Package: proxymix (via r-universe)

June 2, 2026

Type Package

Title KL-Optimal Gaussian Mixture Proxies for Arbitrary Target Densities

Version 0.3.0

Description Fits multivariate Gaussian-mixture proxies that are Kullback-Leibler optimal to user-supplied target densities on real Euclidean space. Three fitting regimes are unified under one verb: (i) closed-form moment matching for a single component, (ii) classical expectation-maximisation when independent samples are available, and (iii) importance-sampled KLD-EM when the target can be evaluated point-wise but not (cheaply) sampled. Closed-form Gaussian-mixture operators (density, sampling, marginalisation, conditioning, divergence) round out the toolkit. Implements the regime hierarchy of Hoek and Elliott (2024) <doi:10.1080/07362994.2024.2372605>.

License MIT + file LICENSE

URL https://github.com/max578/proxymix

BugReports https://github.com/max578/proxymix/issues

Encoding UTF-8

Language en-AU

Depends R (>= 4.3)

Imports S7, stats, mvnfast, cli, rlang

Suggests mclust, ggplot2, viridis, patchwork, withr, knitr, rmarkdown, testthat (>= 3.0.0), vdiffr, spelling, covr

VignetteBuilder knitr

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/testthat/parallel true

Repository https://max578.r-universe.dev

Date/Publication 2026-06-02 06:03:30 UTC

RemoteUrl https://github.com/max578/proxymix

RemoteRef main

RemoteSha 51d879b0d12f318671ddaa530d2b5cd1a20bcd97

Contents

`autoplot.gmm_fit` *Plot a fitted Gaussian-mixture proxy*

Description

An `autoplot()` method for gmm_fit objects, rendering the fitted mixture with `ggplot2`. The displayed coordinates are reduced to the requested one or two dimensions through the closed-form marginal `gmm_marginalise()`, so the method works for a proxy of any ambient dimension `p`.

A one-dimensional request draws the marginal mixture density, optionally with the per-component densities underneath and a rug of the target's samples. A two-dimensional request draws the marginal density as a viridis raster with white contour lines, optionally overlaying each component's mean and a probability-contour ellipse.

Arguments

`object`	A gmm_fit, typically from `fit_proxymix()`.
`dims`	Integer vector of length one or two giving the coordinate(s) to display, in `1:p`. Defaults to the first two coordinates (or the only coordinate when `p == 1`).
`n_grid`	Integer scalar — the number of grid points per axis at which the density is evaluated. A two-dimensional plot evaluates `n_grid^2` points.
`n_sd`	Numeric scalar — how many component standard deviations beyond the extreme component means the plotting window extends.
`level`	Numeric scalar in `(0, 1)` — the probability level of the per-component ellipse drawn on a two-dimensional plot.
`show_components`	Logical scalar — whether to overlay the per-component densities (one dimension) or mean-and-ellipse glyphs (two dimensions).
`show_data`	Logical scalar — whether to overlay the target's samples, when the fitted target carries any.
`...`	Currently ignored, present for generic compatibility.

Details

The method is registered against the `ggplot2::autoplot()` generic only when `ggplot2` is installed; call it as `ggplot2::autoplot(fit)` or load `ggplot2` first. It returns the `ggplot` object, so the usual `+` layering applies for further customisation.

Value

A `ggplot` object.

See Also

Other classes: gmm(), gmm_dim(), gmm_fit(), gmm_n_components(), gmm_target(), is_proposal()

Examples

```
samples <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(samples)
fit <- fit_proxymix(tgt, N = 2L, regime = "sample", max_iter = 25L)
ggplot2::autoplot(fit)
ggplot2::autoplot(fit, dims = 1L)
```

---

banana_target *Banana-shaped 2-D target*

Description

A 2-D "banana" density obtained by warping an isotropic Gaussian through the map $(z_1, z_2) \mapsto (z_1, z_2 + \frac{1}{2}(z_1^2 - 1))$. The map has unit Jacobian, so the resulting density is exactly normalised.

Usage

```
banana_target(with_samples = FALSE, n = 2000L, seed = 1L)
```

Arguments

with_samples	If TRUE, attach n exact samples drawn by the change-of-variables trick. Default FALSE — the target then exposes only its log_density, which is the regime-(iii) use case.
n	Number of samples to attach when with_samples = TRUE.
seed	Optional integer seed used when drawing the samples.

Value

A gmm_target in dimension 2.

See Also

Other targets: donut_target(), gmm_target_from_samples(), mixture_target()

Examples

```
b <- banana_target()
b
b@log_density(matrix(c(0, 0, 1, 0), ncol = 2, byrow = TRUE))
```

bic_aic *Bayesian and Akaike information criteria*

Description

Returns the BIC and AIC of a regime-(ii) fit. Both criteria are computed against the *empirical* log-likelihood of the samples used to fit the model. They are `NA` for regimes that do not have an empirical likelihood (`"moment"`, `"kld"`).

Usage

```
bic_aic(fit)
```

Arguments

`fit`	A gmm_fit.

Value

A list with `bic`, `aic`, and `n_params`.

See Also

Other diagnostics: `ess_summary()`, `ess_trace()`, `hellinger_mc()`, `kld_trace()`

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(x)
fit <- fit_proxymix(tgt, N = 2L, regime = "sample", max_iter = 25L)
bic_aic(fit)
```

dgmm *Density of a Gaussian mixture*

Description

Evaluates the density (or log-density) of a Gaussian mixture at one or more points.

Usage

```
dgmm(x, g, log = FALSE)
```

Arguments

x	A numeric matrix with one observation per row, or a length-p numeric vector (treated as a single observation).
g	A gmm (or gmm_fit) object.
log	Logical. If TRUE, return log-densities.

Value

A numeric vector of length nrow(x).

See Also

Other ops: gmm_canonicalise(), gmm_conditionalise(), gmm_kld(), gmm_marginalise(), rgmm()

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
dgmm(c(0, 0), g)
dgmm(c(0, 0), g, log = TRUE)
```

---

donut_target *Donut-shaped 2-D target*

Description

A rotationally symmetric annulus on $\mathbb{R}^2$, with density

$$f(x) \propto \exp\left(-\frac{(\|x\|-r_0)^2}{2\sigma^2}\right).$$

Numerical integration in polar coordinates fixes the normaliser; the returned target exposes a normalised log_density.

Usage

```
donut_target(r0 = 2.5, sigma = 0.5, with_samples = FALSE, n = 2000L, seed = 1L)
```

Arguments

r0	Centre radius of the annulus.
sigma	Annulus width.
with_samples	If TRUE, attach n exact samples via polar change-of-variables and a one-dimensional rejection step.
n	Number of samples to attach when with_samples = TRUE.
seed	Optional integer seed used when drawing the samples.

Value

A gmm_target in dimension 2.

See Also

Other targets: banana_target(), gmm_target_from_samples(), mixture_target()

Examples

```
d <- donut_target()
d
```

`ess_summary` *Summary of importance-sampling diagnostics*

Description

Convenience accessor returning the headline IS-quality numbers for a regime-(iii) fit: effective sample size and its ratio to `is_size`, the largest self-normalised weight, the support fraction (proportion of draws that received a finite weight), and the Monte-Carlo standard error of the final KLD estimate. Returns `NA` fields for regimes that do not use importance sampling.

Usage

```
ess_summary(fit)
```

Arguments

`fit`	A gmm_fit.

Details

Validation-side numbers (`validation_*`) are populated only when the fit was called with `validation_size > 0`.

Value

A list of numeric scalars (or `NA`s where not applicable).

See Also

Other diagnostics: bic_aic(), ess_trace(), hellinger_mc(), kld_trace()

Examples

```
fit <- fit_proxymix(banana_target(), N = 3L, regime = "kld",
                    is_size = 1500L, max_iter = 20L, seed = 1L,
                    validation_size = 1500L)
ess_summary(fit)
```

ess_trace *Effective sample size of the importance-sampling weights*

Description

Returns the effective sample size (`1 / sum(W^2)`) of the self-normalised importance weights used by a regime-(iii) fit. `NA` for regimes that do not use importance sampling.

Usage

```
ess_trace(fit)
```

Arguments

`fit`	A gmm_fit.

Value

Numeric scalar (or `NA_real_`).

See Also

Other diagnostics: `bic_aic()`, `ess_summary()`, `hellinger_mc()`, `kld_trace()`

Examples

```
fit <- fit_proxymix(banana_target(), N = 2L, regime = "kld",
                    is_size = 1000L, max_iter = 15L, seed = 1L)
ess_trace(fit)
```

fit_em_samples *Classical EM fit on samples*

Description

Implements regime (ii) of Hoek and Elliott (2024). Runs the textbook expectation-maximisation algorithm for Gaussian mixtures on the supplied samples, with diagonal ridge regularisation for numerical stability, optional multi-start, and monotone-log-likelihood checking.

Usage

```
fit_em_samples(
  target,
  N = 2L,
  init = NULL,
  max_iter = 100L,
  tol = 1e-06,
  ridge_eps = 1e-06,
  n_starts = 5L,
  canonicalise = TRUE
)
```

Arguments

target	A gmm_target carrying an n by p samples matrix.
N	Number of mixture components.
init	A gmm initialisation, or NULL to use init_kmeans().
max_iter	Maximum number of EM iterations.
tol	Relative-log-likelihood convergence tolerance.
ridge_eps	Ridge added to each component covariance at every M-step.
n_starts	Number of multi-start initialisations (only when init is NULL). The best fit by final log-likelihood is returned.
canonicalise	Logical. If TRUE (the default), the fitted mixture is post-processed by gmm_canonicalise() so that components are sorted by descending weight and (as a tiebreaker) by descending ||mu||.

Value

A gmm_fit with regime = "sample".

See Also

Other fitting: fit_kld_em(), fit_moment_match(), from_kde()

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(x)
fit <- fit_em_samples(tgt, N = 2L, max_iter = 30L, n_starts = 2L)
fit@diagnostics$loglik_final
```

`fit_kld_em` *Importance-sampled KLD-EM fit (the wedge)*

Description

Implements regime (iii) of Hoek and Elliott (2024). Minimises `KL(f || g_theta)` where `f` is supplied as an evaluable log-density on the target, via expectation-maximisation against importance-sampled draws from a user-chosen proposal `q`.

Usage

```
fit_kld_em(
  target,
  N = 3L,
  proposal = NULL,
  is_size = 5000L,
  init = NULL,
  max_iter = 100L,
  tol = 1e-05,
  ridge_eps = 1e-06,
  min_ess = 50,
  seed = NULL,
  validation_size = 0L,
  validation_proposal = NULL,
  validation_seed = NULL,
  support_warn = TRUE,
  canonicalise = TRUE
)
```

Arguments

`target`	A gmm_target with a non-NULL `log_density`.
`N`	Number of mixture components.
`proposal`	An is_proposal. Defaults to a multivariate-t with `df = 5` in `target@n_dim` dimensions.
`is_size`	Number of importance-sampling draws used for fitting.
`init`	A gmm initialisation, or `NULL` to use a kmeans pass on the importance-resampled draws.
`max_iter`	Maximum number of EM iterations.
`tol`	Convergence tolerance on the relative change in the importance-sampled KLD estimate.
`ridge_eps`	Ridge added to each component covariance at every M-step.
`min_ess`	Minimum effective sample size below which a warning is issued.
`seed`	Optional integer seed used when drawing the fitting IS sample.

`validation_size`
: Number of independent importance-sampling draws to use for held-out validation. Default `0L` (no validation split). Set to a positive integer (typically `is_size / 2` to `is_size`) to enable validation diagnostics.

`validation_proposal`
: Optional is_proposal for the validation sample. Defaults to the same proposal used for fitting.

`validation_seed`
: Optional integer seed used when drawing the validation sample. Defaults to `seed + 1L` when `seed` is supplied, `NULL` otherwise.

`support_warn`
: Logical. If `TRUE` (the default), issue a warning when more than 5% of IS draws receive non-finite weights (typically because the proposal does not dominate the target's support).

`canonicalise`
: Logical. If `TRUE` (the default), the fitted mixture is post-processed by `gmm_canonicalise()`.

Details

The Monte Carlo draws from `q` are computed once at the start; the resulting self-normalised importance-sampling weights are reused at every EM iteration. Adaptive importance sampling — redrawing each round — is a Tier-3 deferral.

Since v0.1.1 the function also draws an *independent* validation IS sample when `validation_size > 0` and reports its own KLD estimate, effective sample size, and largest weight share. This lets users tell the difference between in-sample EM overfit to one particular IS draw and a fit that generalises across independent IS draws.

When the target's `normalised` property is `FALSE` or `NA`, the importance-sampled `kld_final` and `kld_trace` measure $\widehat{KL}(f\|g) - \log Z(f)$ rather than the absolute divergence. The fit's diagnostics list records this via `kld_is_shifted = TRUE` and a `kld_shift_explanation` string. When the target also supplies a finite `log_normalizer`, a corrected absolute estimate is reported as `kld_final_absolute`.

Value

A gmm_fit with `regime = "kld"`. The diagnostics list contains, among others, `kld_trace`, `kld_final`, `kld_is_shifted`, `kld_final_absolute` (when computable), `ess`, `ess_relative` (`ess / is_size`), `max_weight`, `support_fraction`, `mc_se_kld`, `validation_kld`, `validation_ess`, and `validation_max_weight`.

See Also

Other fitting: `fit_em_samples()`, `fit_moment_match()`, `from_kde()`

Examples

```
tgt <- banana_target()
q <- is_mvt(n_dim = 2L, mean = c(0, 0),
            sigma = 4 * diag(2), df = 5)
fit <- fit_kld_em(tgt, N = 3L, proposal = q,
                  is_size = 1500L, max_iter = 25L, seed = 1L,
                  validation_size = 1500L)
```

```
fit@diagnostics$kld_final
fit@diagnostics$validation_kld
```

`fit_kld_em_collider` *KLD-EM under collider-induced conditional-independence constraints (stub)*

Description

Tier-2 stub — signature stable; body not yet implemented.

Usage

```
fit_kld_em_collider(target, dag, N = 3L, ...)
```

Arguments

`target`	A gmm_target with a non-NULL `log_density`.
`dag`	A description of the DAG structure (planned: an adjacency matrix or a `dagitty` object).
`N`	Number of mixture components.
`...`	Future configuration arguments.

Details

Plan: a regime-(iii) KLD-EM that projects each iteration onto the manifold of joint densities respecting a DAG-implied set of conditional-independence constraints. Targets the collider-regularised regression direction described by Sejdinovic et al.

Value

A gmm_fit (when implemented).

See Also

Other stubs: `from_aggregate_likelihood()`, `from_simulator()`, `to_apsim_scenarios()`

Examples

```
try(fit_kld_em_collider(banana_target(), dag = matrix(0, 2, 2), N = 2L))
```

`fit_moment_match` *Closed-form moment-matching fit*

Description

Implements regime (i) of Hoek and Elliott (2024). When `N == 1`, this is the exact moment match: `mu` is the target mean and `Sigma` is the target covariance. When `N > 1`, the function returns the deterministic moment-seed of `init_moment_seed()` wrapped as a gmm_fit, without iterative refinement — useful as a starting point for the iterative regimes.

Usage

```
fit_moment_match(target, N = 1L, ridge_eps = 1e-06, canonicalise = TRUE)
```

Arguments

`target`	A gmm_target.
`N`	Number of components. `N >= 2` returns a moment-seeded mixture without iterative refinement.
`ridge_eps`	Ridge added to the empirical covariance for numerical stability.
`canonicalise`	Logical. If `TRUE` (the default), the fitted mixture is post-processed by `gmm_canonicalise()` so that components are sorted by descending weight and (as a tiebreaker) by descending `||mu||`. Set `FALSE` to retain the raw component order.

Details

Either the `target` must carry an `n` by `p` `samples` matrix, or its `metadata` slot must contain precomputed `moments` of the form `list(mean = <p-vec>, cov = <p-by-p>)`.

Value

A gmm_fit with `regime = "moment"`.

See Also

Other fitting: `fit_em_samples()`, `fit_kld_em()`, `from_kde()`

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(x)
fit_moment_match(tgt, N = 1L)
```

`fit_proxymix` *Fit a Gaussian-mixture proxy to a target density*

Description

The unified front door of `proxymix`. Picks a fitting regime (or honours an explicit choice) and dispatches to the corresponding regime-specific fitter:

Usage

```
fit_proxymix(
  target,
  N = 1L,
  regime = c("auto", "moment", "sample", "kld"),
  ...
)
```

Arguments

`target`	A gmm_target.
`N`	Number of components.
`regime`	One of `"auto"`, `"moment"`, `"sample"`, `"kld"`.
`...`	Additional arguments forwarded to the regime-specific fitter. The most useful pass-throughs are `canonicalise` (whether to apply `gmm_canonicalise()` to the returned fit; default `TRUE`), `validation_size` and `validation_proposal` (held-out IS validation for regime `"kld"`), `max_iter`, `tol`, `n_starts`, and `seed`.

Details

- `"moment"` - closed-form moment matching (`fit_moment_match()`).
- `"sample"` - classical EM on i.i.d. samples (`fit_em_samples()`).
- `"kld"` - importance-sampled KLD-EM (`fit_kld_em()`) — the wedge.

With `regime = "auto"` the choice is made from the shape of the supplied target:

- `N == 1` and the target carries samples or moments: `"moment"`.
- `N >= 2` and the target carries samples: `"sample"`.
- The target carries `log_density` only (no samples): `"kld"`.

Value

A gmm_fit.

Examples

```
## auto: samples + N=2 -> classical EM.
x <- matrix(stats::rnorm(200), ncol = 2)
tgt_s <- gmm_target_from_samples(x)
fit_proxymix(tgt_s, N = 2L, max_iter = 25L)

## explicit "kld" on a log-density-only target.
fit_proxymix(banana_target(), N = 3L, regime = "kld",
             is_size = 1000L, max_iter = 20L, seed = 1L)
```

`from_aggregate_likelihood` *From an aggregate likelihood to a Gaussian-mixture proxy (stub)*

Description

Tier-2 stub — signature stable; body not yet implemented.

Usage

```
from_aggregate_likelihood(y, latent_aggregator, N = 3L, ...)
```

Arguments

`y`	A numeric matrix or vector of aggregate-level observations.
`latent_aggregator`	A function mapping latent `x` to aggregate space.
`N`	Number of Gaussian components in the proxy.
`...`	Future configuration arguments.

Details

Plan: fit a Gaussian-mixture proxy `f_theta(x)` so that, when used as the latent-level density inside an aggregate-likelihood downscaling framework `g(y) = E_{x | y}[f(x)]`, the resulting downscaling likelihood has tractable closed-form marginalisation. Targets the kernel-downsizing application described by Sejdinovic et al.

Value

A gmm_fit (when implemented).

See Also

Other stubs: `fit_kld_em_collider()`, `from_simulator()`, `to_apsim_scenarios()`

Examples

```
try(from_aggregate_likelihood(matrix(0, 1, 1),
                              latent_aggregator = identity,
                              N = 2L))
```

`from_kde` *Compile a kernel-density estimate into a Gaussian-mixture proxy*

Description

Fits an `N`-component Gaussian-mixture proxy to a (Gaussian, diagonal- bandwidth) kernel-density estimate over `samples`, via regime (iii) KLD-EM. The proxy is closed-form marginalisable, conditionable, and samplable; the KDE is none of those things on its own.

Usage

```
from_kde(
  samples,
  N = 3L,
  bandwidth = "silverman",
  proposal = NULL,
  is_size = 5000L,
  max_iter = 100L,
  tol = 1e-05,
  ridge_eps = 1e-06,
  min_ess = 50L,
  seed = NULL,
  validation_size = 0L,
  validation_proposal = NULL,
  validation_seed = NULL,
  support_warn = TRUE,
  canonicalise = TRUE
)
```

Arguments

`samples`	An `n` by `p` numeric matrix of points. `n >= 5`, `p <= 10`.
`N`	Number of mixture components in the proxy.
`bandwidth`	Either `"silverman"`, `"scott"`, a positive numeric scalar (absolute bandwidth applied to every coordinate), or a length-`p` positive numeric vector of per-coordinate absolute bandwidths. Default `"silverman"`.
`proposal`	Optional is_proposal. Default is a multivariate-t centred at `colMeans(samples)`, scale = ridge(cov(samples)) + diag(h^2), `df = 5`.
`is_size`	Importance-sample size for fitting. Default `5000L`.
`max_iter`	Maximum EM iterations. Forwarded to `fit_kld_em()`.

tol	Convergence tolerance. Forwarded to fit_kld_em().
ridge_eps	Ridge added to each component covariance at every M-step. Forwarded to fit_kld_em().
min_ess	Minimum effective sample size below which a warning is issued. Forwarded to fit_kld_em().
seed	Optional integer seed for the fitting IS draw.
validation_size	Held-out IS sample size. Forwarded to fit_kld_em().
validation_proposal	Optional is_proposal for the held-out sample. Forwarded to fit_kld_em().
validation_seed	Optional integer seed for the held-out IS draw. Forwarded to fit_kld_em().
support_warn	Logical. Forwarded to fit_kld_em().
canonicalise	Logical. If TRUE, the fitted mixture is post-processed by gmm_canonicalise(). Forwarded to fit_kld_em().

Details

This is a **compression** operation: take an n-sample KDE and replace it with the closest N-component mixture in the Kullback-Leibler sense (which is much smaller than n for typical use). Bias inherited from the KDE is reproduced in the proxy; the bandwidth controls the bias-variance trade-off.

Dimensional scope. The graduation guard is p <= 5 (recommended), p <= 10 (allowed with warning), p > 10 (rejected). The wedge of KLD-EM is regime (iii) IS, whose effective-sample-size collapses sharply in high dimensions; richer ambient spaces will await the v0.3 affine-Gaussian operator calculus.

Value

A gmm_fit with regime = "kld" and metadata recording the KDE inputs (kde_samples_n, bandwidth, bandwidth_method).

See Also

Other fitting: fit_em_samples(), fit_kld_em(), fit_moment_match()

Other v0_2: gmm_target_from_posterior()

Examples

```
set.seed(1L)
x <- rbind(
  mvnfast::rmvn(120L, mu = c(-2, 0), sigma = diag(2)),
  mvnfast::rmvn(120L, mu = c( 2, 0), sigma = diag(2))
)
fit <- from_kde(x, N = 2L, is_size = 2000L, max_iter = 40L, seed = 1L)
fit
ess_summary(fit)
```

`from_simulator` *Wrap an expensive simulator as a* `gmm_target` *(stub)*

Description

Tier-2 stub — signature stable; body not yet implemented.

Usage

```
from_simulator(simulator, design, ...)
```

Arguments

`simulator`	A function `function(x)` mapping inputs to outputs.
`design`	An `n` by `p` matrix of input design points.
`...`	Future configuration arguments.

Details

Plan: probe an expensive simulator over a Latin-hypercube design, build a kernel-density estimate (or empirical-likelihood surface) on the simulator outputs, and expose the result as a gmm_target with an evaluable `log_density`.

Value

A gmm_target (when implemented).

See Also

Other stubs: `fit_kld_em_collider()`, `from_aggregate_likelihood()`, `to_apsim_scenarios()`

Examples

```
try(from_simulator(simulator = identity,
                   design = matrix(stats::rnorm(20), ncol = 2)))
```

gmm *A Gaussian mixture*

Description

Lightweight S7 class representing an K-component multivariate Gaussian mixture on $\mathbb{R}^p$. Use `gmm()` to construct, `dgmm()` / `rgmm()` to evaluate or sample, and `gmm_marginalise()` / `gmm_conditionalise()` for closed-form operations.

Usage

```
gmm(
  weights = numeric(0),
  means = list(),
  covariances = list(),
  name = "gmm",
  metadata = list()
)
```

Arguments

`weights`	Numeric vector of length K, non-negative, summing to one.
`means`	List of length K, each element a length-p numeric vector.
`covariances`	List of length K, each element a p-by-p symmetric positive-definite numeric matrix.
`name`	Optional human-readable name.
`metadata`	Optional list of arbitrary metadata (regime tags, diagnostic snapshots, etc.).

Value

An S7 object inheriting from `gmm`.

See Also

Other classes: `autoplot.gmm_fit`, `gmm_dim()`, `gmm_fit()`, `gmm_n_components()`, `gmm_target()`, `is_proposal()`

Examples

```
g <- gmm(
  weights = c(0.4, 0.6),
  means = list(c(-1, 0), c(1, 0)),
  covariances = list(diag(2), diag(2))
)
g
```

gmm_affine *Affine pushforward of a Gaussian mixture*

Description

Returns the (closed-form) distribution of $Y = AX + b + \epsilon$ when $X \sim g$ is a Gaussian mixture and $\epsilon \sim \mathcal{N}(0, R)$ is independent additive Gaussian noise.

Usage

```
gmm_affine(g, A, b = 0, noise_cov = NULL, ridge_eps = 1e-06)
```

Arguments

g	A gmm (or gmm_fit) in R^p.
A	An m by p numeric matrix.
b	Numeric scalar or length-m vector. Default 0.
noise_cov	m by m SPD numeric matrix, or NULL (treated as the zero matrix — a deterministic channel).
ridge_eps	Tiny ridge added to the output covariances for numerical hygiene. Set to 0 to disable.

Details

For each component k, the pushed-forward parameters are

$$\mu'_k = A\mu_k + b, \qquad \Sigma'_k = A\Sigma_k A^\top + R,$$

and the mixture weights are unchanged. This is the finite-mixture analogue of a Kalman-style predict step.

The channel is required to be **affine** in x and the noise is required to be Gaussian. Non-linear channels are not closed form and must not be silently approximated; pass them to a Monte Carlo helper instead.

Value

A gmm in R^m with the same number of components and the same weights as g.

See Also

Other operators: gmm_aggregate(), gmm_missing(), gmm_observe()

Other v0_3: gmm_aggregate(), gmm_missing(), gmm_observe()

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
A <- matrix(c(1, 0, 0, 1, 1, -1), nrow = 3L, byrow = TRUE)
gmm_affine(g, A, b = c(0, 0, 0), noise_cov = 0.01 * diag(3))
```

`gmm_aggregate` *Aggregation pushforward of a Gaussian mixture*

Description

A named alias for `gmm_affine()` when `A` is a (row-wise) aggregation matrix — e.g. a block-sum, block-average, or unequal-weight aggregation used in downscaling pipelines. The mathematics is identical to `gmm_affine()`; the alias gives the public API a clearer hook for aggregation-specific diagnostics in later releases.

Usage

```
gmm_aggregate(g, A, noise_cov = NULL, ridge_eps = 1e-06)
```

Arguments

`g`	A gmm (or gmm_fit) in `R^p`.
`A`	An `m` by `p` numeric matrix.
`noise_cov`	Optional `m` by `m` SPD numeric matrix. Default `NULL` (deterministic aggregation).
`ridge_eps`	Tiny ridge added to the output covariances for numerical hygiene.

Value

A gmm in `R^m`.

See Also

Other operators: `gmm_affine()`, `gmm_missing()`, `gmm_observe()`

Other v0_3: `gmm_affine()`, `gmm_missing()`, `gmm_observe()`

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0, 1), c(1, 0, -1)),
         covariances = list(diag(3), diag(3)))
# Sum coordinates 1 and 2 into a single aggregate; pass coord 3 through.
A <- matrix(c(1, 1, 0,
              0, 0, 1), nrow = 2L, byrow = TRUE)
gmm_aggregate(g, A)
```

gmm_canonicalise *Canonicalise the component ordering of a Gaussian mixture*

Description

Returns a new gmm (or gmm_fit) with the components permuted into a canonical order: weight descending, then `||mu||` descending as a tiebreaker. The mixture distribution is unchanged — only the bookkeeping order is — but the canonical ordering removes the EM label-switching nuisance from snapshot tests, cross-run comparisons, and printed summaries.

Usage

```
gmm_canonicalise(g)
```

Arguments

g	A gmm (or gmm_fit) object.

Details

Applied automatically by the regime-specific fitters (`fit_moment_match()`, `fit_em_samples()`, `fit_kld_em()`) and by the top-level dispatcher `fit_proxymix()` when `canonicalise = TRUE` (the default).

Value

A gmm (or gmm_fit) of the same subclass as g, with the components permuted into canonical order.

See Also

Other ops: `dgmm()`, `gmm_conditionalise()`, `gmm_kld()`, `gmm_marginalise()`, `rgmm()`

Examples

```
g <- gmm(weights = c(0.1, 0.6, 0.3),
         means = list(c(0, 0), c(3, 0), c(-1, 1)),
         covariances = list(diag(2), diag(2), diag(2)))
gmm_canonicalise(g)
```

gmm_conditionalise *Conditional of a Gaussian mixture*

Description

Computes the conditional distribution of a Gaussian mixture given fixed values of a subset of coordinates, by the Schur-complement formula applied component-wise and re-weighted by the marginal evidence $p(x_b)$ of each component.

Usage

```
gmm_conditionalise(g, given)
```

Arguments

g	A gmm (or gmm_fit) object.
given	A length-p numeric vector. Coordinates to *condition on* take their numeric value; coordinates left *free* are NA.

Value

A gmm object in dimension equal to the number of free coordinates.

See Also

Other ops: dgmm(), gmm_canonicalise(), gmm_kld(), gmm_marginalise(), rgmm()

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
gmm_conditionalise(g, given = c(NA, 0.5))
```

gmm_dim *Dimension of a Gaussian mixture*

Description

Convenience accessor returning the ambient dimension p.

Usage

```
gmm_dim(x)
```

Arguments

x	A gmm (or gmm_fit) object.

Value

Integer scalar.

See Also

Other classes: autoplot.gmm_fit, gmm(), gmm_fit(), gmm_n_components(), gmm_target(), is_proposal()

Examples

```
g <- gmm(weights = 1, means = list(c(0, 0)),
         covariances = list(diag(2)))
gmm_dim(g)
```

---

gmm_fit *A fitted Gaussian-mixture proxy*

Description

A gmm_fit is the result of fit_proxymix() (or one of the regime-specific fitters). It inherits the mixture parameters of gmm and adds a record of the target it was fitted to, the regime used, and the iteration diagnostics.

Usage

```
gmm_fit(
  weights = numeric(0),
  means = list(),
  covariances = list(),
  name = "gmm",
  metadata = list(),
  target = NULL,
  regime = NA_character_,
  diagnostics = list(),
  converged = NA,
  iterations = NA_integer_,
  call = NULL
)
```

Arguments

weights, means, covariances, name, metadata	See gmm.
target	The gmm_target the mixture was fitted to.
regime	One of "moment", "sample", "kld".
diagnostics	A list of regime-specific diagnostics (see kld_trace(), ess_trace()).
converged	Logical scalar.
iterations	Integer scalar.
call	The matched call.

Value

An S7 object inheriting from gmm_fit (and gmm).

See Also

Other classes: autoplot.gmm_fit, gmm(), gmm_dim(), gmm_n_components(), gmm_target(), is_proposal()

Examples

```
samples <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(samples)
fit <- fit_proxymix(tgt, N = 2L, regime = "sample", max_iter = 25L)
inherits(fit, "proxymix::gmm_fit")
```

---

gmm_kld *Kullback-Leibler divergence between two Gaussian mixtures*

Description

Estimates KL(p || q) between two Gaussian mixtures by Monte Carlo and, optionally, evaluates the Hershey–Olsen variational approximation as a deterministic sanity check.

Usage

```
gmm_kld(p, q, n_mc = 5000L, variational = TRUE)
```

Arguments

p, q	Two gmm (or gmm_fit) objects of the same ambient dimension.
n_mc	Number of Monte Carlo samples drawn from p.
variational	If TRUE, also return the Hershey–Olsen variational approximation.

Details

The Monte Carlo estimator draws `n_mc` samples from `p` and returns the empirical mean of `log p(x) - log q(x)`, together with a Monte Carlo standard error.

The variational approximation is

$$\widehat{D}_{\mathrm{var}}(p\|q) = \sum_a \pi_a \log\left(\frac{\sum_{a'} \pi_{a'}\, e^{-\mathrm{KL}(p_a\|p_{a'})}}{\sum_b \omega_b\, e^{-\mathrm{KL}(p_a\|q_b)}}\right),$$

which is exact when `p == q` and tends to be a usable lower bound when the components of `p` and `q` are well-separated. The closed-form Gaussian–Gaussian KL `KL(p_a || q_b)` is used internally.

Value

A list with components

- `mc` - the Monte Carlo estimate of `KL(p || q)`,
- `mc_se` - its Monte Carlo standard error,
- `variational` - the variational approximation (`NA` if `variational = FALSE`),
- `n_mc` - the number of Monte Carlo samples used.

See Also

Other ops: `dgmm()`, `gmm_canonicalise()`, `gmm_conditionalise()`, `gmm_marginalise()`, `rgmm()`

Examples

```
p <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
q <- gmm(weights = 1,
         means = list(c(0, 0)),
         covariances = list(diag(2) * 2))
gmm_kld(p, q, n_mc = 500L)
```

---

gmm_marginalise *Marginal of a Gaussian mixture*

Description

Computes the marginal distribution of a Gaussian mixture over a subset of coordinates. The marginal of a Gaussian mixture is itself a Gaussian mixture with the same weights.

Usage

```
gmm_marginalise(g, keep)
```

Arguments

g	A gmm (or gmm_fit) object.
keep	Integer vector of coordinate indices to retain (in 1..p).

Value

A gmm object in dimension length(keep).

See Also

Other ops: dgmm(), gmm_canonicalise(), gmm_conditionalise(), gmm_kld(), rgmm()

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0, 2), c(1, 0, -2)),
         covariances = list(diag(3), diag(3)))
gmm_marginalise(g, keep = c(1L, 3L))
```

---

gmm_missing *Condition a Gaussian mixture on the exact values of some coordinates*

Description

A structured wrapper around gmm_conditionalise() for the common case where the observed coordinates are specified by integer index rather than NA-padded vector. Equivalent to gmm_observe() with a selection matrix A and zero noise covariance, but routed through the Schur-complement path for efficiency.

Usage

```
gmm_missing(g, observed, values)
```

Arguments

g	A gmm (or gmm_fit) in R^p.
observed	Integer vector of indices in seq_len(p). The coordinates to condition on (fully observed).
values	Numeric vector of length length(observed). The observed values, in the same order as observed.

Value

A gmm in R^(p - length(observed)).

See Also

Other operators: gmm_affine(), gmm_aggregate(), gmm_observe()

Other v0_3: gmm_affine(), gmm_aggregate(), gmm_observe()

Examples

```
g <- gmm(weights = c(0.4, 0.6),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
# Condition coord 2 on the value 0.5; keep coord 1.
gmm_missing(g, observed = 2L, values = 0.5)
```

---

gmm_n_components *Number of components in a Gaussian mixture*

Description

Number of components in a Gaussian mixture

Usage

```
gmm_n_components(x)
```

Arguments

x	A gmm (or gmm_fit) object.

Value

Integer scalar.

See Also

Other classes: autoplot.gmm_fit, gmm(), gmm_dim(), gmm_fit(), gmm_target(), is_proposal()

Examples

```
g <- gmm(weights = c(0.5, 0.5), means = list(c(0, 0), c(1, 1)),
         covariances = list(diag(2), diag(2)))
gmm_n_components(g)
```

`gmm_observe` *Bayesian update of a Gaussian mixture on a noisy linear observation*

Description

Conditions a Gaussian mixture `g` on a single noisy linear observation $y = Ax + b + \epsilon$, $\epsilon \sim \mathcal{N}(0, R)$. Per component, applies the Kalman gain

$$K_k = \Sigma_k A^\top S_k^{-1}, \qquad S_k = A\Sigma_k A^\top + R,$$

and updates

$$\mu_k' = \mu_k + K_k(y - A\mu_k - b), \qquad \Sigma_k' = (I - K_k A)\Sigma_k.$$

Component weights are multiplied by the marginal evidence $\pi_k \mathcal{N}(y; A\mu_k + b, S_k)$ and renormalised. This is the finite-mixture analogue of a Kalman update step.

Usage

```
gmm_observe(g, A, y, noise_cov, b = 0, ridge_eps = 1e-06)
```

Arguments

`g`	A gmm (or gmm_fit) in `R^p`.
`A`	An `m` by `p` numeric matrix.
`y`	A length-`m` numeric vector (the observation).
`noise_cov`	An `m` by `m` SPD numeric matrix (the observation noise covariance `R`). Required.
`b`	Numeric scalar or length-`m` vector. Default `0`.
`ridge_eps`	Tiny ridge added to updated covariances for numerical hygiene. Set to `0` to disable.

Details

If the marginal evidence vanishes at every component (e.g. `y` is many standard deviations from every component), the function issues a warning and returns `g` unchanged with `metadata$gmm_observe_no_update = TRUE`.

Value

A gmm in `R^p` with the same number of components and the reweighted component weights.

See Also

Other operators: `gmm_affine()`, `gmm_aggregate()`, `gmm_missing()`

Other v0_3: `gmm_affine()`, `gmm_aggregate()`, `gmm_missing()`

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
A <- matrix(c(1, 0), nrow = 1L)
gmm_observe(g, A = A, y = 0.8, noise_cov = matrix(0.25, 1, 1))
```

---

gmm_target *A target density on R^p*

Description

An S7 representation of a target density that proxymix is asked to approximate. A target may carry an evaluable log_density, a matrix of i.i.d. samples, or both. Each of the three fitting regimes consumes a different subset:

Usage

```
gmm_target(
  n_dim = integer(0),
  log_density = NULL,
  samples = NULL,
  normalised = NA,
  log_normalizer = NA_real_,
  name = "gmm_target",
  metadata = list()
)
```

Arguments

n_dim	Integer scalar — the ambient dimension p (the property is called n_dim rather than dim because S7 reserves dim as an attribute name).
log_density	Optional function: function(x) taking a numeric matrix n by p and returning a length-n numeric vector of log f(x).
samples	Optional n by p numeric matrix of i.i.d. samples from the target.
normalised	Logical scalar declaring whether log_density integrates to one. TRUE, FALSE, or NA (unknown). Defaults to NA. Downstream diagnostics treat NA and FALSE identically and label any KLD estimate as shifted.
log_normalizer	Numeric scalar log Z(f) of the supplied log_density, if known. Default NA_real_. When normalised = FALSE and log_normalizer is finite, downstream diagnostics can correct shifted KLD estimates by + log_normalizer.
name	Human-readable name.
metadata	Optional list of additional descriptors.

Details

- regime "moment" needs samples (or both moments via metadata);
- regime "sample" needs samples;
- regime "kld" needs the log-density.

Use gmm_target() or gmm_target_from_samples() to construct.

Importance-sampled KLD-EM (regime "kld") only requires log_density to be specified up to an unknown additive constant — the self-normalised weights are invariant to scaling. The package's *diagnostics* downstream, however, do depend on normalisation: an importance-sampled KLD estimate against an unnormalised log-density measures $\widehat{KL}(f\|g) - \log Z(f)$ rather than $\widehat{KL}(f\|g)$, and a squared-Hellinger Monte Carlo estimate is only meaningful when both densities integrate to one. Declare the target's normalisation explicitly via normalised (and, where possible, supply log_normalizer) so that the package can label shifted KLDs as shifted and refuse misleading Hellinger reports.

Value

An S7 object of class gmm_target.

See Also

Other classes: autoplot.gmm_fit, gmm(), gmm_dim(), gmm_fit(), gmm_n_components(), is_proposal()

Examples

```
tgt <- banana_target()
tgt
```

---

gmm_target_from_posterior

Compile an unnormalised Bayesian posterior into a gmm_target

Description

Generic S3 constructor that turns a Bayesian posterior — represented either by a fitted model object (e.g. from flexyBayes, brms, Stan) or by a bare callable — into a gmm_target suitable for the regime (iii) wedge of fit_proxymix() / fit_kld_em().

Usage

```
gmm_target_from_posterior(model, ...)

## Default S3 method:
gmm_target_from_posterior(model, ...)

## S3 method for class '`function`'
gmm_target_from_posterior(
```

```
  model,
  ...,
  parameter_names = NULL,
  log_normalizer = NA_real_,
  name = NULL
)
```

Arguments

model	One of: • a function — a bare callable satisfying the contract above; • a fitted model object whose class registers a method (e.g. flexybayes::gmm_target_from_posteri
...	Forwarded to method-specific implementations.
parameter_names	Character vector of parameter names. Required for the function method (or attached as attr(model, "parameter_names")). The length determines n_dim.
log_normalizer	Numeric scalar log Z of the posterior, if known. NA_real_ (the default) otherwise; downstream diagnostics will label any KLD estimate as shifted.
name	Optional human-readable target name. Defaults to "posterior".

Details

The contract for the underlying callable is:

- **Vectorised**: accepts a numeric matrix with rows indexing independent parameter draws and columns indexing parameters; returns a length-nrow(theta) numeric vector of log p(theta | data) + const.
- **Unnormalised is fine**: the marginal likelihood log Z is not required. Where the source package can supply it, pass log_normalizer.
- **Side-effect free**: no plotting, no mutable state. Pure function.
- **Domain-safe**: returns -Inf outside support rather than raising an error.

The default method errors with a hint pointing the user at either (a) a Bayesian package that registers a method, or (b) the function method below.

Value

A gmm_target with normalised = FALSE and the user-supplied log_normalizer (or NA_real_).

See Also

Other v0_2: from_kde()

Examples

```
# A trivial unnormalised log-posterior: a 2D banana centred near (1, 0).
log_post <- function(theta) {
  x <- theta[, 1L]
  y <- theta[, 2L]
```

```
  -0.5 * (x^2 + (y - 0.1 * x^2 + 1)^2)
}
tgt <- gmm_target_from_posterior(
  log_post,
  parameter_names = c("x", "y")
)
tgt
```

`gmm_target_from_samples` *Build a target from samples alone*

Description

Wraps a numeric matrix of i.i.d. samples as a gmm_target. The resulting target carries no `log_density`, so it can only feed regimes `"moment"` (via empirical moments) and `"sample"` (classical EM).

Usage

```
gmm_target_from_samples(samples, name = "target_from_samples")
```

Arguments

`samples`	An n by p numeric matrix.
`name`	Optional human-readable name. Defaults to `"target_from_samples"`.

Value

A gmm_target object.

See Also

Other targets: `banana_target()`, `donut_target()`, `mixture_target()`

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(x)
tgt
```

`hellinger_mc` *Monte-Carlo Hellinger distance between a fit and its target*

Description

Estimates the squared Hellinger distance `H^2(f, g) = 1 - integral sqrt(f(x) g(x)) dx` by importance sampling against the proposal stored in the fit (for regime `"kld"`) or by sampling from the fit itself (for regime `"sample"`). The target's `log_density` must be supplied **and normalised**; otherwise the Monte Carlo integral is biased by the missing $\sqrt{Z(f)}$. When the target's `normalised` property is not `TRUE`, a warning is issued and the returned value is flagged.

Usage

```
hellinger_mc(fit, n_mc = 5000L, seed = NULL)
```

Arguments

`fit`	A gmm_fit whose target carries a `log_density`.
`n_mc`	Number of Monte Carlo samples.
`seed`	Optional integer seed.

Value

A list with components

- `h2` - estimate of `H^2(f, g)`,
- `se` - Monte Carlo standard error,
- `n_mc` - sample size used.

See Also

Other diagnostics: `bic_aic()`, `ess_summary()`, `ess_trace()`, `kld_trace()`

Examples

```
fit <- fit_proxymix(banana_target(), N = 3L, regime = "kld",
                    is_size = 2000L, max_iter = 25L, seed = 1L)
hellinger_mc(fit, n_mc = 1000L, seed = 1L)
```

init_kmeans *k-means initialisation*

Description

Runs `stats::kmeans()` on the supplied samples and uses the resulting cluster centres and within-cluster covariances to seed an EM-style fitter.

Usage

```
init_kmeans(samples, N = 2L, ridge_eps = 1e-06, nstart = 10L)
```

Arguments

`samples`	An `n` by `p` numeric matrix of samples from (or close to) the target.
`N`	Number of components.
`ridge_eps`	Ridge added to each cluster covariance for numerical stability when a cluster has fewer than two points.
`nstart`	`stats::kmeans` `nstart` argument.

Value

A gmm of `N` components in dimension `ncol(samples)`.

See Also

Other init: `init_moment_seed()`, `init_random()`, `init_warm_start()`, `multi_start_best_of()`

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
init_kmeans(x, N = 3L)
```

init_moment_seed *Moment-seed initialisation*

Description

Computes the global mean and covariance of the supplied samples and spreads `N` components along the leading principal direction. Useful as a deterministic starting point that survives multi-modal targets better than a single-Gaussian fit.

Usage

```
init_moment_seed(samples, N = 2L, spread = 1.5)
```

Arguments

samples	An n by p numeric matrix.
N	Number of components.
spread	Multiplier on the principal-direction standard deviation used to place the component means symmetrically about the global mean.

Value

A gmm of N components in dimension ncol(samples).

See Also

Other init: init_kmeans(), init_random(), init_warm_start(), multi_start_best_of()

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
init_moment_seed(x, N = 3L)
```

---

init_random *Random initialisation*

Description

Generates an N-component random initialisation by perturbing isotropic means around a centre. Useful as one starting point in a multi-start best-of strategy.

Usage

```
init_random(
  N = 1L,
  p = 2L,
  centre = rep(0, p),
  scale = 1,
  sigma_diag = 1,
  seed = NULL
)
```

Arguments

N	Number of components.
p	Ambient dimension.
centre	Length-p numeric vector — the location around which means are drawn.
scale	Standard deviation of the mean perturbation.
sigma_diag	Diagonal value used for the initial component covariances.
seed	Optional integer seed.

Value

A gmm of N components in dimension p.

See Also

Other init: init_kmeans(), init_moment_seed(), init_warm_start(), multi_start_best_of()

Examples

```
init_random(N = 3L, p = 2L, seed = 1L)
```

---

init_warm_start *Warm-start initialisation from an existing fit*

Description

Returns the input as-is. Provided as a name so that the multi-start driver can include "warm starts" by symbolic name.

Usage

```
init_warm_start(g)
```

Arguments

g	A gmm (or gmm_fit).

Value

The input g, validated.

See Also

Other init: init_kmeans(), init_moment_seed(), init_random(), multi_start_best_of()

Examples

```
g <- gmm(weights = 1, means = list(c(0, 0)),
         covariances = list(diag(2)))
init_warm_start(g)
```

is_mvn *Multivariate-normal proposal*

Description

Builds an is_proposal using a multivariate-normal `N(mean, cov)` density and sampler.

Usage

```
is_mvn(n_dim, mean = rep(0, n_dim), cov = diag(n_dim))
```

Arguments

`n_dim`	Ambient dimension p.
`mean`	Length-p numeric mean vector. Defaults to the zero vector.
`cov`	A p-by-p symmetric positive-definite covariance matrix. Defaults to the identity.

Value

An is_proposal object.

See Also

Other proposals: `is_mvt()`, `is_uniform()`

Examples

```
q <- is_mvn(n_dim = 2L, mean = c(0, 0), cov = 4 * diag(2))
q
```

is_mvt *Multivariate-t proposal*

Description

Builds an is_proposal using a multivariate-Student-t density and sampler with `df` degrees of freedom, location `mean`, and scale matrix `sigma`. Heavier tails than `is_mvn()`, so often a safer importance proposal at moderate dimensions.

Usage

```
is_mvt(n_dim, mean = rep(0, n_dim), sigma = diag(n_dim), df = 5)
```

Arguments

n_dim	Ambient dimension p.
mean	Length-p numeric location vector. Defaults to the zero vector.
sigma	A p-by-p symmetric positive-definite scale matrix. Defaults to the identity.
df	Degrees of freedom (df > 2 recommended for finite variance).

Value

An is_proposal object.

See Also

Other proposals: is_mvn(), is_uniform()

Examples

```
q <- is_mvt(n_dim = 2L, df = 5)
q
```

---

is_proposal *An importance-sampling proposal*

Description

An is_proposal packages a sampler with its corresponding log-density. Pass one to fit_kld_em() (or fit_proxymix() with regime = "kld") to plug an alternative proposal into the regime-(iii) loop.

Usage

```
is_proposal(
  n_dim = integer(0),
  sample = NULL,
  log_density = NULL,
  name = "is_proposal",
  metadata = list()
)
```

Arguments

n_dim	Integer scalar — the ambient dimension p (the property is called n_dim rather than dim because S7 reserves dim as an attribute name).
sample	Function: function(n) returning an n by p numeric matrix of independent draws.
log_density	Function: function(x) taking a numeric matrix n by p and returning a length-n numeric vector of log q(x).
name	Human-readable name.
metadata	Optional list of additional descriptors.

Value

An S7 object of class `is_proposal`.

See Also

Other classes: `autoplot.gmm_fit`, `gmm()`, `gmm_dim()`, `gmm_fit()`, `gmm_n_components()`, `gmm_target()`

Examples

```
q <- is_mvn(n_dim = 2L, mean = c(0, 0), cov = diag(2))
q
```

---

is_uniform *Uniform-on-a-box proposal*

Description

Builds an is_proposal that samples uniformly on the hyperrectangle $[\text{lower}_1, \text{upper}_1] \times \cdots \times [\text{lower}_p, \text{upper}_p]$ and reports the (constant) log-density on that box. Outside the box the log-density is `-Inf`.

Usage

```
is_uniform(n_dim, lower = -1, upper = 1)
```

Arguments

`n_dim`	Ambient dimension p.
`lower`	Length-p numeric vector of lower bounds (recycled from a single value).
`upper`	Length-p numeric vector of upper bounds (recycled from a single value).

Value

An is_proposal object.

See Also

Other proposals: `is_mvn()`, `is_mvt()`

Examples

```
q <- is_uniform(n_dim = 2L, lower = -5, upper = 5)
q
q@sample(3L)
```

kld_trace *Per-iteration KLD trace of a fit*

Description

Returns the per-iteration estimate of `KL(f || g_theta)` produced during a regime-(iii) fit, or `NA` for regimes that do not estimate the KLD internally.

Usage

```
kld_trace(fit)
```

Arguments

`fit`	A gmm_fit.

Value

Numeric vector (or `NA_real_`).

See Also

Other diagnostics: `bic_aic()`, `ess_summary()`, `ess_trace()`, `hellinger_mc()`

Examples

```
fit <- fit_proxymix(banana_target(), N = 2L, regime = "kld",
                    is_size = 1000L, max_iter = 15L, seed = 1L)
kld_trace(fit)
```

mixture_target *Three-component Gaussian-mixture target*

Description

A toy three-component planar mixture target where everything is known exactly: the `log_density` matches the GMM density formula and the attached `samples` are drawn from that same mixture. Useful for sanity- checking the three fitting regimes against ground truth.

Usage

```
mixture_target(with_samples = FALSE, n = 2000L, seed = 1L)
```

Arguments

with_samples	If TRUE, attach n exact mixture draws.
n	Number of samples to attach when with_samples = TRUE.
seed	Optional integer seed used when drawing the samples.

Value

A gmm_target in dimension 2.

See Also

Other targets: banana_target(), donut_target(), gmm_target_from_samples()

Examples

```
m <- mixture_target(with_samples = TRUE, n = 100L)
m
```

---

multi_start_best_of *Multi-start best-of wrapper*

Description

Runs the supplied fitter from each of several initialisations and returns the fit with the best score, following Karlis and Xekalaki (2003)'s recommendation.

Usage

```
multi_start_best_of(fit_fn, inits, score_fn, ...)
```

Arguments

fit_fn	A function with signature function(init, ...) returning a gmm_fit.
inits	A list of gmm initialisations.
score_fn	A function function(fit) returning a numeric score — *larger is better* (typically the final log-target evidence).
...	Additional arguments forwarded to fit_fn.

Value

The gmm_fit with the largest score_fn(fit).

See Also

Other init: init_kmeans(), init_moment_seed(), init_random(), init_warm_start()

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
tgt <- gmm_target_from_samples(x)
inits <- list(init_random(2L, 2L, seed = 1L),
              init_moment_seed(x, N = 2L))
best <- multi_start_best_of(
  fit_fn   = function(init, ...) fit_em_samples(tgt, init = init, ...),
  inits    = inits,
  score_fn = function(fit) fit@diagnostics$loglik_final,
  max_iter = 25L
)
best@diagnostics$loglik_final
```

---

rgmm *Sample from a Gaussian mixture*

Description

Draws n independent samples from a Gaussian mixture.

Usage

```
rgmm(n, g)
```

Arguments

n	Number of samples (positive integer scalar).
g	A gmm (or gmm_fit) object.

Value

A numeric matrix of dimension n by p.

See Also

Other ops: dgmm(), gmm_canonicalise(), gmm_conditionalise(), gmm_kld(), gmm_marginalise()

Examples

```
g <- gmm(weights = c(0.5, 0.5),
         means = list(c(-1, 0), c(1, 0)),
         covariances = list(diag(2), diag(2)))
x <- rgmm(50L, g)
dim(x)
```

`to_apsim_scenarios` *Gaussian-mixture samples to APSIM scenario tables (stub)*

Description

Tier-2 stub — signature stable; body not yet implemented.

Usage

```
to_apsim_scenarios(fit, n = 100L, schema = list(), ...)
```

Arguments

`fit`	A gmm_fit.
`n`	Number of scenarios to generate.
`schema`	A list mapping mixture coordinates to APSIM variable names and units.
`...`	Future configuration arguments.

Details

Plan: convert samples from a fitted gmm_fit into the tabular format consumed by an APSIM scenario runner, optionally honouring a user-supplied schema mapping mixture coordinates to APSIM variable names and units.

Value

A data frame in APSIM scenario format (when implemented).

See Also

Other stubs: `fit_kld_em_collider()`, `from_aggregate_likelihood()`, `from_simulator()`

Examples

```
x <- matrix(stats::rnorm(200), ncol = 2)
fit <- fit_proxymix(gmm_target_from_samples(x), N = 2L, max_iter = 10L)
try(to_apsim_scenarios(fit, n = 100L, schema = list()))
```

Index