot(
  m1,
  add = FALSE,
  col = "purple",
  cr.percent = 0.95,
  plot = TRUE,
  xlim = NULL,
  ylim = NULL,
  xlab = "time",
  ylab = NULL,
  lwd = 1.5,
  lwd.interval = 0.75,
  lty.interval = 2
)
```

Arguments

`m1`	An object of class "`exdqlmLDVB`", "`exdqlmMCMC`", or legacy "`exdqlmISVB`".
`add`	Logical value indicating whether the dynamic quantile will be added to existing plot. Default is `FALSE`.
`col`	Character vector of length 1 giving color of the dynamic quantile to be plotted. Default is `purple`.
`cr.percent`	Numeric in `(0, 1)` indicating the probability mass for the credible intervals (e.g., `0.95`). Default `0.95`.
`plot`	Logical value indicating whether to draw the plot. If `FALSE`, the function only returns the plotted summaries. Default is `TRUE`.
`xlim`, `ylim`	Optional limits passed to the base plotting call when `plot = TRUE`.
`xlab`, `ylab`	Optional axis labels passed to the base plotting call when `plot = TRUE`.
`lwd`, `lwd.interval`	Line widths for the dynamic quantile and credible interval bounds, respectively.
`lty.interval`	Line type for the credible interval bounds.

Value

A list of the following is returned:

- `map.quant` - MAP estimate of the dynamic quantile.
- `lb.quant` - Lower bound of the 95% CrIs of the dynamic quantile.
- `ub.quant` - Upper bound of the 95% CrIs of the dynamic quantile.
- `x` - Time/index values used for plotting.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.samp = 20, tol = 0.2, verbose = FALSE)
exdqlmPlot(M0, col = "blue")
q.summary = exdqlmPlot(M0, plot = FALSE)
options(old)
```

---

exdqlmTransferISVB *Transfer Function exDQLM - legacy ISVB algorithm*

Description

The function applies an Importance Sampling Variational Bayes (ISVB) algorithm to estimate the posterior of an exDQLM with exponential-decay transfer function component. This transfer wrapper is retained as a legacy path; `exdqlmTransferLDVB()` is the main VB transfer entry point.

Usage

```
exdqlmTransferISVB(
  y,
  p0,
  model,
  X,
  df,
  dim.df,
  lam,
  tf.df,
  fix.gamma = FALSE,
  gam.init = NA,
  fix.sigma = FALSE,
  sig.init = NA,
```

```
  dqlm.ind = FALSE,
  exps0,
  tol = 0.1,
  n.IS = 500,
  n.samp = 200,
  PriorSigma = NULL,
  PriorGamma = NULL,
  tf.m0 = NULL,
  tf.C0 = NULL,
  verbose = TRUE
)
```

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.
`model`	List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
`X`	A numeric vector or matrix of transfer-function inputs. Vectors are treated as a univariate input series. Matrices should have one row per time point and one column per covariate.
`df`	Discount factors for each block.
`dim.df`	Dimension of each block of discount factors.
`lam`	Transfer function rate parameter lambda, a value between 0 and 1.
`tf.df`	Discount factor specification for the transfer function component. If `length(tf.df) = 1`, the value is shared by the ζ_t state and the whole ψ_t block. If `length(tf.df) = 2`, it is interpreted as `c(df_zeta, df_psi_shared)`. If `length(tf.df) = k + 1`, where $k = ncol(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \ldots, \psi_{k,t})$.
`fix.gamma`	Logical value indicating whether to fix gamma at `gam.init`. Default is `FALSE`.
`gam.init`	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if `fix.gamma=TRUE`.
`fix.sigma`	Logical value indicating whether to fix sigma at `sig.init`. Default is `FALSE`.
`sig.init`	Initial value for sigma (scale parameter), or value at which sigma will be fixed if `fix.sigma=TRUE`.
`dqlm.ind`	Logical value indicating whether to fix gamma at `0`, reducing the exDQLM to the special case of the DQLM. Default is `FALSE`.
`exps0`	Initial value for dynamic quantile. If `exps0` is not specified, it is set to the DLM estimate of the `p0` quantile.
`tol`	Tolerance for convergence of dynamic quantile estimates. Default is `tol=0.1`.
`n.IS`	Number of particles for the importance sampling of joint variational distribution of sigma and gamma. Default is `n.IS=500`.
`n.samp`	Number of samples to draw from the approximated posterior distribution. Default is `n.samp=200`.

PriorSigma — List of parameters for inverse gamma prior on sigma; shape a_sig and scale b_sig. Default is an inverse gamma with mean 1 (or sig.init if provided) and variance 10.

PriorGamma — List of parameters for truncated student-t prior on gamma; center m_gam, scale s_gam and degrees of freedom df_gam. Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.

tf.m0 — Prior mean of the transfer function component. Defaults to a zero vector of length $k + 1$, where $k = ncol(X)$.

tf.C0 — Prior covariance of the transfer function component. Defaults to the $(k + 1) \times (k + 1)$ identity matrix.

verbose — Logical value indicating whether progress should be displayed.

Details

Advanced options (set via options()):

- exdqlm.use_cpp_kf: use the C++ Kalman filter bridge (default TRUE).
- exdqlm.compute_elbo: compute ELBO every iteration (default TRUE).
- exdqlm.tol_elbo: ELBO convergence tolerance (default 1e-6).
- exdqlm.use_cpp_samplers: use C++ samplers for s_t, u_t, theta (default FALSE). The GIG-based u_t sampler always uses the package C++ Devroye implementation; when FALSE, the remaining samplers fall back to R implementations.
- exdqlm.use_cpp_postpred: use C++ posterior predictive sampler (default FALSE).

Value

An object of class "exdqlmISVB" containing the following:

- run.time - Algorithm run time in seconds.
- iter - Number of iterations until convergence was reached.
- dqlm.ind - Logical value indicating whether gamma was fixed at 0, reducing the exDQLM to the special case of the DQLM.
- model - List of the augmented state-space model including GG, FF, prior parameters m0 and C0.
- p0 - The quantile which was estimated.
- df - Discount factors used for each block, including transfer function component.
- dim.df - Dimension used for each block of discount factors, including transfer function component.
- lam - Transfer function rate parameter lambda.
- sig.init - Initial value for sigma, or value at which sigma was fixed if fix.sigma=TRUE.
- seq.sigma - Sequence of sigma estimated by the algorithm until convergence.
- samp.theta - Posterior sample of the state vector variational distribution.
- samp.post.pred - Sample of the posterior predictive distributions.
- map.standard.forecast.errors - MAP standardized one-step-ahead forecast errors.

- `samp.sigma` - Posterior sample of scale parameter sigma variational distribution.
- `samp.vts` - Posterior sample of latent parameters, v_t, variational distributions.
- `theta.out` - List containing the variational distribution of the state vector including filtered distribution parameters (`fm` and `fC`) and smoothed distribution parameters (`sm` and `sC`).
- `vts.out` - List containing the variational distributions of latent parameters v_t.
- `median.kt` - Median number of time steps until the aggregated transfer effect $|x_t^\top \psi_{t-1}|$ is less than or equal to 1e-3.

If `dqlm.ind=FALSE`, the object also contains:

- `gam.init` - Initial value for gamma, or value at which gamma was fixed if `fix.gamma=TRUE`.
- `seq.gamma` - Sequence of gamma estimated by the algorithm until convergence.
- `samp.gamma` - Posterior sample of skewness parameter gamma variational distribution.
- `samp.sts` - Posterior sample of latent parameters, s_t, variational distributions.
- `gammasig.out` - List containing the IS estimate of the variational distribution of `sigma` and `gamma`.
- `sts.out` - List containing the variational distributions of latent parameters s_t.

Or if `dqlm.ind=TRUE`, the object also contains:

- `sig.out` - As above but for the DQLM case (`gamma = 0`); list containing the IS estimate of the variational distribution of sigma.

Examples

```
data("scIVTmag", package = "exdqlm")
data("ELIanoms", package = "exdqlm")
old = options(exdqlm.max_iter = 20L)
y = scIVTmag[1:120]
X = ELIanoms[1:120]
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
seas.comp = seasMod(365, c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
# Legacy ISVB transfer fit retained for backward-compatible comparisons
M1 = exdqlmTransferISVB(y, p0 = 0.85, model = model,
                         X, df = c(1,1), dim.df = c(1,4),
                         gam.init = -3.5, sig.init = 15,
                         lam = 0.38, tf.df = c(0.97,0.97),
                         n.IS = 20, n.samp = 20, tol = 0.2,
                         verbose = FALSE)
X_multi = cbind(ELIanoms[1:120], scale(scIVTmag[1:120])[, 1])
M2 = exdqlmTransferISVB(y, p0 = 0.85, model = model,
                         X_multi, df = c(1,1), dim.df = c(1,4),
                         gam.init = -3.5, sig.init = 15,
                         lam = 0.38, tf.df = c(0.97, 0.99),
                         n.IS = 20, n.samp = 20, tol = 0.2,
                         verbose = FALSE)
options(old)
```

exdqlmTransferLDVB *Transfer Function exDQLM - LDVB algorithm*

Description

The function applies a Laplace-Delta Variational Bayes (LDVB) algorithm to estimate the posterior of an exDQLM with an exponential-decay transfer function component. For multivariate transfer inputs, each column of `X` has its own instantaneous coefficient state in ψ_t, while a single scalar decay rate `lam` controls persistence of the accumulated transfer effect ζ_t.

Usage

```
exdqlmTransferLDVB(
  y,
  p0,
  model,
  X,
  df,
  dim.df,
  lam,
  tf.df,
  fix.gamma = FALSE,
  gam.init = NA,
  fix.sigma = FALSE,
  sig.init = NA,
  dqlm.ind = FALSE,
  exps0,
  tol = 0.1,
  n.samp = 200,
  PriorSigma = NULL,
  PriorGamma = NULL,
  tf.m0 = NULL,
  tf.C0 = NULL,
  verbose = TRUE,
  debug_shapes = FALSE,
  debug_every = 5
)
```

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.
`model`	List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
`X`	A numeric vector or matrix of transfer-function inputs. Vectors are treated as a univariate input series. Matrices should have one row per time point and one column per covariate.

df	Discount factors for each block.
dim.df	Dimension of each block of discount factors.
lam	Single transfer-function decay-rate parameter λ, a value between 0 and 1. This scalar is shared across all transfer inputs and controls propagation of the accumulated transfer effect ζ_t.
tf.df	Discount factor specification for the transfer function component. These discount factors control the evolution variances of the transfer states, separately from the deterministic decay rate `lam`. If `length(tf.df) = 1`, the value is shared by the ζ_t state and the whole ψ_t block. If `length(tf.df) = 2`, it is interpreted as `c(df_zeta, df_psi_shared)`. If `length(tf.df) = k + 1`, where $k = ncol(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \ldots, \psi_{k,t})$.
fix.gamma	Logical value indicating whether to fix gamma at `gam.init`. Default is `FALSE`.
gam.init	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if `fix.gamma=TRUE`.
fix.sigma	Logical value indicating whether to fix sigma at `sig.init`. Default is `FALSE`.
sig.init	Initial value for sigma (scale parameter), or value at which sigma will be fixed if `fix.sigma=TRUE`.
dqlm.ind	Logical value indicating whether to fix gamma at `0`, reducing the exDQLM to the special case of the DQLM. Default is `FALSE`.
exps0	Initial value for dynamic quantile. If `exps0` is not specified, it is set to the DLM estimate of the `p0` quantile.
tol	Tolerance for convergence of dynamic quantile estimates. Default is `tol=0.1`.
n.samp	Number of samples to draw from the approximated posterior distribution. Default is `n.samp=200`.
PriorSigma	List of parameters for inverse gamma prior on sigma; shape `a_sig` and scale `b_sig`. Default is an inverse gamma with mean 1 (or `sig.init` if provided) and variance 10.
PriorGamma	List of parameters for truncated student-t prior on gamma; center `m_gam`, scale `s_gam` and degrees of freedom `df_gam`. Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
tf.m0	Prior mean of the transfer function component. Defaults to a zero vector of length $k + 1$, where $k = ncol(X)$.
tf.C0	Prior covariance of the transfer function component. Defaults to the $(k + 1) \times (k + 1)$ identity matrix.
verbose	Logical value indicating whether progress should be displayed.
debug_shapes	Logical; if TRUE, print KF input/output shapes every `debug_every` iterations.
debug_every	Integer; frequency (in iterations) for shape prints when `debug_shapes=TRUE`.

Value

An object of class `"exdqlmLDVB"` containing the following:

- `y` - Time-series data used to fit the model.

- `run.time` - Algorithm run time in seconds.
- `iter` - Number of iterations until convergence was reached.
- `dqlm.ind` - Logical value indicating whether gamma was fixed at `0`, reducing the exDQLM to the special case of the DQLM.
- `model` - List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
- `p0` - The quantile which was estimated.
- `df` - Discount factors used for each block.
- `dim.df` - Dimension used for each block of discount factors.
- `sig.init` - Initial value for sigma, or value at which sigma was fixed if `fix.sigma=TRUE`.
- `seq.sigma` - Sequence of sigma estimated by the algorithm until convergence.
- `samp.theta` - Posterior sample of the state vector variational distribution.
- `samp.post.pred` - Sample of the posterior predictive distributions.
- `map.standard.forecast.errors` - MAP standardized one-step-ahead forecast errors.
- `samp.sigma` - Posterior sample of scale parameter sigma variational distribution.
- `samp.vts` - Posterior sample of latent parameters, v_t, variational distributions.
- `theta.out` - List containing the variational distribution of the state vector including filtered distribution parameters (`fm` and `fC`) and smoothed distribution parameters (`sm` and `sC`).
- `vts.out` - List containing the variational distributions of latent parameters v_t.
- `fix.sigma` Logical value indicating whether sigma was fixed at `sig.init`.
- `diagnostics` - List containing ELBO trace, standardized VB iteration trace `diagnostics$vb_trace` (iteration-wise ELBO / sigma / gamma / convergence deltas), and convergence diagnostics (joint stopping status, deltas for state/sigma/gamma/ELBO, and criteria used).

If `dqlm.ind=FALSE`, the list also contains:

- `gam.init` - Initial value for gamma, or value at which gamma was fixed if `fix.gamma=TRUE`.
- `seq.gamma` - Sequence of gamma estimated by the algorithm until convergence.
- `samp.gamma` - Posterior sample of skewness parameter gamma variational distribution.
- `samp.sts` - Posterior sample of latent parameters, s_t, variational distributions.
- `gammasig.out` - List containing the LD (Laplace-Delta) approximation for the variational distribution of `sigma` and `gamma` (means, transformed Hessian, and ELBO components).
- `sts.out` - List containing the variational distributions of latent parameters s_t.
- `fix.gamma` Logical value indicating whether gamma was fixed at `gam.init`.

Or if `dqlm.ind=TRUE`, the list also contains:

- `sig.out` - As above but for the DQLM case (`gamma = 0`), the LD approximation for `sigma`.

Transfer-function return fields

In addition to the standard `exdqlmLDVB()` return values, the returned `model`, `df`, and `dim.df` entries correspond to the transfer-function-augmented state-space model, with appended ζ_t and ψ_t states. The object also contains:

- `lam` - Single transfer-function decay-rate parameter λ.
- `median.kt` - Median number of time steps until the aggregated transfer effect $|x_t^\top \psi_{t-1}|$ is less than or equal to `1e-3`.
- `transfer_input_names` - Column names of the transfer inputs after normalization of `X`.

Examples

```
data("scIVTmag", package = "exdqlm")
data("ELIanoms", package = "exdqlm")
old = options(exdqlm.max_iter = 20L)
y = scIVTmag[1:120]
X = ELIanoms[1:120]
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
seas.comp = seasMod(365, c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
M1 = exdqlmTransferLDVB(
  y, p0 = 0.85, model = model, X = X,
  df = c(1,1), dim.df = c(1,4),
  gam.init = -3.5, sig.init = 15,
  lam = 0.38, tf.df = c(0.97,0.97),
  n.samp = 20, tol = 0.2, verbose = FALSE
)
X_multi = cbind(ELIanoms[1:120], scale(scIVTmag[1:120])[, 1])
M2 = exdqlmTransferLDVB(
  y, p0 = 0.85, model = model, X = X_multi,
  df = c(1,1), dim.df = c(1,4),
  gam.init = -3.5, sig.init = 15,
  lam = 0.38, tf.df = c(0.97, 0.99),
  n.samp = 20, tol = 0.2, verbose = FALSE
)
options(old)
```

`exdqlmTransferMCMC` *Transfer Function exDQLM - MCMC algorithm*

Description

The function applies a Markov chain Monte Carlo (MCMC) algorithm to sample the posterior of an exDQLM with an exponential-decay transfer function component for a fixed transfer rate parameter `lam`. For multivariate transfer inputs, each column of `X` has its own instantaneous coefficient state in ψ_t, while a single scalar decay rate `lam` controls persistence of the accumulated transfer effect ζ_t.

Usage

```
exdqlmTransferMCMC(
  y,
  p0,
  model,
  X,
  df,
  dim.df,
  lam,
  tf.df,
  fix.gamma = FALSE,
  gam.init = NA,
  fix.sigma = FALSE,
  sig.init = NA,
  dqlm.ind = FALSE,
  Sig.mh,
  joint.sample = FALSE,
  n.burn = 2000,
  n.mcmc = 1500,
  init.from.isvb = FALSE,
  PriorSigma = NULL,
  PriorGamma = NULL,
  verbose = TRUE,
  init.from.vb = TRUE,
  vb_init_controls = NULL,
  vb_init_fit = NULL,
  mh.proposal = c("slice", "laplace_rw", "rw"),
  mh.adapt = TRUE,
  mh.adapt.interval = 50L,
  mh.target.accept = c(0.2, 0.45),
  mh.scale.bounds = c(0.1, 10),
  mh.max_scale.step = 0.35,
  mh.min_burn_adapt = 50L,
  slice.width = 0.1,
  slice.max.steps = Inf,
  trace.diagnostics = TRUE,
  trace.every = 1L,
  verbose.every = 500L,
  progress_callback = NULL,
  tf.m0 = NULL,
  tf.C0 = NULL
)
```

Arguments

`y`	A univariate time-series.
`p0`	The quantile of interest, a value between 0 and 1.
`model`	List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.

X	A numeric vector or matrix of transfer-function inputs. Vectors are treated as a univariate input series. Matrices should have one row per time point and one column per covariate.
df	Discount factors for each block.
dim.df	Dimension of each block of discount factors.
lam	Single transfer-function decay-rate parameter λ, a value between 0 and 1. This scalar is shared across all transfer inputs and controls propagation of the accumulated transfer effect ζ_t.
tf.df	Discount factor specification for the transfer function component. These discount factors control the evolution variances of the transfer states, separately from the deterministic decay rate `lam`. If `length(tf.df) = 1`, the value is shared by the ζ_t state and the whole ψ_t block. If `length(tf.df) = 2`, it is interpreted as `c(df_zeta, df_psi_shared)`. If `length(tf.df) = k + 1`, where $k = ncol(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \ldots, \psi_{k,t})$.
fix.gamma	Logical value indicating whether to fix gamma at `gam.init`. Default is `FALSE`.
gam.init	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if `fix.gamma = TRUE`.
fix.sigma	Logical value indicating whether to fix sigma at `sig.init`. Default is `FALSE`.
sig.init	Initial value for sigma (scale parameter), or value at which sigma will be fixed if `fix.sigma = TRUE`.
dqlm.ind	Logical value indicating whether to fix gamma at `0`, reducing the exDQLM to the AL/DQLM special case. Default is `FALSE`.
Sig.mh	Covariance matrix used in the random walk MH step to jointly sample sigma and gamma.
joint.sample	Logical value indicating whether or not to recompute `Sig.mh` based off the initial burn-in samples of gamma and sigma. Default is `FALSE`.
n.burn	Number of MCMC iterations to burn. Default is `n.burn = 2000`.
n.mcmc	Number of MCMC iterations to sample. Default is `n.mcmc = 1500`.
init.from.isvb	Logical value indicating whether to use the legacy ISVB warm start when `init.from.vb = TRUE`. Default is `FALSE`, which favors `LDVB` as the default VB warm start. This flag only chooses the warm-start source; it does not change the subsequent MCMC proposal kernel.
PriorSigma	List of parameters for inverse gamma prior on sigma; shape `a_sig` and scale `b_sig`. Default is an inverse gamma with mean 1, or `sig.init` when supplied, and variance 10.
PriorGamma	List of parameters for truncated Student-t prior on gamma; center `m_gam`, scale `s_gam`, and degrees of freedom `df_gam`. Default is a standard Student-t with 1 degree of freedom, truncated to the support of gamma.
verbose	Logical value indicating whether progress should be displayed.
init.from.vb	Optional logical. If `TRUE`, run a VB pre-initialization step (`LDVB` by default, or `ISVB` when `init.from.isvb = TRUE`) and initialize MCMC from converged VB moments. Default is `TRUE`. If explicitly set to `NULL`, it falls back to `init.from.isvb` behavior for backward compatibility.

vb_init_controls
: Optional list controlling VB warm start. Supported keys: method ("isvb" or "ldvb"), tol, n.IS, n.samp, max_iter, verbose.

vb_init_fit
: Optional precomputed VB fit object. If supplied, warm start uses this object directly and does not rerun VB internally.

mh.proposal
: Character; proposal kernel for the exDQLM scale/skew block. "slice" (default) uses an exact sigma GIG update plus a bounded univariate slice sampler directly on gamma; "laplace_rw" uses a Laplace-informed covariance then RW; and "rw" uses joint random-walk MH on (log sigma, logit gamma). This choice is separate from the VB warm-start method.

mh.adapt
: Logical; adapt MH proposal scale during burn-in.

mh.adapt.interval
: Integer; adaptation interval (iterations).

mh.target.accept
: Numeric length-2 vector with lower/upper target acceptance rates.

mh.scale.bounds
: Numeric length-2 vector with min/max global scaling for MH covariance.

mh.max_scale.step
: Numeric in (0,1); maximum fractional scale change per adaptation step.

mh.min_burn_adapt
: Minimum burn-in iterations required to enable adaptation.

slice.width
: Positive numeric width for the bounded slice sampler when mh.proposal = "slice". Default 0.1 for parity with bqrgal.

slice.max.steps
: Positive integer or Inf; maximum stepping-out expansions for the slice sampler.

trace.diagnostics
: Logical; if TRUE, retain per-iteration sigma/gamma/s/u diagnostics under mh.diagnostics$trace. Set FALSE for lighter-weight runs.

trace.every
: Positive integer; when trace.diagnostics = TRUE, record one diagnostics row every trace.every iterations.

verbose.every
: Positive integer controlling how often console progress is printed when verbose = TRUE. Default 500, independent of trace.every.

progress_callback
: Optional callback invoked with a named list at MCMC start, at each progress checkpoint, and on completion. Intended for workflow-level progress logging.

tf.m0
: Prior mean of the transfer function component. Defaults to a zero vector of length $k+1$, where $k = ncol(X)$.

tf.C0
: Prior covariance of the transfer function component. Defaults to the $(k+1) \times (k+1)$ identity matrix.

Value

An object of class "exdqlmMCMC" containing the following:

- y - Time-series data used to fit the model.

- `run.time` - Algorithm run time in seconds.
- `dqlm.ind` - Logical value indicating whether gamma was fixed at `0`, reducing the exDQLM to the special case of the DQLM.
- `model` - List of the state-space model including `GG`, `FF`, prior parameters `m0` and `C0`.
- `p0` - The quantile which was estimated.
- `df` - Discount factors used for each block.
- `dim.df` - Dimension used for each block of discount factors.
- `samp.theta` - Posterior sample of the state vector.
- `samp.post.pred` - Sample of the posterior predictive distributions.
- `map.standard.forecast.errors` - MAP standardized one-step-ahead forecast errors.
- `samp.sigma` - Posterior sample of scale parameter sigma.
- `samp.vts` - Posterior sample of latent parameters, v_t.
- `theta.out` - List containing the distributions of the state vector including filtered distribution parameters (`fm` and `fC`) and smoothed distribution parameters (`sm` and `sC`).
- `n.burn` Number of MCMC iterations that were burned.
- `n.mcmc` Number of MCMC iterations that were sampled.

If `dqlm.ind=FALSE`, the object also contains the following:

- `samp.gamma` - Posterior sample of skewness parameter gamma.
- `samp.sts` - Posterior sample of latent parameters, s_t.
- `init.log.sigma` - Burned samples of log sigma from the random walk MH joint sampling of sigma and gamma.
- `init.logit.gamma` - Burned samples of logit gamma from the random walk MH joint sampling of sigma and gamma.
- `accept.rate` - Acceptance rate of the MH step.
- `accept.rate.burn` - MH acceptance rate during burn-in.
- `accept.rate.keep` - MH acceptance rate in kept MCMC samples.
- `Sig.mh` - Covariance matrix used in MH step to jointly sample sigma and gamma.
- `mh.diagnostics` - MH tuning diagnostics (proposal mode, scaling path, adaptation summary).
- `diagnostics` - ESS and chain-ready summaries for sigma/gamma.

Transfer-function return fields

In addition to the standard `exdqlmMCMC()` return values, the returned `model`, `df`, and `dim.df` entries correspond to the transfer-function-augmented state-space model, with appended ζ_t and ψ_t states. The object also contains:

- `lam` - Single transfer-function decay-rate parameter λ.
- `median.kt` - Median number of time steps until the aggregated transfer effect $|x_t^\top \psi_{t-1}|$ is less than or equal to `1e-3`.
- `transfer_input_names` - Column names of the transfer inputs after normalization of `X`.

Examples

```
data("scIVTmag", package = "exdqlm")
data("ELIanoms", package = "exdqlm")
y = scIVTmag[1:120]
X = ELIanoms[1:120]
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
seas.comp = seasMod(365, c(1,2), C0 = 10*diag(4))
model = trend.comp + seas.comp
M1 = exdqlmTransferMCMC(
  y, p0 = 0.85, model = model, X = X,
  df = c(1,1), dim.df = c(1,4),
  gam.init = -3.5, sig.init = 15,
  lam = 0.38, tf.df = c(0.97,0.97),
  n.burn = 40, n.mcmc = 40,
  init.from.vb = FALSE, verbose = FALSE
)
X_multi = cbind(ELIanoms[1:120], scale(scIVTmag[1:120])[, 1])
M2 = exdqlmTransferMCMC(
  y, p0 = 0.85, model = model, X = X_multi,
  df = c(1,1), dim.df = c(1,4),
  gam.init = -3.5, sig.init = 15,
  lam = 0.38, tf.df = c(0.97, 0.99),
  n.burn = 40, n.mcmc = 40,
  init.from.vb = FALSE, verbose = FALSE
)
```

`get_gamma_bounds` *Bounds for the exAL shape parameter gamma*

Description

Returns valid lower/upper bounds `(L, U)` for the shape parameter `gamma` of the standardized extended Asymmetric Laplace (exAL), given `p0` in (0,1).

Usage

```
get_gamma_bounds(p0)
```

Arguments

`p0`	Numeric scalar in (0, 1); typically the target quantile level.

Details

This is a user-facing convenience wrapper around the C++ routine `get_gamma_bounds_cpp()`, which performs the actual computation.

Value

A numeric vector of length 2 named `c("L","U")`.

Examples

```
get_gamma_bounds(0.5)
get_gamma_bounds(0.9)
```

`is.exalStaticDiagnostic` `exalStaticDiagnostic` *objects*

Description

`is.exalStaticDiagnostic` tests if its argument is an `exalStaticDiagnostic` object.

Usage

```
is.exalStaticDiagnostic(x)
```

Arguments

`x`	an **R** object

`is.exalStaticLDVB` `exalStaticLDVB` *objects*

Description

`is.exalStaticLDVB` tests if its argument is an `exalStaticLDVB` object.

Usage

```
is.exalStaticLDVB(m)
```

Arguments

`m`	an **R** object

is.exalStaticMCMC — exalStaticMCMC *objects*

Description

is.exalStaticMCMC tests if its argument is an exalStaticMCMC object.

Usage

```
is.exalStaticMCMC(m)
```

Arguments

m	an **R** object

is.exdqlm — exdqlm *objects*

Description

is.exdqlm tests if its argument is a exdqlm object.

Usage

```
is.exdqlm(m)
```

Arguments

m	an **R** object

is.exdqlmDiagnostic — exdqlmDiagnostic *objects*

Description

is.exdqlmDiagnostic tests if its argument is a exdqlmDiagnostic object.

Usage

```
is.exdqlmDiagnostic(x)
```

Arguments

x	an **R** object

`is.exdqlmForecast` — `exdqlmForecast` *objects*

Description

`is.exdqlmForecast` tests if its argument is a `exdqlmForecast` object.

Usage

```
is.exdqlmForecast(x)
```

Arguments

`x`	an **R** object

`is.exdqlmForecastDiagnostic` — `exdqlmForecastDiagnostic` *objects*

Description

`is.exdqlmForecastDiagnostic` tests if its argument is an `exdqlmForecastDiagnostic` object.

Usage

```
is.exdqlmForecastDiagnostic(x)
```

Arguments

`x`	an **R** object.

`is.exdqlmISVB` — `exdqlmISVB` *objects*

Description

`is.exdqlmISVB` tests if its argument is a `exdqlmISVB` object.

Usage

```
is.exdqlmISVB(m)
```

Arguments

`m`	an **R** object

is.exdqlmLDVB — exdqlmLDVB *objects*

Description

is.exdqlmLDVB tests if its argument is a exdqlmLDVB object.

Usage

```
is.exdqlmLDVB(m)
```

Arguments

m	an **R** object

is.exdqlmMCMC — exdqlmMCMC *objects*

Description

is.exdqlmMCMC tests if its argument is a exdqlmMCMC object.

Usage

```
is.exdqlmMCMC(m)
```

Arguments

m	an **R** object

is.exdqlmSynthesis — exdqlmSynthesis *objects*

Description

is.exdqlmSynthesis tests if its argument is an exdqlmSynthesis object returned by quantileSynthesis.

Usage

```
is.exdqlmSynthesis(x)
```

Arguments

x	an **R** object

pexal *Cumulative Distribution Function (CDF) for the exAL Distribution*

Description

Vectorized over q.

Usage

```
pexal(
  q,
  p0 = 0.5,
  mu = 0,
  sigma = 1,
  gamma = 0,
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

q	Numeric vector of quantiles.
p0	Probability level used in the quantile parametrization. Scalar in (0, 1). Default 0.5.
mu	Location parameter (scalar). Default 0.
sigma	Scale parameter (scalar, strictly positive). Default 1.
gamma	Skewness parameter controlling asymmetry (scalar). Must be within valid bounds implied by p0. Default 0.
lower.tail	Logical scalar; if TRUE (default) return $P(X \leq q)$, otherwise $P(X > q)$.
log.p	Logical scalar; if TRUE, return log-probabilities.

Value

Numeric vector of CDF values (same length as q).

Examples

```
pexal(0)
pexal(c(-1, 0, 1), p0 = 0.5, mu = 0, sigma = 1, gamma = 0.1)
```

plot.exalStaticDiagnostic — *Plot Method for* `exalStaticDiagnostic` *Objects*

Description

Plot Method for `exalStaticDiagnostic` Objects

Usage

```
## S3 method for class 'exalStaticDiagnostic'
plot(
  x,
  cols = c("red", "blue"),
  type = c("quantile", "coefficients"),
  beta.ref = NULL,
  include.intercept = TRUE,
  coef.names = NULL,
  xlab = NULL,
  ylab = NULL,
  ylim = NULL,
  legend.labels = NULL,
  beta.ref.label = "reference",
  legend = TRUE,
  ...
)
```

Arguments

`x`	An `exalStaticDiagnostic` object.
`cols`	Character vector of length 1 or 2 giving color(s) used to plot diagnostics.
`type`	Character string; `"quantile"` plots fitted conditional quantile summaries, and `"coefficients"` plots posterior coefficient intervals.
`beta.ref`	Optional coefficient reference vector for `type = "coefficients"`. This is typically available only in simulation benchmarks. It is used as a plotting overlay, not as a package diagnostic metric.
`include.intercept`	Logical; if `FALSE`, omit the first coefficient from `type = "coefficients"` plots.
`coef.names`	Optional names for coefficients in `type = "coefficients"` plots.
`xlab`, `ylab`	Optional axis labels.
`ylim`	Optional y-axis limits.
`legend.labels`	Optional labels for the first and second model intervals in `type = "coefficients"` plots.
`beta.ref.label`	Label for the optional `beta.ref` overlay.

legend	Logical; if TRUE, add a legend to coefficient plots.
...	Additional arguments passed to plotting functions.

---

plot.exalStaticLDVB *Plot Method for* exalStaticLDVB *Objects*

Description

Plot Method for exalStaticLDVB Objects

Usage

```
## S3 method for class 'exalStaticLDVB'
plot(x, X = NULL, add = FALSE, col = "purple", cr.percent = 0.95, ...)
```

Arguments

x	An exalStaticLDVB object.
X	Optional design matrix used to compute fitted quantiles. If omitted, the method uses x$X when available.
add	Logical; add to an existing plot.
col	Character vector of length 1 giving color for fitted quantiles.
cr.percent	Numeric in (0, 1) for credible-interval mass.
...	Additional arguments passed to plot when add = FALSE.

Value

A list with map.quant, lb.quant, and ub.quant.

---

plot.exalStaticMCMC *Plot Method for* exalStaticMCMC *Objects*

Description

Plot Method for exalStaticMCMC Objects

Usage

```
## S3 method for class 'exalStaticMCMC'
plot(x, add = FALSE, col = "purple", cr.percent = 0.95, ...)
```

Arguments

x	An exalStaticMCMC object.
add	Logical; add to an existing plot.
col	Character vector of length 1 giving color for fitted quantiles.
cr.percent	Numeric in (0, 1) for credible-interval mass.
...	Additional arguments passed to plot when add = FALSE.

Value

A list with map.quant, lb.quant, and ub.quant.

---

plot.exdqlmDiagnostic *Plot Method for* exdqlmDiagnostic *Objects*

Description

Plot Method for exdqlmDiagnostic Objects

Usage

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

Arguments

x	An exdqlmDiagnostic object.
...	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.95), dim.df = c(1),
                 gam.init = -3.5, sig.init = 15,
                 n.samp = 20, tol = 0.2, verbose = FALSE)
M0.diags = exdqlmDiagnostics(M0, plot = FALSE)
plot(M0.diags)
options(old)
```

plot.exdqlmForecast — *Plot Method for* exdqlmForecast *Objects*

Description

Plot Method for `exdqlmForecast` Objects

Usage

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

Arguments

`x`	An `exdqlmForecast` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.samp = 20, tol = 0.2, verbose = FALSE)
M0.forecast = exdqlmForecast(start.t = 50, k = 5, m1 = M0)
plot(M0.forecast)
options(old)
```

plot.exdqlmISVB — *Plot Method for* exdqlmISVB *Objects*

Description

Plot Method for `exdqlmISVB` Objects

Usage

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

Arguments

`x`	An `exdqlmISVB` object.
`...`	Additional arguments.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
# Legacy ISVB object retained for backward-compatible plotting methods
M0 = exdqlmISVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.IS = 20, n.samp = 20, tol = 0.2,
                   verbose = FALSE)
plot(M0)
options(old)
```

`plot.exdqlmLDVB` *Plot Method for* `exdqlmLDVB` *Objects*

Description

Plot Method for `exdqlmLDVB` Objects

Usage

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

Arguments

`x`	An `exdqlmLDVB` object.
`...`	Additional arguments.

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.samp = 20, tol = 0.2, verbose = FALSE)
plot(M0)
options(old)
```

plot.exdqlmMCMC — *Plot Method for* exdqlmMCMC *Objects*

Description

Plot Method for exdqlmMCMC Objects

Usage

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

Arguments

x	An exdqlmMCMC object.
...	Additional arguments.

Examples

```
data("scIVTmag", package = "exdqlm")
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M2 = exdqlmMCMC(y, p0=0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.burn = 20, n.mcmc = 20,
                init.from.vb = FALSE, verbose = FALSE)
plot(M2)
```

plot.exdqlmSynthesis — *Plot Method for* exdqlmSynthesis *Objects*

Description

Plot the pointwise posterior predictive interval produced by quantileSynthesis. The method is intentionally separate from quantileSynthesis() so the synthesis step remains a computation, while the returned object still has a standard plotting interface.

Usage

```
## S3 method for class 'exdqlmSynthesis'
plot(
  x,
  y = NULL,
  time = NULL,
```

```
  add = FALSE,
  interval = 0.95,
  show.median = TRUE,
  show.mean = FALSE,
  band.col = grDevices::adjustcolor("lightblue", alpha.f = 0.35),
  median.col = "blue",
  mean.col = "darkblue",
  y.col = "dark grey",
  border = NA,
  xlab = "time",
  ylab = "posterior predictive synthesis",
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`x`	An `exdqlmSynthesis` object.
`y`	Optional observed series to overlay.
`time`	Optional time vector for the synthesized summaries. If omitted, `seq_len(T)` is used, where `T` is the number of synthesized time points.
`add`	Logical; add the synthesis interval to an existing plot.
`interval`	Numeric in `(0, 1)` giving the plotted central interval. Currently `0.50` and `0.95` are supported from stored summaries.
`show.median`	Logical; draw the synthesized posterior median.
`show.mean`	Logical; draw the synthesized posterior mean.
`band.col`	Fill color for the predictive interval.
`median.col`	Color for the posterior median line.
`mean.col`	Color for the posterior mean line.
`y.col`	Color for the optional observed series.
`border`	Border color for the predictive interval polygon.
`xlab`, `ylab`, `main`	Graphical labels.
`xlim`, `ylim`	Optional axis limits.
`...`	Additional graphical arguments passed to the initial `plot()` call when `add = FALSE`.

polytrendMod — *Create an n-th order polynomial exDQLM component*

Description

The function creates an n-th order polynomial exDQLM component.

Usage

```
polytrendMod(order, m0, C0, backend = c("auto", "R", "cpp"))
```

Arguments

`order`	Numeric order n of the polynomial model.
`m0`	Optional numeric prior mean. Defaults to $n \times 1$ vector of zeros.
`C0`	Optional numeric prior covariance. Defaults to matrix $10^3 I_n$.
`backend`	Backend selection for matrix construction: `"auto"` (default), `"R"`, or `"cpp"`.

Value

An object of class `"exdqlm"` containing the following:

- `FF` - $n \times 1$ observational vector.
- `GG` - $n \times n$ evolution matrix.
- `m0` - $n \times 1$ prior mean of the state vector.
- `C0` - $n \times n$ prior covariance matrix of the state vector.

Examples

```
# create a second order polynomial component
trend.comp = polytrendMod(2, rep(0, 2), 10*diag(2))
```

print.exalStaticDiagnostic — *Print Method for* `exalStaticDiagnostic` *Objects*

Description

Print Method for `exalStaticDiagnostic` Objects

Usage

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

Arguments

`x`	An `exalStaticDiagnostic` object.
`...`	Additional arguments (unused).

---

print.exalStaticLDVB — *Print Method for* `exalStaticLDVB` *Objects*

Description

Print Method for `exalStaticLDVB` Objects

Usage

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

Arguments

`x`	An `exalStaticLDVB` object.
`...`	Additional arguments (unused).

---

print.exalStaticMCMC — *Print Method for* `exalStaticMCMC` *Objects*

Description

Print Method for `exalStaticMCMC` Objects

Usage

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

Arguments

`x`	An `exalStaticMCMC` object.
`...`	Additional arguments (unused).

`print.exdqlm` *Print exDQLM model details*

Description

Print the details of the exDQLM model.

Usage

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

Arguments

`x`	a `exdqlm` object.
`...`	further arguments (unused).

`print.exdqlmDiagnostic` *Print Method for* `exdqlmDiagnostic` *Objects*

Description

Print Method for `exdqlmDiagnostic` Objects

Usage

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

Arguments

`x`	An `exdqlmDiagnostic` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.95), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.samp = 20, tol = 0.2, verbose = FALSE)
M0.diags = exdqlmDiagnostics(M0, plot=FALSE)
print(M0.diags)
```

```
options(old)
```

`print.exdqlmForecast` *Print Method for* `exdqlmForecast` *Objects*

Description

Print Method for `exdqlmForecast` Objects

Usage

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

Arguments

`x`	An `exdqlmForecast` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                 gam.init = -3.5, sig.init = 15,
                 n.samp = 20, tol = 0.2, verbose = FALSE)
M0.forecast = exdqlmForecast(start.t = 50, k = 5, m1 = M0)
print(M0.forecast)
options(old)
```

`print.exdqlmForecastDiagnostic` *Print Method for* `exdqlmForecastDiagnostic` *Objects*

Description

Print Method for `exdqlmForecastDiagnostic` Objects

Usage

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

Arguments

x	An `exdqlmForecastDiagnostic` object.
...	Additional arguments (unused).

`print.exdqlmISVB` *Print Method for* `exdqlmISVB` *Objects*

Description

Print Method for `exdqlmISVB` Objects

Usage

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

Arguments

x	An `exdqlmISVB` object.
...	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
# Legacy ISVB object retained for backward-compatible inspection methods
M0 = exdqlmISVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.IS = 20, n.samp = 20, tol = 0.2,
                   verbose = FALSE)
print(M0)
options(old)
```

print.exdqlmLDVB *Print Method for* `exdqlmLDVB` *Objects*

Description

Print Method for `exdqlmLDVB` Objects

Usage

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

Arguments

`x`	An `exdqlmLDVB` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.samp = 20, tol = 0.2, verbose = FALSE)
print(M0)
options(old)
```

print.exdqlmMCMC *Print Method for* `exdqlmMCMC` *Objects*

Description

Print Method for `exdqlmMCMC` Objects

Usage

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

Arguments

`x`	An `exdqlmMCMC` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M2 = exdqlmMCMC(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.burn = 20, n.mcmc = 20,
                init.from.vb = FALSE, verbose = FALSE)
print(M2)
```

`print.exdqlmSynthesis` *Print Method for* `exdqlmSynthesis` *Objects*

Description

Print Method for `exdqlmSynthesis` Objects

Usage

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

Arguments

`x`	An `exdqlmSynthesis` object.
`...`	Additional arguments (unused).

`qexal` *Quantile Function for the exAL Distribution*

Description

Vectorized over p.

Usage

```
qexal(
  p,
  p0 = 0.5,
  mu = 0,
  sigma = 1,
  gamma = 0,
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

`p`	Numeric vector of probabilities in (0, 1).
`p0`	Probability level used in the quantile parametrization. Scalar in (0, 1). Default `0.5`.
`mu`	Location parameter (scalar). Default `0`.
`sigma`	Scale parameter (scalar, strictly positive). Default `1`.
`gamma`	Skewness parameter controlling asymmetry (scalar). Must be within valid bounds implied by `p0`. Default `0`.
`lower.tail`	Logical scalar; if `TRUE` (default) return $P(X \leq q)$, otherwise $P(X > q)$.
`log.p`	Logical scalar; if `TRUE`, return log-probabilities.

Value

Numeric vector of quantiles (same length as `p`).

Examples

```
p <- seq(0.1, 0.9, by = 0.2)
q <- qexal(p, p0 = 0.5, mu = 0, sigma = 1, gamma = 0)
all.equal(p, pexal(q, p0 = 0.5, mu = 0, sigma = 1, gamma = 0), tol = 1e-4)
```

`quantileSynthesis` *Synthesize a unified posterior predictive distribution from multiple quantile-model draws*

Description

The function synthesizes posterior predictive draws from multiple fitted quantile models into a single posterior predictive distribution. It uses a two-step correction: (i) isotonic regression at the grid of target quantiles to align the fitted quantile levels, and (ii) distributional alignment (shift each model's draws so its tau-quantile matches the isotone anchor). It then builds a single predictive quantile function per time by piecewise-linear blending across adjacent quantile models with optional global monotone rearrangement.

Usage

```
quantileSynthesis(
  draws_list,
  p,
  enforce_isotonic = TRUE,
  rearrange = TRUE,
  grid_M = 1001L,
  n_samp = 1000L,
  seed = NULL,
  T_expected = NULL
)
```

Arguments

- `draws_list` — List of length `L`; each element is either: (i) a numeric matrix of posterior predictive draws (`T x ns` or `ns x T`), (ii) a dynamic fit object (`exdqlmMCMC`, `exdqlmLDVB`, or legacy `exdqlmISVB`) with `samp.post.pred`, or (iii) an `exdqlmForecast` object with `samp.fore`. Rows are coerced to time.
- `p` — Numeric vector of target quantile levels in `(0,1)` of length `L` (same order as `draws_list`, not necessarily sorted). Duplicate levels are not allowed.
- `enforce_isotonic` — Logical; apply isotonic regression (PAVA) over the grid `p` at each time t to remove crossing. Default `TRUE`.
- `rearrange` — Logical; apply monotone rearrangement (evaluate -> sort -> reinterpolate) on a dense grid over `u in (0,1)`. Default `TRUE`.
- `grid_M` — Integer; size of dense grid `M` for rearrangement (`u_k = k/(M+1)`). Default `1001L`.
- `n_samp` — Integer; number of synthesized draws per time. Default `1000L`.
- `seed` — NULL or integer for reproducible synthesized draws. Default `NULL`.
- `T_expected` — Optional integer; if provided, forces the time dimension to `T_expected` when orienting each matrix to `T x ns`. This avoids accidental transposes.

Value

An object of class "`exdqlmSynthesis`", which is a list containing:

- `draws` - Numeric matrix `T x n_samp` of synthesized draws.
- `levels` - Sorted copy of `p` (length `L`).
- `quantiles` - Numeric matrix `T x L` of isotone anchors `m^*_{i,t}`.
- `summary` - List with row-wise summaries of `draws` (`mean`, `q025`, `q250`, `q500`, `q750`, `q975`).
- `method` - List of synthesis settings used (`name`, `isotonic`, `rearrange`, `grid_M`, `T_inferred`).

Examples

```
# short example
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 10L)
TT = 50
y = scIVTmag[1:TT]

# create a compact trend model
trend.comp = polytrendMod(1, stats::quantile(y, 0.85), 10)
model = trend.comp

# fit quantiles using LDVB and save posterior predictive samples
fits <- draws <- NULL
p0s = c(0.10, 0.50, 0.90)
for(i in 1:length(p0s)){
  fits[[i]] = exdqlmLDVB(
    y, p0 = p0s[i], model, df = 0.98, dim.df = 1,
    sig.init = 15, n.samp = 20, tol = 0.2, verbose = FALSE
```

```
  )
  draws[[i]] = fits[[i]]$samp.post.pred
}

# synthesize posterior predictive from all quantiles
syn = quantileSynthesis(
  draws_list = draws,
  p = p0s,
  T_expected = TT)

# alternatively, pass fitted dynamic objects directly
syn2 = quantileSynthesis(
  draws_list = fits,
  p = p0s,
  T_expected = TT)

# plot the synthesized 95% posterior predictive interval
plot(syn2, y = y)
options(old)
```

`regMod` *Create a standard regression component for an exDQLM*

Description

The function constructs a regression block where the observation vector at time t is $F_t = X_t$ (row of the design matrix), and the state evolves as $\theta_t = \theta_{t-1}$ (i.e., $G_t = I_n$).

Usage

```
regMod(X, m0, C0)
```

Arguments

`X`	A numeric matrix of dimension $T \times n$ (T time points, n regressors). Vectors are accepted and treated as $T \times 1$.
`m0`	Optional numeric prior mean (length n). Defaults to zeros.
`C0`	Optional numeric prior covariance ($n \times n$). Defaults to $10^3 I_n$.

Details

Input `X` is a $T \times n$ matrix of regressors; the returned `FF` is an $n \times T$ matrix (i.e., `t(X)`), consistent with component composition via `+.exdqlm`.

Value

An object of class "exdqlm" with elements:

- FF - $n \times T$ matrix with column t equal to $F_t = X_t$.
- GG - $n \times n$ identity matrix (static coefficients).
- m0, C0 - Prior mean/covariance for regression coefficients.

Examples

```
data("climateIndices", package = "exdqlm")

T <- 150
bt_dates <- seq(as.Date("1987-01-01"), by = "month", length.out = T)
idx <- match(bt_dates, climateIndices$date)
X <- scale(climateIndices[idx, c("noi", "amo")])

# Single regressor (T x 1)
reg1 = regMod(X[, "noi"])
# Multiple regressors (T x n)
reg2 = regMod(X)

# Combine with trend/seasonal components
trend.comp = polytrendMod(order = 3, m0 = rep(0,3), C0 = diag(3))
seas.comp  = seasMod(p = 12, h = 1, C0 = diag(1, 2))
base.mod   = trend.comp + seas.comp
model.std  = base.mod + reg2
```

---

rexal *Random Sample Generation for the exAL Distribution*

Description

Random Sample Generation for the exAL Distribution

Usage

```
rexal(n, p0 = 0.5, mu = 0, sigma = 1, gamma = 0)
```

Arguments

n	Positive integer number of samples to draw (scalar).
p0	Probability level used in the quantile parametrization. Scalar in (0, 1). Default 0.5.
mu	Location parameter (scalar). Default 0.
sigma	Scale parameter (scalar, strictly positive). Default 1.
gamma	Skewness parameter controlling asymmetry (scalar). Must be within valid bounds implied by p0. Default 0.

Value

Numeric vector of length n.

Examples

```
set.seed(1)
rexal(3, p0 = 0.5, mu = c(-1, 0, 1))
```

---

scIVTmag *Time series of daily average magnitude IVT in Santa Cruz, CA.*

Description

ECMWF Re-Analysis 5 (ERA5) daily average magnitude IVT in Santa Cruz, CA (approximately 22 N, 122 W) from January 1, 1979 to December 31, 2019 with all February 29ths omitted.

Usage

```
scIVTmag
```

Format

A time series of length 14965.

Source

https://cds.climate.copernicus.eu

References

Hersbach, H, Bell, B, Berrisford, P, et al. *The ERA5 global reanalysis.* Q J R Meteorol Soc. 2020; 146: 1999– 2049. doi:10.1002/qj.3803

---

seasMod *Create Fourier representation of a periodic exDQLM component*

Description

The function creates a Fourier form periodic component for given period and harmonics.

Usage

```
seasMod(p, h, m0, C0, backend = c("auto", "R", "cpp"))
```

Arguments

p	Numeric period.
h	Numeric vector of harmonics to be included.
m0	Optional numeric prior mean. Defaults to $q \times 1$ vector of zeros where q is the dimension of the period component.
C0	Optional numeric prior covariance. Defaults to matrix $10^3 I_q$.
backend	Backend selection for matrix construction: "auto" (default), "R", or "cpp".

Value

An object of class "exdqlm" containing the following:

- FF - $q \times 1$ observational vector.
- GG - $q \times q$ evolution matrix.
- m0 - $q \times 1$ prior mean of the state vector.
- C0 - $q \times q$ prior covariance matrix of the state vector.

Examples

```
# create a seasonal component with first, second and fourth harmonics of a period of 365
seas.comp = seasMod(365, c(1, 2, 4), C0 = 10*diag(6))
```

---

summary.exalStaticDiagnostic

Summary Method for exalStaticDiagnostic *Objects*

Description

Summary Method for exalStaticDiagnostic Objects

Usage

```
## S3 method for class 'exalStaticDiagnostic'
summary(object, ...)
```

Arguments

object	An exalStaticDiagnostic object.
...	Additional arguments (unused).

summary.exalStaticLDVB

Summary Method for `exalStaticLDVB` *Objects*

Description

Summary Method for `exalStaticLDVB` Objects

Usage

```
## S3 method for class 'exalStaticLDVB'
summary(object, ...)
```

Arguments

`object`	An `exalStaticLDVB` object.
`...`	Additional arguments (unused).

summary.exalStaticMCMC

Summary Method for `exalStaticMCMC` *Objects*

Description

Summary Method for `exalStaticMCMC` Objects

Usage

```
## S3 method for class 'exalStaticMCMC'
summary(object, ...)
```

Arguments

`object`	An `exalStaticMCMC` object.
`...`	Additional arguments (unused).

`summary.exdqlm` *Summary exDQLM model details*

Description

Print the details of the exDQLM model.

Usage

```
## S3 method for class 'exdqlm'
summary(object, ...)
```

Arguments

`object`	a `exdqlm` object.
`...`	further arguments (unused).

`summary.exdqlmDiagnostic` *Summary Method for* `exdqlmDiagnostic` *Objects*

Description

Summary Method for `exdqlmDiagnostic` Objects

Usage

```
## S3 method for class 'exdqlmDiagnostic'
summary(object, ...)
```

Arguments

`object`	An `exdqlmDiagnostic` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.95), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.samp = 20, tol = 0.2, verbose = FALSE)
M0.diags = exdqlmDiagnostics(M0, plot = FALSE)
summary(M0.diags)
```

```
options(old)
```

---

summary.exdqlmForecast *Summary Method for* exdqlmForecast *Objects*

Description

Summary Method for `exdqlmForecast` Objects

Usage

```
## S3 method for class 'exdqlmForecast'
summary(object, ...)
```

Arguments

`object`	An `exdqlmForecast` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                 gam.init = -3.5, sig.init = 15,
                 n.samp = 20, tol = 0.2, verbose = FALSE)
M0.forecast = exdqlmForecast(start.t = 50, k = 5, m1 = M0)
summary(M0.forecast)
options(old)
```

---

summary.exdqlmForecastDiagnostic *Summary Method for* exdqlmForecastDiagnostic *Objects*

Description

Summary Method for `exdqlmForecastDiagnostic` Objects

Usage

```
## S3 method for class 'exdqlmForecastDiagnostic'
summary(object, ...)
```

Arguments

object	An `exdqlmForecastDiagnostic` object.
...	Additional arguments (unused).

---

summary.exdqlmISVB — *Summary Method for* `exdqlmISVB` *Objects*

Description

Summary Method for `exdqlmISVB` Objects

Usage

```
## S3 method for class 'exdqlmISVB'
summary(object, ...)
```

Arguments

object	An `exdqlmISVB` object.
...	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
# Legacy ISVB object retained for backward-compatible inspection methods
M0 = exdqlmISVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.IS = 20, n.samp = 20, tol = 0.2,
                   verbose = FALSE)
summary(M0)
options(old)
```

summary.exdqlmLDVB — *Summary Method for* `exdqlmLDVB` *Objects*

Description

Summary Method for `exdqlmLDVB` Objects

Usage

```
## S3 method for class 'exdqlmLDVB'
summary(object, ...)
```

Arguments

`object`	An `exdqlmLDVB` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
old = options(exdqlm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqlmLDVB(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                   gam.init = -3.5, sig.init = 15,
                   n.samp = 20, tol = 0.2, verbose = FALSE)
summary(M0)
options(old)
```

summary.exdqlmMCMC — *Summary Method for* `exdqlmMCMC` *Objects*

Description

Summary Method for `exdqlmMCMC` Objects

Usage

```
## S3 method for class 'exdqlmMCMC'
summary(object, ...)
```

Arguments

`object`	An `exdqlmMCMC` object.
`...`	Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqlm")
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M2 = exdqlmMCMC(y, p0 = 0.85, model, df = c(0.98), dim.df = c(1),
                gam.init = -3.5, sig.init = 15,
                n.burn = 20, n.mcmc = 20,
                init.from.vb = FALSE, verbose = FALSE)
summary(M2)
```

`summary.exdqlmSynthesis`

Summary Method for `exdqlmSynthesis` *Objects*

Description

Summary Method for `exdqlmSynthesis` Objects

Usage

```
## S3 method for class 'exdqlmSynthesis'
summary(object, time = NULL, ...)
```

Arguments

`object`	An `exdqlmSynthesis` object.
`time`	Optional vector of time values. If supplied, it must have length equal to the number of rows in `object$draws`.
`...`	Additional arguments (unused).

Value

A data frame containing pointwise summaries of the synthesized posterior predictive draws.

Index