Package: Langevin (via r-universe)

July 2, 2024

Type Package

Title Langevin Analysis in One and Two Dimensions

Version 1.3.2

Date 2024-07-01

Description Estimate drift and diffusion functions from time series and generate synthetic time series from given drift and diffusion coefficients.

Encoding UTF-8

License GPL (>= 2)

URL https://gitlab.uni-oldenburg.de/TWiSt/Langevin

LazyLoad yes

ByteCompile yes

NeedsCompilation yes

Depends R (>= 3.0.2)

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp, RcppArmadillo (>= 0.4.600.0)

RoxygenNote 7.3.1

Author Philip Rinn [aut, cre], Pedro G. Lind [aut], David Bastine [ctb]

Maintainer Philip Rinn <philip.rinn@uni-oldenburg.de>

Repository CRAN

Date/Publication 2024-07-01 17:40:02 UTC

Contents

Langevin1D *Calculate the Drift and Diffusion of one-dimensional stochastic processes*

Description

Langevin1D calculates the Drift and Diffusion vectors (with errors) for a given time series.

Usage

```
Langevin1D(
  data,
  bins,
  steps,
  sf = ifelse(is.ts(data), frequency(data), 1),
  bin_min = 100,
  reqThreads = -1,
  kernel = FALSE,
  h
)
```

Arguments

data	a vector containing the time series or a time-series object.
bins	a scalar denoting the number of bins to calculate the conditional moments on.
steps	a vector giving the τ steps to calculate the conditional moments (in samples (=$\tau * sf$)). Only used if kernel is FALSE.
sf	a scalar denoting the sampling frequency (optional if data is a time-series object).
bin_min	a scalar denoting the minimal number of events per bin. Defaults to 100.
reqThreads	a scalar denoting how many threads to use. Defaults to -1 which means all available cores. Only used if kernel is FALSE.
kernel	a logical denoting if the kernel based Nadaraya-Watson estimator should be used to calculate drift and diffusion vectors.
h	a scalar denoting the bandwidth of the data. Defaults to Scott's variation of Silverman's rule of thumb. Only used if kernel is TRUE.

Value

`Langevin1D` returns a list with thirteen (six if `kernel` is `TRUE`) components:

- `D1` — a vector of the Drift coefficient for each `bin`.
- `eD1` — a vector of the error of the Drift coefficient for each `bin`.
- `D2` — a vector of the Diffusion coefficient for each `bin`.
- `eD2` — a vector of the error of the Diffusion coefficient for each `bin`.
- `D4` — a vector of the fourth Kramers-Moyal coefficient for each `bin`.
- `mean_bin` — a vector of the mean value per `bin`.
- `density` — a vector of the number of events per `bin`. If `kernel` is `FALSE`.
- `M1` — a matrix of the first conditional moment for each τ. Rows correspond to `bin`, columns to τ. If `kernel` is `FALSE`.
- `eM1` — a matrix of the error of the first conditional moment for each τ. Rows correspond to `bin`, columns to τ. If `kernel` is `FALSE`.
- `M2` — a matrix of the second conditional moment for each τ. Rows correspond to `bin`, columns to τ. If `kernel` is `FALSE`.
- `eM2` — a matrix of the error of the second conditional moment for each τ. Rows correspond to `bin`, columns to τ. If `kernel` is `FALSE`.
- `M4` — a matrix of the fourth conditional moment for each τ. Rows correspond to `bin`, columns to τ. If `kernel` is `FALSE`.
- `U` — a vector of the `bin` borders. If `kernel` is `FALSE`.

Author(s)

Philip Rinn

See Also

`Langevin2D`

Examples

```
# Set number of bins, steps and the sampling frequency
bins <- 20
steps <- c(1:5)
sf <- 1000

#### Linear drift, constant diffusion

# Generate a time series with linear D^1 = -x and constant D^2 = 1
x <- timeseries1D(N = 1e6, d11 = -1, d20 = 1, sf = sf)
# Do the analysis
est <- Langevin1D(data = x, bins = bins, steps = steps, sf = sf)
# Plot the result and add the theoretical expectation as red line
plot(est$mean_bin, est$D1)
lines(est$mean_bin, -est$mean_bin, col = "red")
```

```
plot(est$mean_bin, est$D2)
abline(h = 1, col = "red")

#### Cubic drift, constant diffusion

# Generate a time series with cubic D^1 = x - x^3 and constant D^2 = 1
x <- timeseries1D(N = 1e6, d13 = -1, d11 = 1, d20 = 1, sf = sf)
# Do the analysis
est <- Langevin1D(data = x, bins = bins, steps = steps, sf = sf)
# Plot the result and add the theoretical expectation as red line
plot(est$mean_bin, est$D1)
lines(est$mean_bin, est$mean_bin - est$mean_bin^3, col = "red")
plot(est$mean_bin, est$D2)
abline(h = 1, col = "red")
```

Langevin2D *Calculate the Drift and Diffusion of two-dimensional stochastic processes*

Description

`Langevin2D` calculates the Drift (with error) and Diffusion matrices for given time series.

Usage

```
Langevin2D(
  data,
  bins,
  steps,
  sf = ifelse(is.mts(data), frequency(data), 1),
  bin_min = 100,
  reqThreads = -1
)
```

Arguments

`data`	a matrix containing the time series as columns or a time-series object.
`bins`	a scalar denoting the number of `bins` to calculate Drift and Diffusion on.
`steps`	a vector giving the τ steps to calculate the moments (in samples).
`sf`	a scalar denoting the sampling frequency (optional if `data` is a time-series object).
`bin_min`	a scalar denoting the minimal number of events per `bin`. Defaults to `100`.
`reqThreads`	a scalar denoting how many threads to use. Defaults to `-1` which means all available cores.

Value

`Langevin2D` returns a list with nine components:

- `D1` — a tensor with all values of the drift coefficient. Dimension is `bins` x `bins` x 2. The first `bins` x `bins` elements define the drift $D_1^{(1)}$ for the first variable and the rest define the drift $D_2^{(1)}$ for the second variable.
- `eD1` — a tensor with all estimated errors of the drift coefficient. Dimension is `bins` x `bins` x 2. Same expression as above.
- `D2` — a tensor with all values of the diffusion coefficient. Dimension is `bins` x `bins` x 3. The first `bins` x `bins` elements define the diffusion $D_{11}^{(2)}$, the second `bins` x `bins` elements define the diffusion $D_{22}^{(2)}$ and the rest define the diffusion $D_{12}^{(2)} = D_{21}^{(2)}$.
- `mean_bin` — a matrix of the mean value per `bin`. Dimension is `bins` x `bins` x 2. The first `bins` x `bins` elements define the mean for the first variable and the rest for the second variable.
- `density` — a matrix of the number of events per `bin`. Rows label the `bin` of the first variable and columns the second variable.
- `M1` — a tensor of the first moment for each `bin` (line label) and each τ step (column label). Dimension is `bins` x `bins` x `2length(steps)`.
- `eM1` — a tensor of the standard deviation of the first moment for each bin (line label) and each τ step (column label). Dimension is `bins` x `bins` x `2length(steps)`.
- `M2` — a tensor of the second moment for each bin (line label) and each τ step (column label). Dimension is `bins` x `bins` x `3length(steps)`.
- `U` — a matrix of the `bin` borders

Author(s)

Philip Rinn

See Also

Langevin1D

`plot.Langevin` *Plot estimated drift and diffusion coefficients*

Description

plot method for class "Langevin". This method is only implemented for one-dimensional analysis for now.

Usage

```
## S3 method for class 'Langevin'
plot(x, pch = 20, ...)
```

Arguments

x	an object of class "Langevin".
pch	Either an integer specifying a symbol or a single character to be used as the default in plotting points. See points for possible values and their interpretation. Default is `pch = 20`.
...	Arguments to be passed to methods, such as `par`.

Author(s)

Philip Rinn

`print.Langevin` *Print estimated drift and diffusion coefficients*

Description

print method for class "Langevin".

Usage

```
## S3 method for class 'Langevin'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	an object of class "Langevin".
digits	integer, used for number formatting with `signif()`.
...	further arguments to be passed to or from other methods. They are ignored in this function.

Value

The function `print.Langevin()` returns an overview of the estimated drift and diffusion coefficients.

Author(s)

Philip Rinn

`summary.Langevin` *Summarize estimated drift and diffusion coefficients*

Description

summary method for class "Langevin".

Usage

```
## S3 method for class 'Langevin'
summary(object, ..., digits = max(3, getOption("digits") - 3))
```

Arguments

`object`	an object of class "Langevin".
`...`	arguments to be passed to or from other methods. They are ignored in this function.
`digits`	integer, used for number formatting with `signif()`.

Value

The function `summary.Langevin()` returns a summary of the estimated drift and diffusion coefficients

Author(s)

Philip Rinn

`timeseries1D` *Generate a 1D Langevin process*

Description

`timeseries1D` generates a one-dimensional Langevin process using a simple Euler integration. The drift function is a cubic polynomial, the diffusion function a quadratic.

Usage

```
timeseries1D(
  N,
  startpoint = 0,
  d13 = 0,
  d12 = 0,
  d11 = -1,
  d10 = 0,
```

```
    d22 = 0,
    d21 = 0,
    d20 = 1,
    sf = 1000,
    dt = 0
  )
```

Arguments

N	a scalar denoting the length of the time-series to generate.
startpoint	a scalar denoting the starting point of the time series.
d13, d12, d11, d10	scalars denoting the coefficients for the drift polynomial.
d22, d21, d20	scalars denoting the coefficients for the diffusion polynomial.
sf	a scalar denoting the sampling frequency.
dt	a scalar denoting the maximal time step of integration. Default dt=0 yields dt=1/sf.

Value

timeseries1D returns a time-series object of length N with the generated time-series.

Author(s)

Philip Rinn

See Also

timeseries2D

Examples

```
# Generate standardized Ornstein-Uhlenbeck-Process (d11=-1, d20=1)
# with integration time step 0.01 and sampling frequency 1
s <- timeseries1D(N=1e4, sf=1, dt=0.01);
t <- 1:1e4;
plot(t, s, t="l", main=paste("mean:", mean(s), " var:", var(s)));
```

---

timeseries2D *Generate a 2D Langevin process*

Description

timeseries2D generates a two-dimensional Langevin process using a simple Euler integration. The drift function is a cubic polynomial, the diffusion function a quadratic.

Usage

```
timeseries2D(
  N,
  startpointx = 0,
  startpointy = 0,
  D1_1 = matrix(c(0, -1, rep(0, 14)), nrow = 4),
  D1_2 = matrix(c(0, 0, 0, 0, -1, rep(0, 11)), nrow = 4),
  g_11 = matrix(c(1, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 3),
  g_12 = matrix(c(0, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 3),
  g_21 = matrix(c(0, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 3),
  g_22 = matrix(c(1, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 3),
  sf = 1000,
  dt = 0
)
```

Arguments

`N`	a scalar denoting the length of the time-series to generate.
`startpointx`	a scalar denoting the starting point of the time series x.
`startpointy`	a scalar denoting the starting point of the time series y.
`D1_1`	a 4x4 matrix denoting the coefficients of D1 for x.
`D1_2`	a 4x4 matrix denoting the coefficients of D1 for y.
`g_11`	a 3x3 matrix denoting the coefficients of g11 for x.
`g_12`	a 3x3 matrix denoting the coefficients of g12 for x.
`g_21`	a 3x3 matrix denoting the coefficients of g21 for y.
`g_22`	a 3x3 matrix denoting the coefficients of g22 for y.
`sf`	a scalar denoting the sampling frequency.
`dt`	a scalar denoting the maximal time step of integration. Default `dt=0` yields `dt=1/sf`.

Details

The elements a_{ij} of the matrices are defined by the corresponding equations for the drift and diffusion terms:

$$D^1_{1,2} = \sum_{i,j=1}^{4} a_{ij} x_1^{(i-1)} x_2^{(j-1)}$$

with $a_{ij} = 0$ for $i + j > 5$.

$$g_{11,12,21,22} = \sum_{i,j=1}^{3} a_{ij} x_1^{(i-1)} x_2^{(j-1)}$$

with $a_{ij} = 0$ for $i + j > 4$

Value

`timeseries2D` returns a time-series object with the generated time-series as columns.

Author(s)

Philip Rinn

See Also

`timeseries1D`

Index