Package 'rareflow'

May 9, 2026

Title Variational Flow-Based Inference for Rare Events and Large Deviations

Version 0.1.0

Description Variational flow-based methods for modeling rare events using Kullback–Leibler (KL) divergence, normalizing flows, Girsanov change of measure, and Freidlin–Wentzell action functionals. The package provides tools for rare-event inference, minimum-action paths, and quasi-potential computation in stochastic dynamical systems. Methods are based on Rezende and Mohamed (2015) <doi:10.48550/arXiv.1505.05770>, Girsanov (1960) <doi:10.1137/1105027>, and Freidlin and Wentzell (2012, ISBN:978-0387955477).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Suggests knitr, rmarkdown, ggplot2, gganimate, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Pietro Piu [aut, cre]

Maintainer Pietro Piu <pietro.piu.si@gmail.com>

Repository CRAN

Date/Publication 2026-02-13 16:40:08 UTC

Contents

`ELBOflow` *Evidence Lower Bound (ELBO) for Flow-Based Variational Inference*

Description

Computes the ELBO:

Usage

```
ELBOflow(flow, Qobs, pxgivenz, nmc = 256)
```

Arguments

`flow`	A flow model created by `makeflow()`.
`Qobs`	Observed empirical distribution (probability vector).
`pxgivenz`	A function mapping a latent vector z to a categorical pmf.
`nmc`	Number of Monte Carlo samples.

Details

$\log p(x\|z) + \log p(z) - \log q(z)$

where:

- q(z) is the flow-based variational posterior
- p(z) is a standard Gaussian prior
- p(x | z) is provided by the user as `pxgivenz`

The expectation is approximated via Monte Carlo sampling.

Value

A numeric ELBO value.

Examples

```
f <- makeflow("planar", list(u = 0.1, w = 0.2, b = 0))
px <- function(z) c(0.3, 0.4, 0.3)
ELBOflow(f, Qobs = c(0.2, 0.5, 0.3), pxgivenz = px, nmc = 100)
```

`fitflowvariational` *Fit a Flow-Based Variational Posterior for Sanov Inference*

Description

This function fits a variational posterior q(z) using a chosen normalizing flow. The objective is the ELBO:

Usage

```
fitflowvariational(
  observed,
  states = NULL,
  flowtype = c("maf", "splinepwlin", "planar", "radial"),
  flowspec = list(),
  inittheta = NULL,
  pxgivenz,
  nmc = 256,
  control = list()
)
```

Arguments

`observed`	Observed empirical distribution Q (probability vector).
`states`	Optional vector of category names.
`flowtype`	One of "maf", "splinepwlin", "planar", "radial".
`flowspec`	A list specifying structural parameters (d, K, etc.).
`inittheta`	Optional initial parameter vector for trainable flows.
`pxgivenz`	A function mapping latent z to a categorical pmf.
`nmc`	Number of Monte Carlo samples for ELBO estimation.
`control`	List of control parameters passed to `optim()`.

Details

$\log p(x|z) + \log p(z) - \log q(z)$

The flow parameters (theta) are optimized via `optim()` when applicable (MAF and spline flows). Planar and radial flows have no trainable parameters.

This function performs generic variational inference using a chosen normalizing flow and a user-provided likelihood `pxgivenz`.

For specialized rare-event inference using:

- Girsanov change of measure
- Freidlin–Wentzell quasi-potential

see the wrapper functions:

- `fitflow_girsanov()`
- `fitflow_FW()`

These wrappers construct a tilted likelihood and then call `fitflowvariational()` internally.

Value

A list containing:

- flow: the fitted flow model
- elbo: final ELBO value
- theta: optimized parameter vector (if applicable)
- convergence: optim() convergence code

`fitflow_FW` *Fit Flow Using Freidlin-Wentzell Quasi-Potential*

Description

Computes the Freidlin-Wentzell quasi-potential between x_0 and x_1, constructs a tilted likelihood proportional to $\exp(-V/\mathrm{eps})$, and fits a flow-based variational posterior.

Usage

```
fitflow_FW(
  observed,
  states = NULL,
  flowtype = "maf",
  flowspec = list(),
  inittheta = NULL,
  drift,
  x0,
  x1,
  T = 200,
  dt = 0.01,
  eps = 0.1,
  nmc = 256,
  control = list()
)
```

Arguments

observed	Empirical distribution Q.
states	Optional category names.
flowtype	Flow type.
flowspec	Structural parameters for the flow.
inittheta	Optional initial theta.
drift	Drift function $b(x)$.
x0	Starting point.
x1	Target point.
T	Number of time steps.
dt	Time step.
eps	Noise strength (small parameter).
nmc	Monte Carlo samples.
control	Control list for `optim()`.

Details

This is useful for rare-event inference in small-noise diffusions, where the quasi-potential acts as an effective energy landscape.

Value

Output of `fitflowvariational()`.

`fitflow_girsanov` *Fit Flow with Girsanov-Tilted Likelihood*

Description

Applies a Girsanov change of measure to tilt the likelihood and then fits a flow-based variational posterior using `fitflowvariational()`.

Usage

```
fitflow_girsanov(
  observed,
  states = NULL,
  flowtype = "maf",
  flowspec = list(),
  inittheta = NULL,
  base_pxgivenz,
  theta_path,
  Winc,
```

```
  dt,
  nmc = 256,
  control = list()
)
```

Arguments

observed	Empirical distribution Q (probability vector).
states	Optional category names.
flowtype	Flow type ("maf", "splinepwlin", "planar", "radial").
flowspec	Structural parameters for the flow.
inittheta	Optional initial theta for trainable flows.
base_pxgivenz	Likelihood $p(x \mid z)$ before tilting.
theta_path	Drift-tilting function or vector for Girsanov.
Winc	Brownian increments.
dt	Time step.
nmc	Monte Carlo samples.
control	Control list for `optim()`.

Details

This is useful when the target distribution arises from a drift-tilted diffusion process, where the Radon-Nikodym derivative is given by the Girsanov theorem.

Value

Output of `fitflowvariational()`.

`Freidlin_Wentzell_action`

Freidlin-Wentzell Action Functional

Description

Computes the discrete Freidlin-Wentzell action for a path $\phi(t)$ represented as a matrix of size $T \times d$. The continuous action is:

Usage

```
Freidlin_Wentzell_action(phi, drift, dt)
```

Arguments

phi	Matrix of path values of dimension $T \times d$.
drift	Drift function $b(x)$ returning a numeric vector.
dt	Time step.

Details

$$I[\phi] = \frac{1}{2}\int_0^T \|\dot{\phi}(t) - b(\phi(t))\|^2 \, dt$$

and the discrete approximation is:

$$I \approx \frac{1}{2}\sum_{t=1}^{T-1} \|(\phi_{t+1} - \phi_t)/dt - b(\phi_t)\|^2 \, dt$$

Value

Numeric action value.

`FW_quasipotential` *Freidlin-Wentzell Quasi-Potential via Path Minimization*

Description

Computes an approximate Freidlin-Wentzell quasi-potential between two points x_0 and x_1 by minimizing the FW action functional over discretized paths.

Usage

```
FW_quasipotential(
  x0,
  x1,
  drift,
  T = 200,
  dt = 0.01,
  niter = 200,
  stepsize = 0.1
)
```

Arguments

`x0`	Starting point (numeric vector).
`x1`	Target point (numeric vector).
`drift`	Drift function $b(x)$.
`T`	Number of time steps.
`dt`	Time step.
`niter`	Number of gradient descent iterations.
`stepsize`	Gradient descent step size.

Details

The algorithm:

1. Initializes a straight-line path between x_0 and x_1.
2. Performs simple gradient descent on the FW action.

This is a naive but effective illustrative method for low-dimensional systems. More advanced solvers (string method, MAM, etc.) can be plugged in.

Value

A list with:

- path: matrix of size $T \times d$
- action: FW action of the optimized path

`girsanov_logratio` *Girsanov Log-Ratio for Drift-Tilted Diffusions*

Description

Computes the Radon-Nikodym derivative (log form) associated with a Girsanov change of measure for an SDE:

Usage

```
girsanov_logratio(theta_path, Winc, dt)
```

Arguments

`theta_path`	Numeric vector of drift tilts θ_t.
`Winc`	Numeric vector of Brownian increments ΔW_t.
`dt`	Time step size.

Details

$dX_t = b(X_t)\,dt + dW_t$

tilted by an alternative drift:

$dX_t = (b(X_t) + \theta_t)\,dt + dW_t$.

The log-likelihood ratio is:

$$\log \frac{dQ}{dP} = \sum_t \left(\theta_t W_t - \frac{1}{2}\theta_t^2\,dt \right)$$

This function returns the log-ratio for a given path of drift tilts `theta_path`, Brownian increments `Winc`, and time step `dt`.

Value

A numeric log-likelihood ratio.

`KLdiv` *Kullback–Leibler Divergence*

Description

Computes the KL divergence D(Q ‖ P) between two discrete probability distributions Q and P.

Usage

```
KLdiv(Q, P)
```

Arguments

`Q`	Observed distribution (probability vector).
`P`	Baseline or reference distribution (probability vector).

Value

A numeric KL divergence value.

`mafflowmodel` *Masked Autoregressive Flow (MAF)*

Description

A readable and structured implementation of a Masked Autoregressive Flow. This version supports:

- arbitrary dimension d
- K sequential flow steps
- a single parameter vector theta containing all weights

Usage

```
mafflowmodel(d = 3, K = 2, theta = NULL)
```

Arguments

`d`	Dimension of the latent space.
`K`	Number of flow steps.
`theta`	Optional parameter vector. If NULL, random initialization.

Value

A flow model object with methods:

- sampleq(n)
- logq(z0)
- applyflow(z0)

---

makebiolik *Biological Two-Separator Likelihood*

Description

Same structure as `makeneurolik()`, but with a different default separation parameter. This can be used to model simple biological switching systems or coarse-grained gene expression states.

Usage

```
makebiolik(a = 0.2)
```

Arguments

`a`	Separation parameter controlling the spacing between the two logits.

Value

A function mapping a latent vector z to a probability vector.

---

makeflow *Flow Factory*

Description

A unified constructor for all flow models in the rareflow package. This function dispatches to the appropriate flow implementation based on the `flowtype` argument.

Usage

```
makeflow(flowtype, params = list())
```

Arguments

`flowtype`	Character string specifying the flow type.
`params`	A named list of parameters required by the chosen flow.

Details

Supported flow types:

- "planar" -> planarflowmodel()
- "radial" -> radialflowmodel()
- "maf" -> mafflowmodel()
- "splinepwlin" -> splinepwlinflowmodel()

Value

A flow model object with methods:

- sampleq(n)
- logq(z0)
- applyflow(z0)

Examples

```
# Create a planar flow
f <- makeflow("planar", list(u = 0.1, w = 0.2, b = 0))
s <- f$sampleq(10)

# Create a 2D spline flow
f2 <- makeflow("splinepwlin", list(d = 2, K = 8))
```

---

makeneurolik *Neural Two-Separator Likelihood*

Description

Constructs a simple 3-category likelihood model based on a latent vector z. The likelihood is defined by two logistic separators:

Usage

```
makeneurolik(a = 0.3)
```

Arguments

a	Separation parameter controlling the spacing between the two logits.

Details

p1 = sigmoid(mean(z) - a) p2 = sigmoid(mean(z) + a)

producing a 3-class probability vector:

(1 - p1, p1 - p2, p2)

This likelihood is useful for toy neural classification models or simple latent-to-categorical mappings.

Value

A function mapping a latent vector z to a probability vector.

---

planarflowmodel *Planar Normalizing Flow (1D)*

Description

A simple and readable implementation of a 1-dimensional planar flow: z_K = z_0 + u * h(wz_0 + b)

Usage

```
planarflowmodel(u, w, b)
```

Arguments

u	Scalar parameter controlling the magnitude of the deformation.
w	Scalar parameter controlling the slope of the activation.
b	Scalar bias term.

Details

where:

- h is a smooth activation (tanh)
- the log-determinant is computed analytically

This flow is mainly useful for pedagogical purposes or as a lightweight building block in variational inference.

Value

A flow model object with methods:

- sampleq(n)
- logq(z0)
- applyflow(z0)

`radialflowmodel` *Radial Normalizing Flow (1D)*

Description

A readable implementation of a 1-dimensional radial flow:

Usage

```
radialflowmodel(z_ref, alpha, beta)
```

Arguments

`z_ref`	Reference point for the radial transformation.
`alpha`	Positive scalar controlling the denominator.
`beta`	Scalar controlling the magnitude of the deformation.

Details

z_K = z_0 + beta / (alpha + |z_0 - z_ref|) * (z_0 - z_ref)

where:

- z_ref is a reference point
- alpha > 0 ensures numerical stability
- beta controls the strength of the radial deformation

The log-determinant is computed analytically for the 1D case.

Value

A flow model object with methods:

- sampleq(n)
- logq(z0)
- applyflow(z0)

sanovprob *Sanov Probability Bound*

Description

Computes the classical Sanov upper bound:

Usage

```
sanovprob(Q, P, n)
```

Arguments

Q	Observed empirical distribution.
P	True distribution.
n	Sample size.

Details

$$P(Q_n \approx Q) \leq \exp\{-n\, KL(Q\|P)\}$$

where Q is the empirical distribution and P is the true distribution.

Value

A numeric upper bound.

splinepwlinflowmodel *Piecewise-Linear Spline Flow (Monotone)*

Description

A readable implementation of a monotone piecewise-linear spline flow. Each dimension is transformed independently using learned spline parameters.

Usage

```
splinepwlinflowmodel(d = 2, K = 8, theta = NULL)
```

Arguments

d	Dimension of the latent space.
K	Number of spline bins.
theta	Optional parameter vector. If NULL, random initialization.

Details

The spline flow uses:

- K bins with learned widths w and heights h
- a softmax transformation to ensure positivity and normalization
- a sigmoid reparameterization for numerical stability

The flow is invertible and differentiable almost everywhere.

Value

A flow model object with methods:

- sampleq(n)
- logq(z0)
- applyflow(z0)

Index