

Package: exdqlm (via r-universe)

June 3, 2026

Title Extended Dynamic Quantile Linear Models

Version 0.4.0

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](https://doi.org/10.1214/21-AOAS1497)>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 3.5)

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

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

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

LinkingTo BH, Rcpp, RcppArmadillo, RcppEigen

SystemRequirements C++17; OpenMP (optional)

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

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

Config/pak/sysreqs libglpk-dev make libxml2-dev

Repository <https://antonioapdl.r-universe.dev>

Date/Publication 2026-05-03 16:25:00 UTC

RemoteUrl <https://github.com/antonioapdl/exdqlm>

RemoteRef HEAD

RemoteSha d8abe7b92690d63c31f0f5b7b998f5aa2db3b8cb

Contents

exdqlm-package	4
+exdqlm	6
as.exdqlm	6
BTflow	7
climateIndices	8
compPlot	8
dexal	9
ELIanomns	10
exal_make_mcmc_control	11
exal_make_mcmc_dqlm_sigma_control	12
exal_make_mcmc_latent_state_control	12
exal_make_mcmc_sigmagam_control	13
exal_make_mcmc_theta_control	14
exal_make_vb_control	15
exal_make_vb_sigmagam_control	16
exal_make_vb_sts_control	17
exalStaticDiagnostics	17
exalStaticLDVB	19
exalStaticMCMC	23
exdqlmDiagnostics	27
exdqlmForecast	29
exdqlmISVB	31
exdqlmLDVB	34
exdqlmMCMC	37
exdqlmPlot	41
exdqlmTransferISVB	42
exdqlmTransferLDVB	45
exdqlmTransferMCMC	49
get_gamma_bounds	54
is.exalStaticDiagnostic	54
is.exalStaticLDVB	55
is.exalStaticMCMC	55
is.exdqlm	55
is.exdqlmDiagnostic	56

is.exdqlmForecast	56
is.exdqlmISVB	56
is.exdqlmLDVB	57
is.exdqlmMCMC	57
is.exdqlmSynthesis	57
pexal	58
plot.exalStaticDiagnostic	59
plot.exalStaticLDVB	59
plot.exalStaticMCMC	60
plot.exdqlmDiagnostic	60
plot.exdqlmForecast	61
plot.exdqlmISVB	62
plot.exdqlmLDVB	62
plot.exdqlmMCMC	63
plot.exdqlmSynthesis	64
polytrendMod	65
print.exalStaticDiagnostic	66
print.exalStaticLDVB	66
print.exalStaticMCMC	67
print.exdqlm	67
print.exdqlmDiagnostic	68
print.exdqlmForecast	68
print.exdqlmISVB	69
print.exdqlmLDVB	70
print.exdqlmMCMC	70
print.exdqlmSynthesis	71
qexal	71
quantileSynthesis	72
regMod	74
rexal	75
scIVTmag	76
seasMod	77
summary.exalStaticDiagnostic	78
summary.exalStaticLDVB	78
summary.exalStaticMCMC	79
summary.exdqlm	79
summary.exdqlmDiagnostic	80
summary.exdqlmForecast	80
summary.exdqlmISVB	81
summary.exdqlmLDVB	82
summary.exdqlmMCMC	82
summary.exdqlmSynthesis	83

 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, while version 0.4.0 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()`.
- 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 in 0.4.0

- 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.

Runtime options

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

- Antonio Aguirre

Other contributors:

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

See Also

Useful links:

- <https://github.com/AntonioAPDL/exdqml>
- Report bugs at <https://github.com/AntonioAPDL/exdqml/issues>

+.exdqlm	<i>Addition for exdqlm objects</i>
----------	------------------------------------

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	<i>exdqlm objects</i>
-----------	-----------------------

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 "exdq1m" 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  
)
```

Arguments

m1	An object of class "exdq1mLDVB", "exdq1mMCMC", or legacy "exdq1mISVB".
index	Vector of consecutive integers in $\{1, \dots, q\}$ 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.

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).

Examples

```
data("scIVTmag", package = "exdq1m")
old = options(exdq1m.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 = exdq1mLDVB(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")
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](https://doi.org/10.1007/s00382019050048)

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 <code>exal_make_vb_control()</code> .
sigmagam	Optional list, usually from <code>exal_make_mcmc_sigmagam_control()</code> .
theta	Optional list, usually from <code>exal_make_mcmc_theta_control()</code> .
latent_state	Optional list, usually from <code>exal_make_mcmc_latent_state_control()</code> .
dqlm_sigma	Optional list, usually from <code>exal_make_mcmc_dqlm_sigma_control()</code> .
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

<code>freeze_burnin_iters</code>	Non-negative integer; number of burn-in iterations to hold the sigma-only block fixed.
<code>freeze_only_during_burn</code>	Logical; if TRUE, hard freeze only applies during burn-in.
<code>force_after_warmup</code>	Logical; force one immediate post-warmup update.
<code>trace</code>	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 `exdqImMCMC()`.

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$sigma`.

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 `exdq1mMCMC()`.

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 `exdq1mLDVB()`. 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

<code>max_iter</code> , <code>tol</code> , <code>n_samp_xi</code> , <code>verbose</code>	Core VB controls.
<code>sigmagam</code>	Optional list, usually from <code>exal_make_vb_sigmagam_control()</code> .
<code>sts</code>	Optional list, usually from <code>exal_make_vb_sts_control()</code> , for the dynamic latent <code>s_t</code> block in <code>exdq1mLDVB()</code> .
<code>control</code>	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$sigmatgam`.

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 `exdq1mLDVB()`.

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`.

 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.

Usage

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

Arguments

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

Details

Unlike [exdqImDiagnostics](#), 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 truth/reference errors, and compact comparison plots.

Value

An object of class "exalStaticDiagnostic" containing fitted-quantile summaries, residual summaries (when `y` is provided), optional reference-curve error metrics, 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)

```

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,
  dqIm.ind = FALSE,
  al.ind = NULL,
  n.samp = 200,
  n_samp_xi = 200,
  ld_controls = NULL,
  vb_control = NULL,
  verbose = TRUE
)

```

Arguments

<code>y</code>	Numeric vector (length n).
<code>X</code>	Numeric matrix ($n \times p$).
<code>p0</code>	Target quantile in $(0,1)$.
<code>max_iter</code>	Integer; maximum LDVB iterations (default 1000).
<code>tol</code>	Numeric; convergence tolerance based on relative ELBO changes (default $1e-4$).
<code>b0, V0</code>	Prior mean and covariance for $\beta \sim \mathcal{N}(b_0, V_0)$.
<code>beta_prior</code>	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).
<code>beta_prior_controls</code>	Optional list of prior-specific controls. For RHS-family priors (that is, when <code>beta_prior</code> is "rhs" or "rhs_ns"), supported keys include: <code>tau0</code> , <code>nu</code> , <code>s</code> or <code>s2</code> , <code>shrink_intercept</code> , <code>intercept_prec</code> , <code>n_inner</code> , <code>eta_bounds</code> , <code>freeze_tau_iters</code> , <code>freeze_tau_warmup_iters</code> , <code>update_every</code> , <code>update_every_warmup</code> , <code>update_every_warmup_iters</code> , <code>force_tau_after_warmup</code> , <code>collapse_tau_ratio_tol</code> , <code>collapse_beta_max_abs_tol</code> , <code>collapse_invV_med_tol</code> , <code>collapse_beta_l2_tol</code> , <code>collapse_small_beta_frac_tol</code> , <code>small_beta_abs_tol</code> , <code>warn_on_collapse</code> , <code>var_floor</code> , <code>h_curv</code> , <code>verbose</code> , <code>init_lambda</code> , <code>init_log_lambda</code> , <code>init_tau</code> , <code>init_log_tau</code> , <code>init_c2</code> , and <code>init_log_c2</code> . For <code>beta_prior = "rhs_ns"</code> , optional slab controls <code>a_zeta</code> , <code>b_zeta</code> , and <code>zeta2_fixed</code> 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 <code>beta_prior</code> is "rhs" or "rhs_ns", <code>b0</code> and <code>V0</code> are retained only for backward-compatible ridge behavior and are ignored for the shrunk coefficients. If both <code>init_log_tau</code> and <code>init_tau</code> are omitted (or NULL), the RHS global scale initializes at $\tau = 1$ (<code>init_log_tau = 0</code>) instead of <code>tau0</code> . By default (<code>shrink_intercept = FALSE</code>), the intercept is excluded from horseshoe shrinkage and uses <code>intercept_prec</code> .
<code>a_sigma, b_sigma</code>	Prior for $\sigma \sim IG(a_\sigma, b_\sigma)$ with density $p(\sigma) \propto \sigma^{-(a_\sigma+1)} e^{-b_\sigma/\sigma}$.
<code>gamma_bounds</code>	Two-vector (L, U) support for gamma. Defaults to $c(L, \text{fn}(p0))$, $U, \text{fn}(p0)$.
<code>log_prior_gamma</code>	Function <code>g -> log pi (gamma=g)</code> . Default is a truncated Student-t prior centered at 0 on the admissible gamma support.
<code>init</code>	Optional list with starting values: <code>beta</code> , <code>sigma</code> , <code>gamma</code> ; if missing, reasonable defaults are used.
<code>dqlm.ind</code>	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, <code>al.ind = TRUE</code> is the clearer spelling for this AL special case.
<code>al.ind</code>	Optional static-model alias for <code>dqlm.ind</code> . Prefer this argument when requesting the static AL special case. When supplied, this flag maps directly to <code>dqlm.ind</code> . If both are given, they must agree.
<code>n.samp</code>	Number of samples to draw from the approximated posterior distribution after convergence. Default is <code>n.samp = 200</code> .

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 <code>exal_make_vb_control()</code> . When supplied, the core VB arguments and warmup blocks are read from <code>vb_control</code> first and then merged with the explicit function arguments. When omitted, the package applies its conservative default warmup profile for exAL (<code>sigma</code> , <code>gamma</code>) updates while retaining the built-in RHS-family <code>tau</code> 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 $\eta = \text{logit}((\gamma - L)/(U - L))$ 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 (`dq1m.ind = TRUE`), `samp.gamma` is a vector of zeros.
- `converged`, `iter`, `run.time`, and `misc` (including $p\theta$, 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$dq1m.ind

```

exalStaticMCMC	<i>exAL (static) - MCMC algorithm</i>
----------------	---------------------------------------

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,
  dqIm.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

<code>y</code>	Numeric vector of length n .
<code>X</code>	Numeric matrix $n \times p$ (design).
<code>p0</code>	Quantile level in $(0, 1)$.
<code>b0, V0</code>	Prior mean and covariance for β (Normal). Defaults: $b_0 = \mathbf{0}_p$, $V_0 = 10^6 I_p$.
<code>beta_prior</code>	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).
<code>beta_prior_controls</code>	Optional list of prior-specific controls. For RHS-family priors (that is, when <code>beta_prior</code> is "rhs" or "rhs_ns"), supported keys include: <code>tau0</code> , <code>nu</code> , <code>s</code> or <code>s2</code> , <code>shrink_intercept</code> , <code>intercept_prec</code> , <code>n_inner</code> , <code>eta_bounds</code> , <code>var_floor</code> , <code>h_curv</code> , <code>verbose</code> , <code>init_lambda</code> , <code>init_log_lambda</code> , <code>init_tau</code> , <code>init_log_tau</code> , <code>init_c2</code> , <code>init_log_c2</code> , <code>collapse_tau_ratio_tol</code> , <code>collapse_beta_max_abs_tol</code> , <code>collapse_invV_med_tol</code> , <code>collapse_beta_l2_tol</code> , <code>collapse_small_beta_frac_tol</code> , <code>small_beta_abs_tol</code> , <code>slice_width</code> , and <code>slice_max_steps</code> . For <code>beta_prior = "rhs_ns"</code> , optional slab controls <code>a_zeta</code> , <code>b_zeta</code> , and <code>zeta2_fixed</code> 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 <code>beta_prior</code> is "rhs" or "rhs_ns", <code>b0</code> and <code>V0</code> are ignored for the shrunk coefficients and retained only for backward-compatible ridge behavior. If both <code>init_log_tau</code> and <code>init_tau</code> are omitted (or NULL), the RHS global scale initializes at <code>tau = 1</code> (<code>init_log_tau = 0</code>) instead of <code>tau0</code> . By default (<code>shrink_intercept = FALSE</code>), the intercept is excluded from horseshoe shrinkage and uses <code>intercept_prec</code> .
<code>a_sigma, b_sigma</code>	Hyperparameters for an inverse-gamma prior on <code>sigma</code> , with density proportional to $\text{sigma}^{-(a_sigma+1)} \exp(-b_sigma/\text{sigma})$.
<code>gamma_bounds</code>	Numeric length-2 vector (L, U) for <code>gamma</code> . Defaults to <code>c(L.fn(p0), U.fn(p0))</code> .
<code>log_prior_gamma</code>	Function <code>function(g) log pi(g)</code> for <code>gamma</code> on (L, U). Default is a truncated Student-t prior centered at 0 on the admissible support.
<code>init</code>	Optional list with starting values: <code>beta</code> , <code>sigma</code> , <code>gamma</code> , <code>v</code> (length n), <code>s</code> (length n), and for RHS-family priors optionally <code>lambda</code> , <code>tau</code> , and <code>c2</code> . For <code>beta_prior = "rhs_ns"</code> , the squared-scale parameterization <code>lambda2</code> , <code>tau2</code> , <code>zeta2</code> , and optional auxiliaries <code>nu</code> , <code>xi</code> are also accepted. Missing pieces are filled sensibly.
<code>dqlm.ind</code>	Logical; if TRUE, fit the reduced AL model (<code>gamma = 0</code>), corresponding to Bayesian linear quantile regression under the AL working likelihood. This removes the <code>gamma</code> - and <code>s</code> -blocks and leaves conjugate Gibbs updates for <code>beta</code> , <code>sigma</code> , and

	v. This argument is retained for consistency with the dynamic exDQLM API; in static examples, <code>al.ind = TRUE</code> is the clearer spelling for this AL special case.
<code>al.ind</code>	Optional static-model alias for <code>dqlm.ind</code> . Prefer this argument when requesting the static AL special case. When supplied, this flag maps directly to <code>dqlm.ind</code> . If both are given, they must agree.
<code>n.burn</code>	Number of burn-in iterations. Default 2000.
<code>n.mcmc</code>	Number of kept MCMC iterations (after burn). Default 1500.
<code>thin</code>	Integer; save every <code>thin</code> -th iteration after burn. We internally run <code>n.burn + n.mcmc * thin</code> iterations to return exactly <code>n.mcmc</code> saved draws.
<code>init.from.vb</code>	Logical; if TRUE, run static VB first and use its posterior moments as MCMC initialization.
<code>vb_init_controls</code>	Optional list controlling VB warm start. Supported keys: <code>max_iter</code> , <code>tol</code> , <code>n_samp_xi</code> , <code>verbose</code> , and <code>ld_controls</code> (passed through to <code>exalStaticLDVB()</code>).
<code>vb_init_fit</code>	Optional precomputed static VB fit object used as the warm-start reference when <code>init.from.vb = TRUE</code> .
<code>mcmc_control</code>	Optional normalized MCMC control list, usually from <code>exal_make_mcmc_control()</code> . When supplied, the core MCMC arguments and warmup blocks are read from <code>mcmc_control</code> first and then merged with the explicit function arguments. When omitted, the package applies its conservative default exAL (<code>sigma</code> , <code>gamma</code>) warmup profile and keeps the RHS-family tau warmup defaults active through the shared prior layer.
<code>sigmagam_controls</code>	Optional list controlling warmup/freeze behavior for the nonconjugate (<code>sigma</code> , <code>gamma</code>) block. Prefer <code>exal_make_mcmc_sigmagam_control()</code> or the <code>sigmagam</code> block of <code>exal_make_mcmc_control()</code> for new code; this argument remains available as a direct advanced override.
<code>mh.proposal</code>	Character string controlling the exAL nonconjugate update kernel. "slice" (default) uses an exact bounded univariate slice sampler on <code>gamma</code> (with <code>sigma</code> updated from its conditional), and "slice_eta" does the same on transformed <code>eta</code> . "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 <code>gamma</code> draw retained for compatibility and not recommended for routine use. Only "laplace_local" is approximate.
<code>mh.adapt</code>	Logical; adapt the random-walk proposal scale during burn-in for "rw" and "laplace_rw". Ignored for "laplace_local", "slice", and "slice_eta".
<code>mh.adapt.interval</code>	Integer adaptation window for RW-based kernels.
<code>mh.target.accept</code>	Numeric length-2 target acceptance band.
<code>mh.scale.bounds</code>	Numeric length-2 lower/upper bounds for RW proposal scale multiplier.

<code>mh.max_scale.step</code>	Numeric multiplicative adaptation cap in $(0, 1)$.
<code>mh.min_burn_adapt</code>	Integer minimum burn-in before adaptation starts.
<code>slice.width</code>	Positive numeric width for bounded slice updates when <code>mh.proposal = "slice"</code> or <code>"slice_eta"</code> .
<code>slice.max.steps</code>	Positive integer or <code>Inf</code> ; maximum stepping-out expansions for the slice sampler.
<code>gamma.substeps</code>	Positive integer; number of consecutive gamma-kernel refreshes per outer MCMC iteration. Default 1.
<code>p.global.eta.jump</code>	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.
<code>global.eta.jump.scale</code>	Positive numeric scale multiplier applied to the Laplace proposal SD used in global eta jumps.
<code>trace.diagnostics</code>	Logical; if TRUE, retain per-iteration gamma/s diagnostics under <code>mh.diagnostics\$trace</code> . Set FALSE for lighter-weight runs.
<code>trace.every</code>	Positive integer; when <code>trace.diagnostics=TRUE</code> , record one diagnostics row every <code>trace.every</code> iterations.
<code>verbose</code>	Print progress every <code>progress_every</code> iterations.
<code>progress_callback</code>	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$dqIm.ind

```

Description

The function computes the following for the model(s) provided: the posterior predictive loss criterion based off the check loss, the CRPS from posterior predictive draws, the one-step-ahead distribution sequence, and both the forward and reversed KL divergences from normality. 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

```

exdqImDiagnostics(
  m1,
  m2 = NULL,
  plot = TRUE,
  cols = c("red", "blue"),
  ref = NULL
)

```

Arguments

<code>m1</code>	An object of class "exdqImLDVB", "exdqImMCMC", or legacy "exdqImISVB".
<code>m2</code>	An optional additional object of class "exdqImLDVB", "exdqImMCMC", or legacy "exdqImISVB" to compare with <code>m1</code> .
<code>plot</code>	Logical value indicating whether the following will be plotted for <code>m1</code> and <code>m2</code> (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.
<code>cols</code>	Character vector of length 1 or 2 giving color(s) used to plot diagnostics. Default <code>c("red", "blue")</code> .
<code>ref</code>	Optional reference sample of size <code>length(m1\$y)</code> from a standard normal distribution. Used to compute the KL divergences.

Value

An object of class "exdqImDiagnostic" containing the following:

- `m1.uts` - The one-step-ahead distribution sequence of `m1`.
- `m1.KL` - The KL divergence of `m1.uts` and a standard normal.
- `m1.KL.flip` - The reversed ("flipped") KL divergence of `m1.uts` and a standard normal.
- `m1.CRPS` - The mean CRPS computed from posterior predictive draws.
- `m1.pp1c` - 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`.

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	<i>k-step-ahead quantile forecasts</i>
----------------	--

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 <code>m1</code> .
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 <code>m1</code> .
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 <code>c("purple", "magenta")</code> .
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 <code>samp.fore</code> . Default is FALSE.
n.samp	Optional positive integer specifying how many forecast draws to return when <code>return.draws = TRUE</code> . If omitted, all available posterior (σ, γ) draws from <code>m1</code> are used.
seed	Optional integer random seed used only for forecast-draw generation when <code>return.draws = TRUE</code> . If provided, the previous R RNG state is restored on exit.

Value

An object of class "exdqImForecast" 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 ($k \times n.samp$) returned when `return.draws = TRUE`.

Examples

```
# Toy example
data("scIVTmag", package = "exdqIm")
old = options(exdqIm.max_iter = 20L)
y = scIVTmag[1:100]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqImLTVB(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)

```

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, [exdqlmLTVB\(\)](#) 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.

<code>model</code>	List of the state-space model including GG, FF, prior parameters m_0 and C_0 .
<code>df</code>	Discount factors for each block.
<code>dim.df</code>	Dimension of each block of discount factors.
<code>fix.gamma</code>	Logical value indicating whether to fix gamma at <code>gam.init</code> . Default is FALSE.
<code>gam.init</code>	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if <code>fix.gamma=TRUE</code> .
<code>fix.sigma</code>	Logical value indicating whether to fix sigma at <code>sig.init</code> . Default is FALSE.
<code>sig.init</code>	Initial value for sigma (scale parameter), or value at which sigma will be fixed if <code>fix.sigma=TRUE</code> .
<code>dqlm.ind</code>	Logical value indicating whether to fix gamma at \emptyset , reducing the exDQLM to the special case of the DQLM. Default is FALSE.
<code>exps\emptyset</code>	Initial value for dynamic quantile. If <code>exps\emptyset</code> is not specified, it is set to the DLM estimate of the p_0 quantile.
<code>tol</code>	Tolerance for convergence of dynamic quantile estimates. Default is <code>tol=0.1</code> .
<code>n.IS</code>	Number of particles for the importance sampling of joint variational distribution of sigma and gamma. Default is <code>n.IS=500</code> .
<code>n.samp</code>	Number of samples to draw from the approximated posterior distribution. Default is <code>n.samp=200</code> .
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1 (or <code>sig.init</code> if provided) and variance 10.
<code>PriorGamma</code>	List of parameters for truncated student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> and degrees of freedom <code>df_gam</code> . Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
<code>verbose</code>	Logical value indicating whether progress should be displayed.
<code>debug_shapes</code>	Logical; if TRUE, print KF input/output shapes every <code>debug_every</code> iterations.
<code>debug_every</code>	Integer; frequency (in iterations) for shape prints when <code>debug_shapes=TRUE</code> .

Details

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

- `exdqml.use_cpp_kf`: use the C++ Kalman filter bridge (default TRUE).
- `exdqml.compute_elbo`: compute ELBO every iteration (default TRUE).
- `exdqml.tol_elbo`: ELBO convergence tolerance (default $1e-6$).
- `exdqml.tol_sigma`: sigma-delta convergence tolerance (default: `tol`).
- `exdqml.tol_gamma`: gamma-delta convergence tolerance (default: `tol`).
- `exdqml.vb.min_iter`: minimum iterations before convergence can trigger (default 10).
- `exdqml.vb.patience`: number of consecutive joint-converged iterations required (default 3).
- `exdqml.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.
- `exdqml.use_cpp_postpred`: use C++ posterior predictive sampler (default FALSE).

Value

An object of class "exdqImISVB" 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.
- `dqIm.ind` - Logical value indicating whether gamma was fixed at θ , reducing the exDQLM to the special case of the DQLM.
- `model` - List of the state-space model including GG, FF, prior parameters m_0 and C_0 .
- `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 `dqIm.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 `dqIm.ind=TRUE`, the object also contains:

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

Examples

```

data("scIVTmag", package = "exdqml")
old = options(exdqml.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 = exdqmlISVB(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 = exdqmlISVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                  dqml.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)

```

exdqmlLDVB

exDQLM - LDVB algorithm (Laplace-Delta)

Description

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

Arguments

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

<code>exps0</code>	Initial value for dynamic quantile. If <code>exps0</code> is not specified, it is set to the DLM estimate of the p_0 quantile.
<code>tol</code>	Tolerance for convergence of dynamic quantile estimates. Default is <code>tol=0.1</code> .
<code>n.samp</code>	Number of samples to draw from the approximated posterior distribution. Default is <code>n.samp=200</code> .
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1 (or <code>sig.init</code> if provided) and variance 10.
<code>PriorGamma</code>	List of parameters for truncated student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> and degrees of freedom <code>df_gam</code> . Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
<code>vb_control</code>	Optional normalized VB control list, usually from <code>exal_make_vb_control()</code> . When supplied, the core VB arguments and warmup blocks are read from <code>vb_control</code> first and then merged with the explicit function arguments. When omitted, exAL-style VB fits use the package's conservative default (<code>sigma</code> , <code>gamma</code>) warmup profile automatically; explicit controls remain the advanced override path.
<code>verbose</code>	Logical value indicating whether progress should be displayed.
<code>debug_shapes</code>	Logical; if TRUE, print KF input/output shapes every <code>debug_every</code> iterations.
<code>debug_every</code>	Integer; frequency (in iterations) for shape prints when <code>debug_shapes=TRUE</code> .

Details

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

- `exdqml.use_cpp_kf`: use the C++ Kalman filter bridge (default TRUE).
- `exdqml.compute_elbo`: compute ELBO every iteration (default TRUE).
- `exdqml.tol_elbo`: ELBO convergence tolerance (default 1e-6).
- `exdqml.tol_sigma`: sigma-delta convergence tolerance (default: `tol`).
- `exdqml.tol_gamma`: gamma-delta convergence tolerance (default: `tol`).
- `exdqml.vb.min_iter`: minimum iterations before convergence can trigger (default 10).
- `exdqml.vb.patience`: number of consecutive joint-converged iterations required (default 3).
- `exdqml.use_cpp_samplers`: use C++ samplers for s_t , u_t , θ (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.
- `exdqml.use_cpp_postpred`: use C++ posterior predictive sampler (default FALSE).
- `exdqml.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 "exdqmlLDVB" 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 θ , reducing the exDQLM to the special case of the DQLM.
- `model` - List of the state-space model including GG, FF, prior parameters m_0 and C_0 .
- `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 (f_m and f_C) and smoothed distribution parameters (s_m and s_C).
- `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 ($\gamma = \theta$), the LD approximation for sigma.

Examples

```

data("scIVTmag", package = "exdqIm")
old = options(exdqIm.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 = exdqImLDVB(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 = exdqImLDVB(y, p0 = 0.85, model, df = c(1,1), dim.df = c(1,4),
                  dqIm.ind = TRUE, sig.init = 15, tol = 0.2,
                  n.samp = 20, verbose = FALSE)

options(old)

```

exdqImMCMC

exDQLM - MCMC algorithm

Description

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

Arguments

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

<code>n.burn</code>	Number of MCMC iterations to burn. Default is <code>n.burn = 2000</code> .
<code>n.mcmc</code>	Number of MCMC iterations to sample. Default is <code>n.mcmc = 1500</code> .
<code>init.from.isvb</code>	Logical value indicating whether to use the legacy ISVB warm start when <code>init.from.vb = TRUE</code> . Default is <code>FALSE</code> , 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.
<code>init.from.vb</code>	Optional logical. If <code>TRUE</code> , run a VB pre-initialization step (LDVB by default, or ISVB when <code>init.from.isvb = TRUE</code>) and initialize MCMC from converged VB moments. Default is <code>TRUE</code> . If explicitly set to <code>NULL</code> , it falls back to <code>init.from.isvb</code> behavior for backward compatibility.
<code>vb_init_controls</code>	Optional list controlling VB warm start. Supported keys: <code>method</code> ("isvb" or "ldvb"), <code>tol</code> , <code>n.IS</code> , <code>n.samp</code> , <code>max_iter</code> , <code>verbose</code> .
<code>vb_init_fit</code>	Optional precomputed VB fit object. If supplied, warm start uses this object directly and does not rerun VB internally.
<code>mcmc_control</code>	Optional normalized MCMC control list, usually from <code>exal_make_mcmc_control()</code> . When supplied, the core MCMC arguments and warmup blocks are read from <code>mcmc_control</code> first and then merged with the explicit function arguments. When omitted, exAL-style dynamic MCMC uses the package's conservative default (<code>sigma</code> , <code>gamma</code>) warmup profile automatically; explicit controls remain the advanced override path.
<code>sigmagam_controls</code>	Optional list controlling warmup/freeze for the exDQLM sigma/gamma block during MCMC.
<code>latent_state_controls</code>	Optional list controlling early latent-state warmup/freeze in dynamic MCMC. Supported keys include <code>freeze_burnin_iters</code> , <code>freeze_only_during_burn</code> , <code>force_after_warmup</code> , and <code>mode</code> ("u_only" or "u_st_pair").
<code>theta_state_controls</code>	Optional list controlling early theta-state warmup/freeze in dynamic MCMC. Supported keys include <code>freeze_burnin_iters</code> , <code>freeze_only_during_burn</code> , and <code>force_after_warmup</code> .
<code>dqlm_sigma_controls</code>	Optional list controlling sigma-only warmup/freeze in the DQLM branch. Supported keys mirror <code>sigmagam_controls</code> .
<code>mh.proposal</code>	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.
<code>mh.adapt</code>	Logical; adapt MH proposal scale during burn-in.
<code>mh.adapt.interval</code>	Integer; adaptation interval (iterations).
<code>mh.target.accept</code>	Numeric length-2 vector with lower/upper target acceptance rates.

<code>mh.scale.bounds</code>	Numeric length-2 vector with min/max global scaling for MH covariance.
<code>mh.max_scale.step</code>	Numeric in (0,1); maximum fractional scale change per adaptation step.
<code>mh.min_burn_adapt</code>	Minimum burn-in iterations required to enable adaptation.
<code>slice.width</code>	Positive numeric width for the bounded slice sampler when <code>mh.proposal = "slice"</code> . Default 0.1 for parity with <code>bqrgal</code> .
<code>slice.max.steps</code>	Positive integer or <code>Inf</code> ; maximum stepping-out expansions for the slice sampler.
<code>trace.diagnostics</code>	Logical; if <code>TRUE</code> , retain per-iteration sigma/gamma/s/u diagnostics under <code>mh.diagnostics\$trace</code> . Set <code>FALSE</code> for lighter-weight runs.
<code>trace.every</code>	Positive integer; when <code>trace.diagnostics = TRUE</code> , record one diagnostics row every <code>trace.every</code> iterations.
<code>verbose.every</code>	Positive integer controlling how often console progress is printed when <code>verbose = TRUE</code> . Default 500, independent of <code>trace.every</code> .
<code>progress_callback</code>	Optional callback invoked with a named list at MCMC start, at each progress checkpoint, and on completion. Intended for workflow-level progress logging.
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1, or <code>sig.init</code> when supplied, and variance 10.
<code>PriorGamma</code>	List of parameters for truncated Student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> , and degrees of freedom <code>df_gam</code> . Default is a standard Student-t with 1 degree of freedom, truncated to the support of gamma.
<code>verbose</code>	Logical value indicating whether progress should be displayed.

Value

An object of class "exdq1mMCMC" containing the following:

- `y` - Time-series data used to fit the model.
- `run.time` - Algorithm run time in seconds.
- `dq1m.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 m_0 and C_0 .
- `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 = "exdqIm")
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 = exdqImMCMC(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 = exdqImMCMC(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)
```

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.

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.

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")
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.

<code>dim.df</code>	Dimension of each block of discount factors.
<code>lam</code>	Transfer function rate parameter lambda, a value between 0 and 1.
<code>tf.df</code>	Discount factor specification for the transfer function component. If <code>length(tf.df) = 1</code> , the value is shared by the ζ_t state and the whole ψ_t block. If <code>length(tf.df) = 2</code> , it is interpreted as <code>c(df_zeta, df_psi_shared)</code> . If <code>length(tf.df) = k + 1</code> , where $k = ncol(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \dots, \psi_{k,t})$.
<code>fix.gamma</code>	Logical value indicating whether to fix gamma at <code>gam.init</code> . Default is FALSE.
<code>gam.init</code>	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if <code>fix.gamma=TRUE</code> .
<code>fix.sigma</code>	Logical value indicating whether to fix sigma at <code>sig.init</code> . Default is FALSE.
<code>sig.init</code>	Initial value for sigma (scale parameter), or value at which sigma will be fixed if <code>fix.sigma=TRUE</code> .
<code>dqlm.ind</code>	Logical value indicating whether to fix gamma at 0, reducing the exDQLM to the special case of the DQLM. Default is FALSE.
<code>exps0</code>	Initial value for dynamic quantile. If <code>exps0</code> is not specified, it is set to the DLM estimate of the p_0 quantile.
<code>tol</code>	Tolerance for convergence of dynamic quantile estimates. Default is <code>tol=0.1</code> .
<code>n.IS</code>	Number of particles for the importance sampling of joint variational distribution of sigma and gamma. Default is <code>n.IS=500</code> .
<code>n.samp</code>	Number of samples to draw from the approximated posterior distribution. Default is <code>n.samp=200</code> .
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1 (or <code>sig.init</code> if provided) and variance 10.
<code>PriorGamma</code>	List of parameters for truncated student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> and degrees of freedom <code>df_gam</code> . Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
<code>tf.m0</code>	Prior mean of the transfer function component. Defaults to a zero vector of length $k + 1$, where $k = ncol(X)$.
<code>tf.C0</code>	Prior covariance of the transfer function component. Defaults to the $(k + 1) \times (k + 1)$ identity matrix.
<code>verbose</code>	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 "exdqImISVB" containing the following:

- `run.time` - Algorithm run time in seconds.
- `iter` - Number of iterations until convergence was reached.
- `dqIm.ind` - Logical value indicating whether gamma was fixed at θ , reducing the exDQLM to the special case of the DQLM.
- `model` - List of the augmented state-space model including GG, FF, prior parameters m_0 and C_0 .
- `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 `dqIm.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 `dqIm.ind=TRUE`, the object also contains:

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

Examples

```

data("scIVTmag", package = "exdqIm")
data("ELIanomS", package = "exdqIm")
old = options(exdqIm.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 = exdqImTransferISVB(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 = exdqImTransferISVB(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)

```

exdqImTransferLDVB *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 λ controls persistence of the accumulated transfer effect ζ_t .

Usage

```

exdqImTransferLDVB(
  y,
  p0,
  model,
  X,
  df,
  dim.df,
  lam,
  tf.df,

```

```

fix.gamma = FALSE,
gam.init = NA,
fix.sigma = FALSE,
sig.init = NA,
dqIm.ind = FALSE,
exp0,
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

<code>y</code>	A univariate time-series.
<code>p0</code>	The quantile of interest, a value between 0 and 1.
<code>model</code>	List of the state-space model including GG, FF, prior parameters m_0 and C_0 .
<code>X</code>	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.
<code>df</code>	Discount factors for each block.
<code>dim.df</code>	Dimension of each block of discount factors.
<code>lam</code>	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 .
<code>tf.df</code>	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 <code>lam</code> . If <code>length(tf.df) = 1</code> , the value is shared by the ζ_t state and the whole ψ_t block. If <code>length(tf.df) = 2</code> , it is interpreted as <code>c(df_zeta, df_psi_shared)</code> . If <code>length(tf.df) = k + 1</code> , where $k = \text{ncol}(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \dots, \psi_{k,t})$.
<code>fix.gamma</code>	Logical value indicating whether to fix gamma at <code>gam.init</code> . Default is FALSE.
<code>gam.init</code>	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if <code>fix.gamma=TRUE</code> .
<code>fix.sigma</code>	Logical value indicating whether to fix sigma at <code>sig.init</code> . Default is FALSE.
<code>sig.init</code>	Initial value for sigma (scale parameter), or value at which sigma will be fixed if <code>fix.sigma=TRUE</code> .
<code>dqIm.ind</code>	Logical value indicating whether to fix gamma at 0, reducing the exDQLM to the special case of the DQLM. Default is FALSE.

<code>exps0</code>	Initial value for dynamic quantile. If <code>exps0</code> is not specified, it is set to the DLM estimate of the p_0 quantile.
<code>tol</code>	Tolerance for convergence of dynamic quantile estimates. Default is <code>tol=0.1</code> .
<code>n.samp</code>	Number of samples to draw from the approximated posterior distribution. Default is <code>n.samp=200</code> .
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1 (or <code>sig.init</code> if provided) and variance 10.
<code>PriorGamma</code>	List of parameters for truncated student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> and degrees of freedom <code>df_gam</code> . Default is a standard student-t with 1 degree of freedom, truncated to the support of gamma.
<code>tf.m0</code>	Prior mean of the transfer function component. Defaults to a zero vector of length $k + 1$, where $k = ncol(X)$.
<code>tf.C0</code>	Prior covariance of the transfer function component. Defaults to the $(k + 1) \times (k + 1)$ identity matrix.
<code>verbose</code>	Logical value indicating whether progress should be displayed.
<code>debug_shapes</code>	Logical; if TRUE, print KF input/output shapes every <code>debug_every</code> iterations.
<code>debug_every</code>	Integer; frequency (in iterations) for shape prints when <code>debug_shapes=TRUE</code> .

Value

An object of class "exdqImLDVB" 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.
- `dqIm.ind` - Logical value indicating whether gamma was fixed at θ , 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 `dqIm.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 `dqIm.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 `exdqImLDVB()` 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 = "exdqIm")
data("ELIAnoms", package = "exdqIm")
old = options(exdqIm.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 = exdqImTransferLDVB(
  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(ELIanom[1:120], scale(scIVTmag[1:120])[, 1])
M2 = exdqImTransferLDVB(
  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)

```

exdqImTransferMCMC *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 λ . For multivariate transfer inputs, each column of X has its own instantaneous coefficient state in ψ_t , while a single scalar decay rate λ controls persistence of the accumulated transfer effect ζ_t .

Usage

```

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

<code>y</code>	A univariate time-series.
<code>p0</code>	The quantile of interest, a value between 0 and 1.
<code>model</code>	List of the state-space model including GG, FF, prior parameters m_0 and C_0 .
<code>X</code>	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.
<code>df</code>	Discount factors for each block.
<code>dim.df</code>	Dimension of each block of discount factors.
<code>lam</code>	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 .
<code>tf.df</code>	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 <code>lam</code> . If <code>length(tf.df) = 1</code> , the value is shared by the ζ_t state and the whole ψ_t block. If <code>length(tf.df) = 2</code> , it is interpreted as <code>c(df_zeta, df_psi_shared)</code> . If <code>length(tf.df) = k + 1</code> , where $k = ncol(X)$, the values are applied componentwise to $(\zeta_t, \psi_{1,t}, \dots, \psi_{k,t})$.
<code>fix.gamma</code>	Logical value indicating whether to fix gamma at <code>gam.init</code> . Default is FALSE.
<code>gam.init</code>	Initial value for gamma (skewness parameter), or value at which gamma will be fixed if <code>fix.gamma = TRUE</code> .
<code>fix.sigma</code>	Logical value indicating whether to fix sigma at <code>sig.init</code> . Default is FALSE.
<code>sig.init</code>	Initial value for sigma (scale parameter), or value at which sigma will be fixed if <code>fix.sigma = TRUE</code> .
<code>dqlm.ind</code>	Logical value indicating whether to fix gamma at 0, reducing the exDQLM to the AL/DQLM special case. Default is FALSE.

<code>Sig.mh</code>	Covariance matrix used in the random walk MH step to jointly sample sigma and gamma.
<code>joint.sample</code>	Logical value indicating whether or not to recompute <code>Sig.mh</code> based off the initial burn-in samples of gamma and sigma. Default is FALSE.
<code>n.burn</code>	Number of MCMC iterations to burn. Default is <code>n.burn = 2000</code> .
<code>n.mcmc</code>	Number of MCMC iterations to sample. Default is <code>n.mcmc = 1500</code> .
<code>init.from.isvb</code>	Logical value indicating whether to use the legacy ISVB warm start when <code>init.from.vb = TRUE</code> . 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.
<code>PriorSigma</code>	List of parameters for inverse gamma prior on sigma; shape <code>a_sig</code> and scale <code>b_sig</code> . Default is an inverse gamma with mean 1, or <code>sig.init</code> when supplied, and variance 10.
<code>PriorGamma</code>	List of parameters for truncated Student-t prior on gamma; center <code>m_gam</code> , scale <code>s_gam</code> , and degrees of freedom <code>df_gam</code> . Default is a standard Student-t with 1 degree of freedom, truncated to the support of gamma.
<code>verbose</code>	Logical value indicating whether progress should be displayed.
<code>init.from.vb</code>	Optional logical. If TRUE, run a VB pre-initialization step (LDVB by default, or ISVB when <code>init.from.isvb = TRUE</code>) and initialize MCMC from converged VB moments. Default is TRUE. If explicitly set to NULL, it falls back to <code>init.from.isvb</code> behavior for backward compatibility.
<code>vb_init_controls</code>	Optional list controlling VB warm start. Supported keys: <code>method</code> ("isvb" or "ldvb"), <code>tol</code> , <code>n.IS</code> , <code>n.samp</code> , <code>max_iter</code> , <code>verbose</code> .
<code>vb_init_fit</code>	Optional precomputed VB fit object. If supplied, warm start uses this object directly and does not rerun VB internally.
<code>mh.proposal</code>	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.
<code>mh.adapt</code>	Logical; adapt MH proposal scale during burn-in.
<code>mh.adapt.interval</code>	Integer; adaptation interval (iterations).
<code>mh.target.accept</code>	Numeric length-2 vector with lower/upper target acceptance rates.
<code>mh.scale.bounds</code>	Numeric length-2 vector with min/max global scaling for MH covariance.
<code>mh.max_scale.step</code>	Numeric in (0,1); maximum fractional scale change per adaptation step.
<code>mh.min_burn_adapt</code>	Minimum burn-in iterations required to enable adaptation.
<code>slice.width</code>	Positive numeric width for the bounded slice sampler when <code>mh.proposal = "slice"</code> . Default 0.1 for parity with <code>bqrgal</code> .

<code>slice.max.steps</code>	Positive integer or Inf; maximum stepping-out expansions for the slice sampler.
<code>trace.diagnostics</code>	Logical; if TRUE, retain per-iteration sigma/gamma/s/u diagnostics under <code>mh.diagnostics\$trace</code> . Set FALSE for lighter-weight runs.
<code>trace.every</code>	Positive integer; when <code>trace.diagnostics = TRUE</code> , record one diagnostics row every <code>trace.every</code> iterations.
<code>verbose.every</code>	Positive integer controlling how often console progress is printed when <code>verbose = TRUE</code> . Default 500, independent of <code>trace.every</code> .
<code>progress_callback</code>	Optional callback invoked with a named list at MCMC start, at each progress checkpoint, and on completion. Intended for workflow-level progress logging.
<code>tf.m0</code>	Prior mean of the transfer function component. Defaults to a zero vector of length $k + 1$, where $k = ncol(X)$.
<code>tf.C0</code>	Prior covariance of the transfer function component. Defaults to the $(k + 1) \times (k + 1)$ identity matrix.

Value

An object of class "exdqImMCMC" containing the following:

- `y` - Time-series data used to fit the model.
- `run.time` - Algorithm run time in seconds.
- `dqIm.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 `dqIm.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	<i>Bounds for the exAL shape parameter gamma</i>
------------------	--

Description

Returns valid lower/upper bounds (L, U) for the shape parameter gamma of the standardized extended Asymmetric Laplace (exAL), given p_0 in (0,1).

Usage

```
get_gamma_bounds(p0)
```

Arguments

p_0 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	<i>exalStaticDiagnostic objects</i>
-------------------------	-------------------------------------

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.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

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

Arguments

x	An exalStaticDiagnostic object.
cols	Character vector of length 1 or 2 giving color(s) used to plot diagnostics.
...	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 <code>plot</code> when <code>add = FALSE</code> .

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 <code>plot</code> when <code>add = FALSE</code> .

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 = "exdqIm")
old = options(exdqIm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqImLDVB(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 = exdqImDiagnostics(M0, plot = FALSE)
plot(M0.diags)
options(old)

```

plot.exdqImForecast *Plot Method for exdqImForecast Objects*

Description

Plot Method for exdqImForecast Objects

Usage

```

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

```

Arguments

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

Examples

```

data("scIVTmag", package = "exdqIm")
old = options(exdqIm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqImLDVB(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 = exdqImForecast(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

<code>x</code>	An <code>exdqlmSynthesis</code> object.
<code>y</code>	Optional observed series to overlay.
<code>time</code>	Optional time vector for the synthesized summaries. If omitted, <code>seq_len(T)</code> is used, where <code>T</code> is the number of synthesized time points.
<code>add</code>	Logical; add the synthesis interval to an existing plot.
<code>interval</code>	Numeric in $(0, 1)$ giving the plotted central interval. Currently <code>0.50</code> and <code>0.95</code> are supported from stored summaries.
<code>show.median</code>	Logical; draw the synthesized posterior median.
<code>show.mean</code>	Logical; draw the synthesized posterior mean.
<code>band.col</code>	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	<i>Create an n-th order polynomial exDQLM component</i>
--------------	---

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 = exdqlmLTVB(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.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

<code>p</code>	Numeric vector of probabilities in (0, 1).
<code>p0</code>	Probability level used in the quantile parametrization. Scalar in (0, 1). Default 0.5.
<code>mu</code>	Location parameter (scalar). Default 0.
<code>sigma</code>	Scale parameter (scalar, strictly positive). Default 1.
<code>gamma</code>	Skewness parameter controlling asymmetry (scalar). Must be within valid bounds implied by <code>p0</code> . Default 0.
<code>lower.tail</code>	Logical scalar; if TRUE (default) return $P(X \leq q)$, otherwise $P(X > q)$.
<code>log.p</code>	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	<i>Synthesize a unified posterior predictive distribution from multiple quantile-model draws</i>
-------------------	--

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

<code>draws_list</code>	List of length L ; each element is either: (i) a numeric matrix of posterior predictive draws ($T \times ns$ or $ns \times T$), (ii) a dynamic fit object (<code>exdq1mMCMC</code> , <code>exdq1mLDVB</code> , or legacy <code>exdq1mISVB</code>) with <code>samp.post.pred</code> , or (iii) an <code>exdq1mForecast</code> object with <code>samp.fore</code> . Rows are coerced to time.
<code>p</code>	Numeric vector of target quantile levels in $(0, 1)$ of length L (same order as <code>draws_list</code> , not necessarily sorted). Duplicate levels are not allowed.
<code>enforce_isotonic</code>	Logical; apply isotonic regression (PAVA) over the grid p at each time t to remove crossing. Default <code>TRUE</code> .
<code>rearrange</code>	Logical; apply monotone rearrangement (evaluate \rightarrow sort \rightarrow reinterpolate) on a dense grid over u in $(0, 1)$. Default <code>TRUE</code> .
<code>grid_M</code>	Integer; size of dense grid M for rearrangement ($u_k = k/(M+1)$). Default <code>1001L</code> .
<code>n_samp</code>	Integer; number of synthesized draws per time. Default <code>1000L</code> .
<code>seed</code>	<code>NULL</code> or integer for reproducible synthesized draws. Default <code>NULL</code> .
<code>T_expected</code>	Optional integer; if provided, forces the time dimension to <code>T_expected</code> when orienting each matrix to $T \times ns$. This avoids accidental transposes.

Value

An object of class "exdq1mSynthesis", which is a list containing:

- `draws` - Numeric matrix $T \times n_samp$ of synthesized draws.
- `levels` - Sorted copy of p (length L).
- `quantiles` - Numeric matrix $T \times L$ of isotone anchors $m^*_{\{i, t\}}$.
- `summary` - List with row-wise summaries of draws (mean, q_{025} , q_{250} , q_{500} , q_{750} , q_{975}).
- `method` - List of synthesis settings used (name, isotonic, rearrange, `grid_M`, `T_inferred`).

Examples

```
# short example
data("scIVTmag", package = "exdq1m")
old = options(exdq1m.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 $T \times 1$.
m_0	Optional numeric prior mean (length n). Defaults to zeros.
C_0	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).
- m_0, C_0 - 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
```

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](https://doi.org/10.1002/qj.3803)

seasMod	<i>Create Fourier representation of a periodic exDQLM component</i>
---------	---

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 "exdq1m" 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

<code>object</code>	An exalStaticMCMC object.
<code>...</code>	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

<code>object</code>	a exdqlm object.
<code>...</code>	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 exdqImForecast object.
 ... Additional arguments (unused).

Examples

```
data("scIVTmag", package = "exdqIm")
old = options(exdqIm.max_iter = 15L)
y = scIVTmag[1:60]
model = polytrendMod(1, stats::quantile(y, 0.85), 10)
M0 = exdqImLTVB(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 = exdqImForecast(start.t = 50, k = 5, m1 = M0)
summary(M0.forecast)
options(old)
```

summary.exdqImISVB *Summary Method for exdqImISVB Objects*

Description

Summary Method for exdqImISVB Objects

Usage

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

Arguments

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

Examples

```
data("scIVTmag", package = "exdqIm")
old = options(exdqIm.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 = exdqImISVB(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

- * **datasets**
 - BTflow, 7
 - climateIndices, 8
 - ELIAnoms, 10
 - scIVTmag, 76
- * **package**
 - exdqlm-package, 4
- + .exdqlm, 6
- as.exdqlm, 6
- BTflow, 7
- climateIndices, 8
- compPlot, 8
- dexal, 9
- ELIAnoms, 10
- exal_make_mcmc_control, 11
- exal_make_mcmc_control(), 25, 38
- exal_make_mcmc_dqlm_sigma_control, 12
- exal_make_mcmc_dqlm_sigma_control(), 11
- exal_make_mcmc_latent_state_control, 12
- exal_make_mcmc_latent_state_control(), 11
- exal_make_mcmc_sigmagam_control, 13
- exal_make_mcmc_sigmagam_control(), 11, 25
- exal_make_mcmc_theta_control, 14
- exal_make_mcmc_theta_control(), 11
- exal_make_vb_control, 15
- exal_make_vb_control(), 11, 21, 35
- exal_make_vb_sigmagam_control, 16
- exal_make_vb_sigmagam_control(), 15
- exal_make_vb_sts_control, 17
- exal_make_vb_sts_control(), 15
- exalStaticDiagnostics, 17
- exalStaticLDVB, 19
- exalStaticLDVB(), 4, 15, 16
- exalStaticMCMC, 23
- exalStaticMCMC(), 4, 11, 13, 16
- exdqlm (exdqlm-package), 4
- exdqlm-package, 4
- exdqlmDiagnostics, 18, 27
- exdqlmForecast, 29
- exdqlmISVB, 31
- exdqlmISVB(), 4, 29
- exdqlmLDVB, 34
- exdqlmLDVB(), 4, 15–17, 29, 31
- exdqlmMCMC, 37
- exdqlmMCMC(), 4, 11–14, 16, 29
- exdqlmPlot, 41
- exdqlmTransferISVB, 42
- exdqlmTransferISVB(), 4
- exdqlmTransferLDVB, 45
- exdqlmTransferLDVB(), 4, 42
- exdqlmTransferMCMC, 49
- exdqlmTransferMCMC(), 4
- get_gamma_bounds, 54
- is.exalStaticDiagnostic, 54
- is.exalStaticLDVB, 55
- is.exalStaticMCMC, 55
- is.exdqlm, 55
- is.exdqlmDiagnostic, 56
- is.exdqlmForecast, 56
- is.exdqlmISVB, 56
- is.exdqlmLDVB, 57
- is.exdqlmMCMC, 57
- is.exdqlmSynthesis, 57
- pexal, 58
- plot, 59, 60
- plot.exalStaticDiagnostic, 59
- plot.exalStaticLDVB, 59
- plot.exalStaticMCMC, 60
- plot.exdqlmDiagnostic, 60

plot.exdqlmForecast, 61
plot.exdqlmISVB, 62
plot.exdqlmLDVB, 62
plot.exdqlmMCMC, 63
plot.exdqlmSynthesis, 64
polytrendMod, 65
polytrendMod(), 4
print.exalStaticDiagnostic, 66
print.exalStaticLDVB, 66
print.exalStaticMCMC, 67
print.exdqlm, 67
print.exdqlmDiagnostic, 68
print.exdqlmForecast, 68
print.exdqlmISVB, 69
print.exdqlmLDVB, 70
print.exdqlmMCMC, 70
print.exdqlmSynthesis, 71

qexal, 71
quantileSynthesis, 57, 64, 72
quantileSynthesis(), 4

regMod, 74
regMod(), 4
rexal, 75

scIVTmag, 76
seasMod, 77
seasMod(), 4
summary.exalStaticDiagnostic, 78
summary.exalStaticLDVB, 78
summary.exalStaticMCMC, 79
summary.exdqlm, 79
summary.exdqlmDiagnostic, 80
summary.exdqlmForecast, 80
summary.exdqlmISVB, 81
summary.exdqlmLDVB, 82
summary.exdqlmMCMC, 82
summary.exdqlmSynthesis, 83