Flexible stochastic differential equation estimation with smoothSDE

Théo Michelot

2021-04-20

The R package smoothSDE implements varying-coefficient versions of several popular stochastic differential equations (SDEs). It can be used to estimate linear or non-linear relationships between SDE parameters and time-varying covariates. It also provides functions for model visualisation, uncertainty estimation, and residual analysis.

1 Package description

1.1 Varying-coefficient stochastic differential equations

Varying-coefficient SDEs, and the inferential method implemented in smoothSDE, are described by Michelot et al. (2021). The main model formulations currently implemented in the package are Brownian motion, Ornstein-Uhlenbeck process, and continuous-time correlated random walk. Brownian motion and the Ornstein-Uhlenbeck process have a wide range of applications (e.g., ecology, finance), and the continuous-time correlated random walk is a popular model of animal movement described by Johnson et al. (2008). The package supports multivariate isotropic processes, i.e., with the same set of parameters describing the dynamics in each dimension.

Model	Name	Parameters		Link functions
Brownian motion	“`BM`”	drift	$\mu \in (-\infty, +\infty)$	identity
		diffusion	$\sigma > 0$	log
Ornstein-Uhlenbeck process	“`OU`”	mean	$\mu \in (-\infty, +\infty)$	identity
		time scale	$\tau > 0$	log
		variance	$\kappa > 0$	log
Continuous-time correlated RW	“`CTCRW`”	reversion	$\beta > 0$	log
		diffusion	$\sigma > 0$	log

1.2 Brownian motion

Varying-coefficient Brownian motion (with drift) is the process (Z_t) defined as the solution to

$$dZ_t = \mu_t \; dt + \sigma_t \; dW_t,$$

where $\mu_t \in \mathbb{R}$ is the drift parameter (expected direction of change), and $\sigma_t > 0$ is the diffusion parameter (variability).

The approximate transition density of this model is

$$Z_{t_1} | \{Z_{t_0} = z_0\} \sim N\left[z_0 + \mu_{t_0}\Delta,\; \sigma_{t_0}^2 \Delta\right],$$

where $\Delta = t_1 - t_0$ is the time interval.

Figure 1 shows simulated trajectories from 1-dimensional and 2-dimensional Brownian motion (with constant parameters).

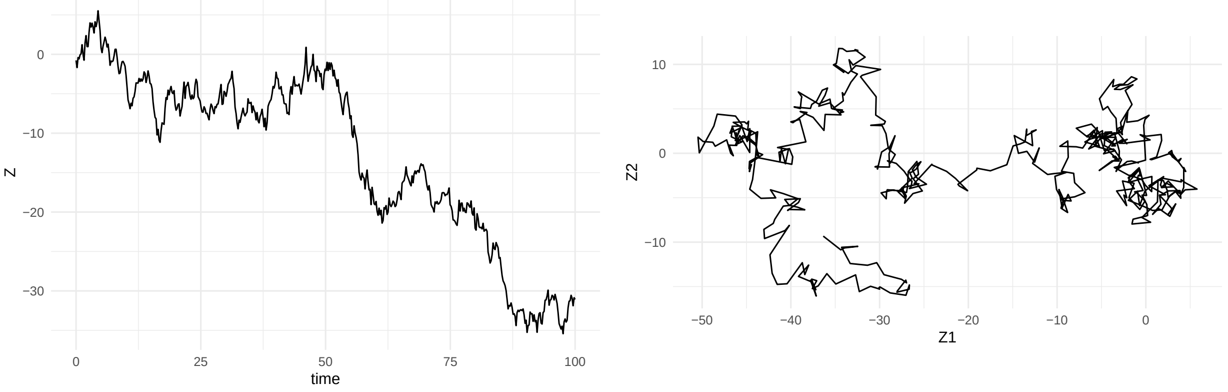


Figure 1: Illustration of 1d and 2d (isotropic) Brownian motion.

1.3 Ornstein-Uhlenbeck

The Ornstein-Uhlenbeck (OU) process is usually described as the solution to

$$dZ_t = \beta_t(\mu_t - Z_t)\ dt + \sigma_t\ dt,$$

where $\mu_t \in \mathbb{R}$ is the long-term mean, $\beta_t > 0$ is the strength of reversion to the mean, and $\sigma_t > 0$ is the diffusion parameter.

In our experience, the parameterisation shown above suffers from identifiability issues. In `smoothSDE`, the varying-coefficient OU process is therefore specified as the solution to

$$dZ_t = \frac{1}{\tau_t}(\mu_t - Z_t)\ dt + \sqrt{\frac{2\kappa_t}{\tau_t}}\ dt,$$

where μ_t has the same interpretation as before, $\tau_t > 0$ measures the time scale of autocorrelation of the process, and $\kappa_t > 0$ is the long-term variance of the process. The following formulas can be used to go from one parameterisation to the other:

$$\begin{cases} \beta_t = \frac{1}{\tau_t} \\ \sigma_t = \sqrt{\frac{2\kappa_t}{\tau_t}} \end{cases} \quad \Leftrightarrow \begin{cases} \tau_t = \frac{1}{\beta_t} \\ \kappa_t = \frac{\sigma_t^2}{2\beta_t} \end{cases}$$

The approximate transition density of the varying-coefficient OU process is

$$Z_{t_1} | \{Z_{t_0} = z_0\} \sim N\left[e^{-\Delta/\tau_t} z_0 + (1 - e^{-\Delta/\tau_t})\mu_t,\ \kappa_t(1 - e^{-2\Delta/\tau_t})\right].$$

From this, we see that the stationary distribution of the process (obtained when $\Delta \to \infty$) is

$$Z_t \sim N(\mu_t,\ \kappa_t).$$

Simulations from the standard OU process (with constant parameters) are shown in Figure 2 for illustration. The process oscillates around its long-term mean.

1.4 Continuous-time correlated random walk

The varying-coefficient continuous-time correlated random walk (CTCRW), described by Johnson et al. (2008), is the process (Z_t) solution to

$$\begin{cases} dZ_t = V_t\ dt \\ dV_t = -\beta_t V_t\ dt + \sigma_t\ dW_t. \end{cases}$$

In animal movement analyses, Z_t usually denotes the location of the animal at time t, and V_t is its velocity. The velocity V_t is modelled with an OU process to capture autocorrelation

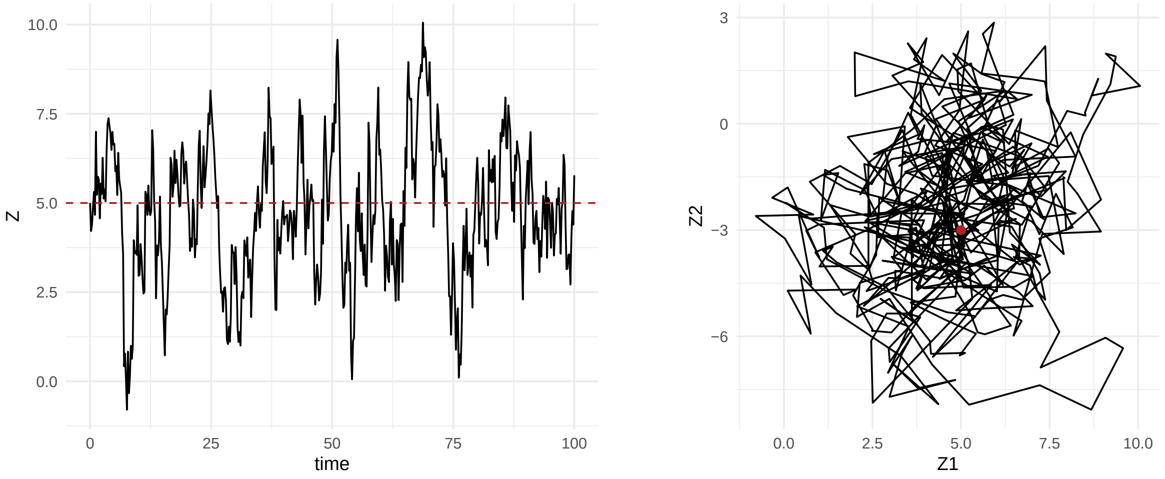


Figure 2: Illustration of 1d and 2d (isotropic) Ornstein-Uhlenbeck process. The mean of the process is shown as a horizontal dotted line in the 1-d example, and as a red point in the 2-d example.

in speed and direction of movement, and the location Z_t is its integral, i.e. $Z_t = \int_{s=0}^{t} V_s \; ds$. The parameters of the velocity process are $\beta_t > 0$, the strength of reversion to the mean, and $\sigma_t > 0$, the variability of the velocity.

Simulated examples of the CTCRW process are shown in Figure 3. Compared to Brownian motion, realisations of the CTCRW are smooth and display persistence in the direction and speed of change.

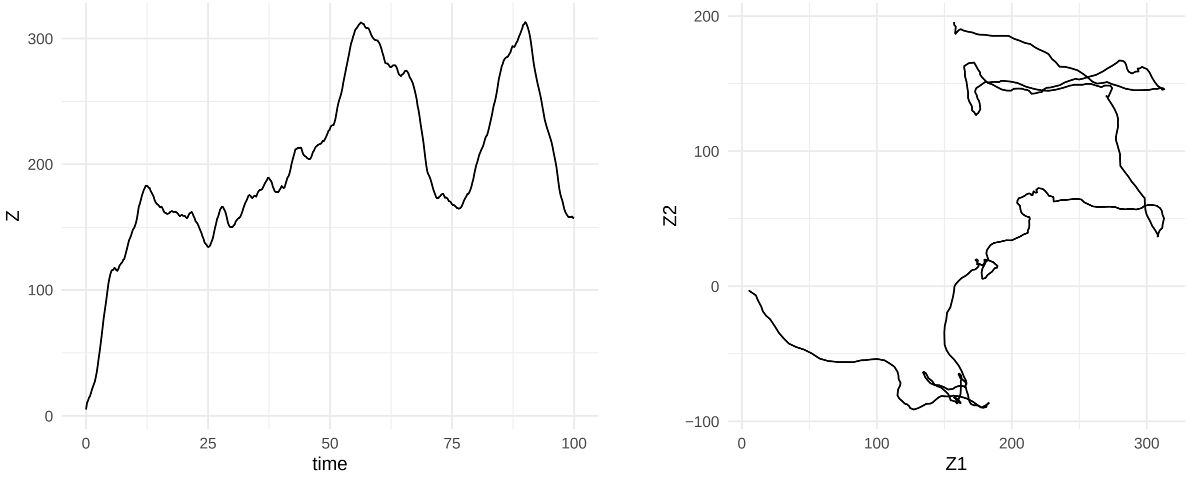


Figure 3: Illustration of 1d and 2d (isotropic) continuous-time correlated random walk.

1.5 SDE class

The package is centred around the R6 class `SDE`, which encapsulates the model formulas and the data for a varying-coefficient SDE model. The constructor `SDE$new` can be used to create a model object, and its main arguments are:

- `formulas`: List of formulas for the parameters of the SDE. The formulas can include terms from standard R expressions, as well as smooth terms and random effects from the mgcv package (Wood (2017)).
- `data`: Data frame with columns for the response variable, for the covariate, for ID, and for time.
- `type`: Type of SDE; options are "BM" (Brownian motion), "OU" (Ornstein-Uhlenbeck), "CTCRW" (continuous-time correlated random walk).
- `response`: Name of response variable(s).

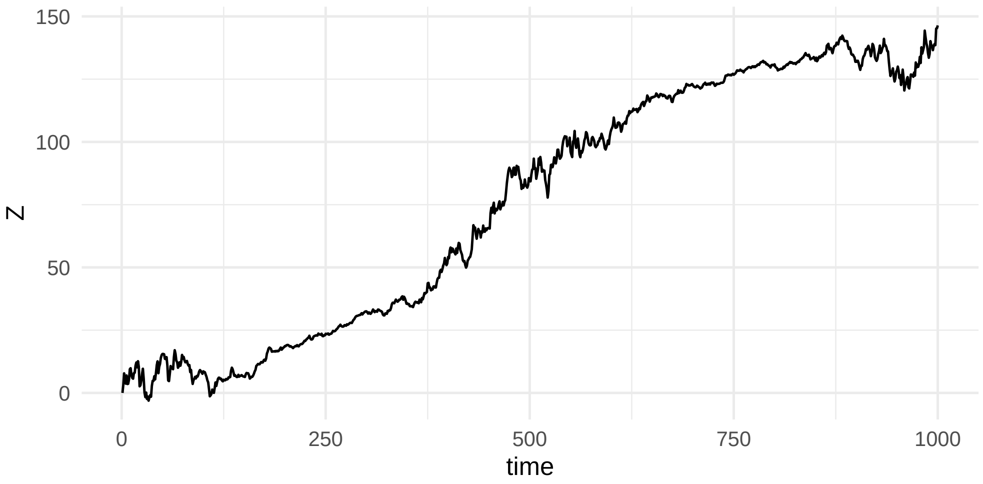
2 Example 1: Brownian motion with drift

We illustrate some of the functionalities of the package using simulated data for a varying-coefficient Brownian motion with drift. We first simulate a process with constant drift (`mu`) and time-varying diffusion parameter (`sigma`) over 1000 time steps.

```r
n <- 1000
times <- 1:n
mu <- rep(0.1, n)
sigma <- exp(cos(2*pi*times/500))
dZ <- rnorm(n - 1, mean = mu[-n], sd = sigma[-n])
Z <- cumsum(c(0, dZ))
```

We define a data frame with columns `ID` (identifier for time series segment), `Z` (process of interest), and `time` (times of observation).

```r
data <- data.frame(ID = 1,
                   Z = Z,
                   time = times)

ggplot(data, aes(time, Z)) + geom_line()
```



In this example, we are interested in estimating the smooth relationship between the diffusion parameter and time. We specify this dependence using a list of R formulas with one entry for each SDE parameter. Here, the diffusion parameter sigma is modelled with a thin-plate regression spline basis with 10 knots, using the mgcv syntax.

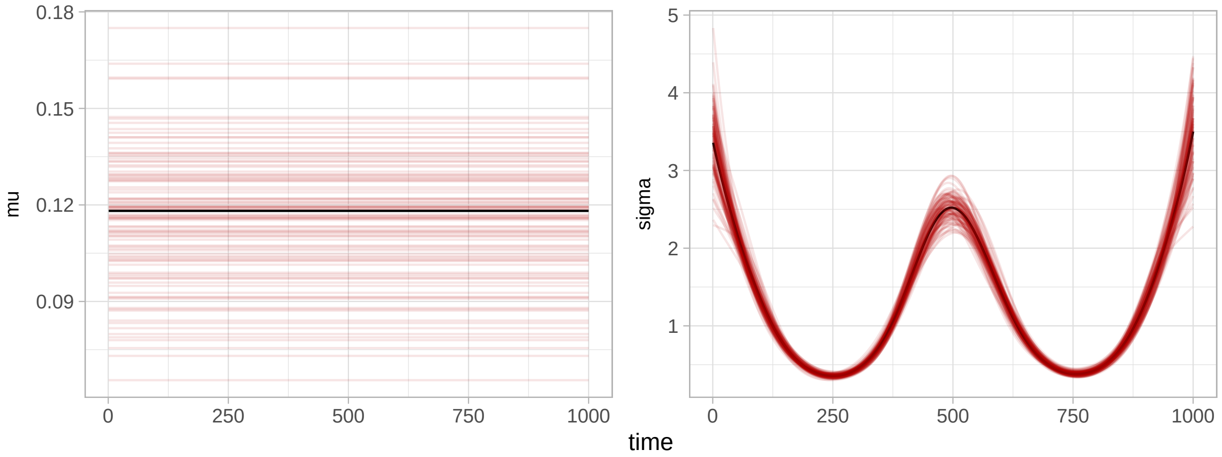
```
formulas <- list(mu = ~1,
                 sigma = ~ s(time, k = 10, bs = "ts"))
```

We create an SDE object that encapsulates the formulas, the data, the type of model (here, Brownian motion), and the name of the response variable (here, `Z`). Then, we can fit the model using the method `fit`.

```
bm <- SDE$new(formulas = formulas,
              data = data,
              type = "BM",
              response = "Z")

bm$fit()
```

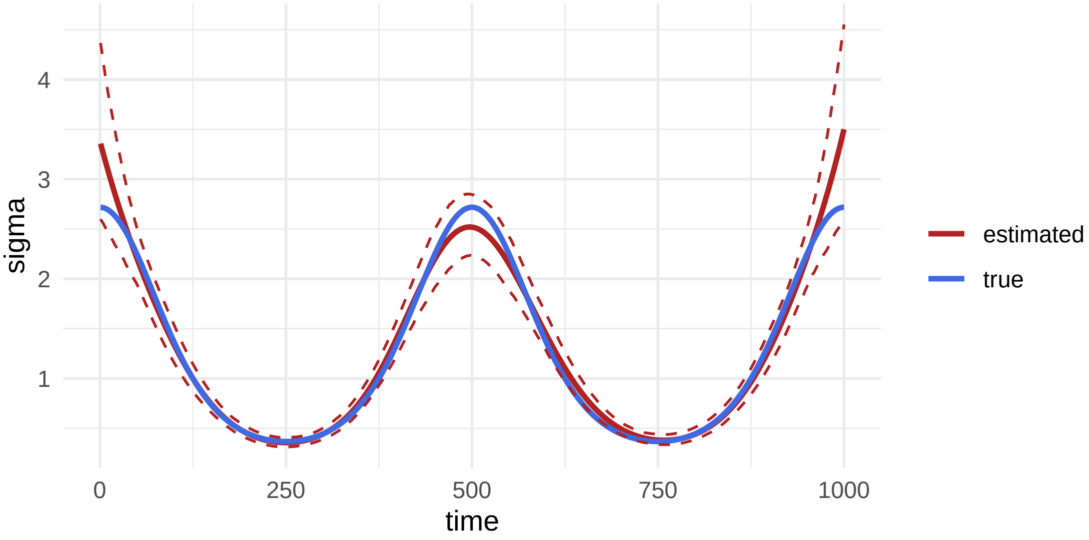
Once the model is fitted, we can visualise the relationship between the SDE parameters and the covariates with `plot_par`. The option `n_post` can be used to visualise uncertainty with posterior samples.

```
bm$plot_par("time", n_post = 100)
```



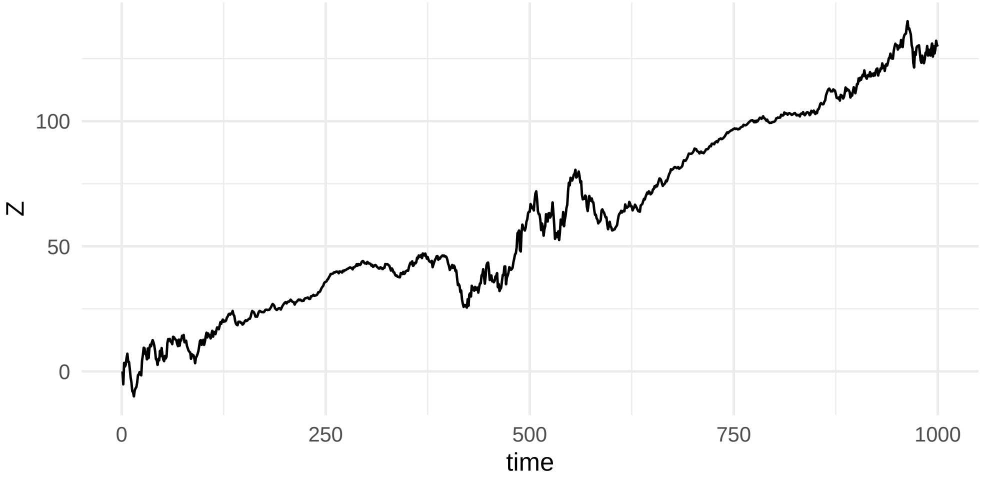
The smoothing splines captured the oscillating patterns in the diffusion parameter, and the drift parameter was slightly underestimated. We can also evaluate the SDE parameters on a grid of covariate values, to compare directly to the true parameters.

```
# Get point estimates and CIs for sigma over time
bm_par <- bm$predict_par(new_data = data, CI = TRUE)

# Data frame for point estimates
bm_par_df <- data.frame(time = times,
                        sigma = c(sigma, bm_par$estimate[, "sigma"]),
                        type = rep(c("true", "estimated"), each = n))

# Data frame for CIs
bm_ci_df <- data.frame(time = times,
                       low = bm_par$low[,"sigma"],
                       upp = bm_par$upp[,"sigma"])

ggplot(bm_par_df, aes(time, sigma, col = type)) +
  scale_color_manual(values = c("firebrick", "royalblue"), name = "") +
  geom_line(size = 1) +
  geom_line(aes(time, low), data = bm_ci_df, col = "firebrick", lty = 2) +
  geom_line(aes(time, upp), data = bm_ci_df, col = "firebrick", lty = 2)
```



We can simulate from the fitted model with the method `simulate` (only available for Brownian motion and Ornstein-Uhlenbeck models for now), for example to assess whether the model captures relevant features of the observed data.

```
# Pass 'data' to use observed covariates in simulation
sim <- bm$simulate(data = data)

ggplot(sim, aes(time, Z)) + geom_line()
```

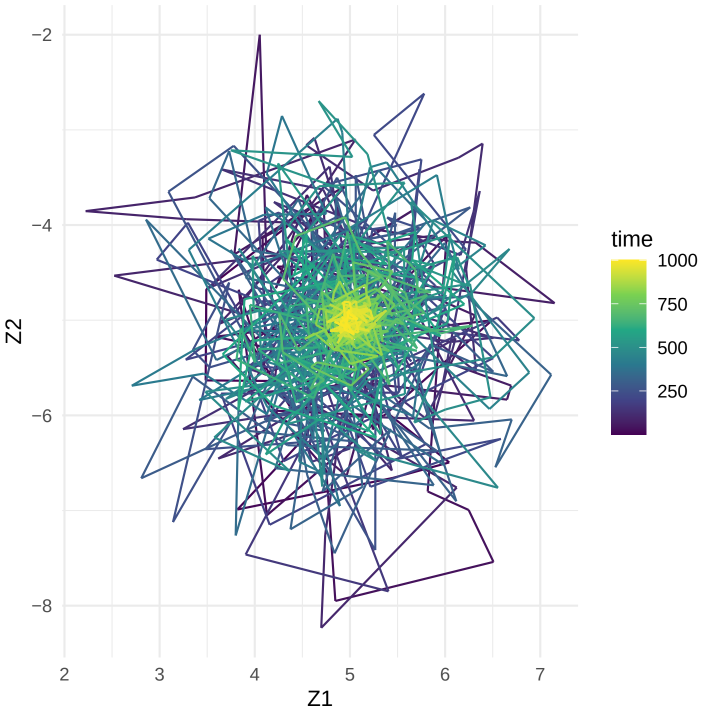


Similarly to the observed time series, the simulated process alternates between periods of low variability (e.g., around $t = 250$ and $t = 750$) and periods of high variability (e.g., around $t = 0$, $t = 500$ and $t = 1000$).

3 Example 2: Ornstein-Uhlenbeck

We present a second simulated example, based on a varying-coefficient Ornstein-Uhlenbeck (OU) process. We generate 1000 observations from a 2-dimensional isotropic OU process with constant mean (`mu`), constant mean reversion (`beta`), and time-varying diffusion (`sigma`). In this example, the parameters are selected such that the diffusion (and therefore the variance of the process around its mean) decreases in time.

```r
# Mean parameter (centre of attraction)
mu <- matrix(rep(c(5, -5), each = n), ncol = 2)
# Time scale of autocorrelation
tau <- rep(2, n)
# Time-varying variance parameter
kappa <- plogis(-(times-500)/100)

# Simulate OU process over 1000 time steps
Z <- mu
for(i in 2:n) {
    mean <- exp(-1/tau[i-1]) * Z[i-1,] + (1 - exp(-1/tau[i-1])) * mu[i-1,]
    sd <- sqrt(kappa[i-1]) * sqrt(1 - exp(-2 / tau[i-1]))
    Z[i,] <- rnorm(2, mean, sd)
}

# Create data frame for smoothSDE
data <- data.frame(ID = 1,
                   Z1 = Z[,1],
                   Z2 = Z[,2],
                   time = times)

# Plot simulated data, coloured by time
ggplot(data, aes(Z1, Z2, col = time)) +
    geom_path() +
    coord_equal() +
    scale_color_viridis_c()
```

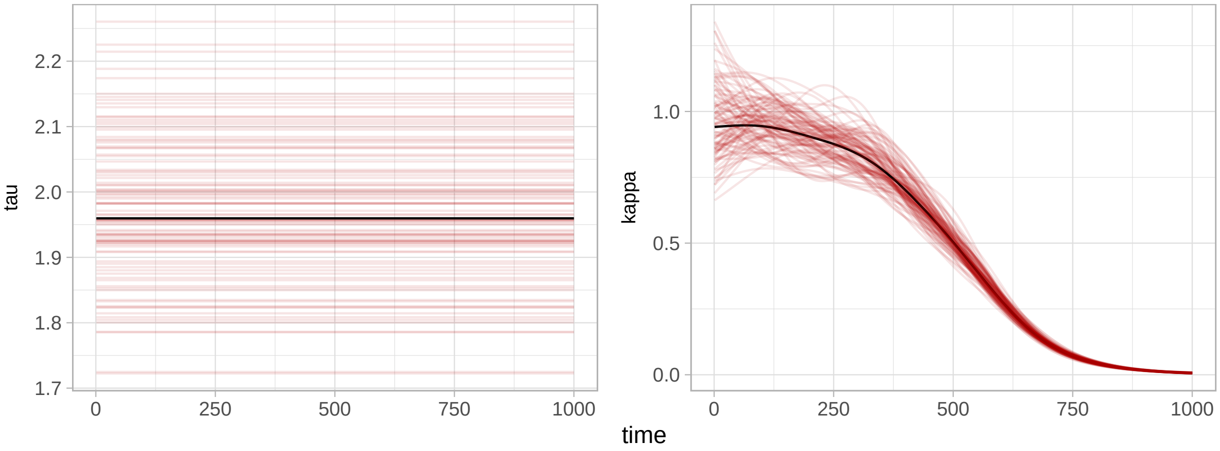


Like in the previous example, we specify the model for the OU parameters with a list of formulas. There are four parameters: `mu1` (mean in first dimension), `mu2` (mean in second dimension), `tau` (time scale of autocorrelation) and `kappa` (variance). In this example, we assume that the mean parameter is known, and we therefore use the option `fixpar = c("mu1", "mu2")` to indicate that it should not be estimated, and we add the argument `par0` to give the known mean parameter (and initial values for the other parameters). We pass a vector of length 2 for `response`, to indicate that we want to model the two variables with a 2-dimensional isotropic OU process.

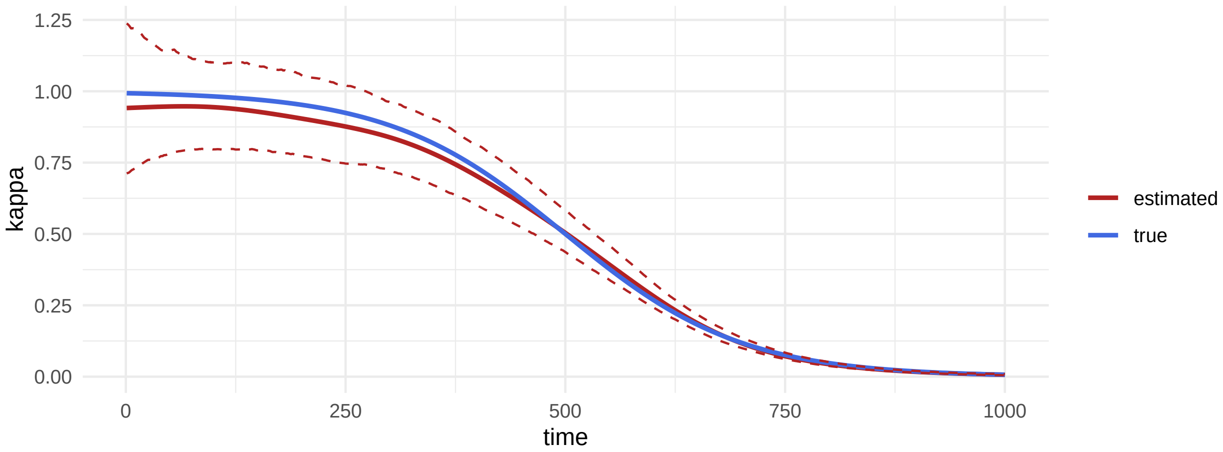
```
formulas <- list(mu1 = ~1,
                 mu2 = ~1,
                 tau = ~1,
                 kappa = ~s(time, k = 10, bs = "ts"))

par0 <- c(mu1 = 5, mu2 = -5, tau = 1, kappa = 3)

ou <- SDE$new(formulas = formulas,
              data = data,
              type = "OU",
              response = c("Z1", "Z2"),
              par0 = par0,
                fixpar = c("mu1", "mu2"))

ou$fit()
```

We can visualise the results to check that the estimated parameters correctly captured the true parameters used in the simulation.

```
# Plot tau and kappa against time
ou$plot_par("time", par_names = c("tau", "kappa"))
```



```
# Point estimates and CIs for kappa over time
ou_par <- ou$predict_par(new_data = data, CI = TRUE)

# Data frame for point estimates
ou_par_df <- data.frame(time = times,
                        kappa = c(kappa, ou_par$estimate[, "kappa"]),
                        type = rep(c("true", "estimated"), each = n))

# Data frame for CIs
ou_ci_df <- data.frame(time = times,
                       low = ou_par$low[, "kappa"],
                       upp = ou_par$upp[, "kappa"])

ggplot(ou_par_df, aes(time, kappa, col = type)) +
  scale_color_manual(values = c("firebrick", "royalblue"), name = "") +
  geom_line(size = 1) +
  geom_line(aes(time, low), data = ou_ci_df, col = "firebrick", lty = 2) +
  geom_line(aes(time, upp), data = ou_ci_df, col = "firebrick", lty = 2)
```



The decreasing variance parameter seems to be captured well by the model.

4 Example 3: elephant movement analysis

The following code can be used to fit the varying-coefficient CTCRW model to elephant GPS data from Wall et al. (2014), as described in Section 3.2 of Michelot et al. (2021).

We first download the data from the Movebank data repository, and keep the relevant rows and columns. Note that the CTCRW model requires projected Easting-Northing locations (rather than longitude-latitude), so that's what we are working with here.

```
# Load data and keep relevant columns
URL <- paste0("https://www.datarepository.movebank.org/bitstream/handle/",
              "10255/move.373/Elliptical%20Time-Density%20Model%20%28Wall%",
              "20et%20al.%202014%29%20African%20Elephant%20Dataset%20%",
              "28Source-Save%20the%20Elephants%29.csv")
raw <- read.csv(url(URL))
keep_cols <- c(11, 13, 14, 17, 4, 5, 6)
raw_cols <- raw[, keep_cols]
colnames(raw_cols) <- c("ID", "x", "y", "date", "lon", "lat", "temp")

# Only keep five months to eliminate seasonal effects
track <- subset(raw_cols, ID == unique(ID)[1])
dates <- as.POSIXlt(track$date, tz = "GMT")
times <- as.numeric(dates - min(dates))/3600
keep_rows <- which(dates > as.POSIXct("2009-05-01 00:00:00") &
                     dates < as.POSIXct("2009-09-30 23:59:59"))
```

```
track <- track[keep_rows,]
dates <- dates[keep_rows]
times <- times[keep_rows]

# Convert to km
track$x <- track$x/1000
track$y <- track$y/1000
```

We create a data frame that includes columns for

- time series identifier `ID`. This is required when fitting the model to several time series, treated as independent realisations of the same underlying process. Here, we only have one time series, so all rows have the same ID.
- responses `x` and `y`. Here, there are two response variables because we are fitting an isotropic CTCRW model to the two-dimensional observations (Easting and Northing).
- covariate `temp`. We will later estimate the effect of this covariate on the parameters of the elephant's velocity process.
- `time`. This should be a numeric column for time, used to compute time intervals between observations.

```
data <- data.frame(ID = 1,
                   x = track$x,
                   y = track$y,
                   temp = track$temp,
                   time = times)

head(data)
```

```
  ID        x        y temp time
1  1 572.3427 1675.424   33 9703
2  1 572.5443 1675.392   32 9704
3  1 572.6159 1675.339   31 9705
4  1 572.7745 1675.101   31 9706
5  1 572.8844 1675.065   31 9707
6  1 573.6659 1674.322   30 9708
```

In this analysis, we want to look into the effects of external temperature on the elephant's movement. We specify this by expressing both parameters of the CTCRW model (velocity reversion β and velocity diffusion σ) as smooth functions of the temperature. For these smooth terms, we use the syntax from the R package mgcv; here defining cubic regression splines with 10 basis functions and with shrinkage. See the mgcv documentation for more details.

```
formulas <- list(beta = ~ s(temp, k = 10, bs = "cs"),
                 sigma = ~ s(temp, k = 10, bs = "cs"))
```

Finally, we use the constructor of the SDE class to create an object for this model, passing as input the formulas, data, type of model, and name of response variables.

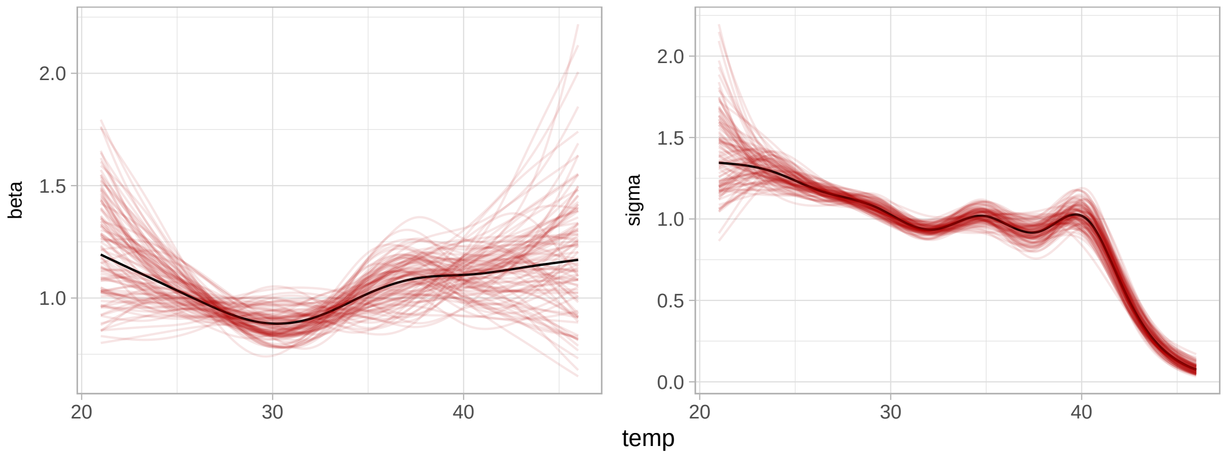
```
my_sde <- SDE$new(formulas = formulas,
                  data = data,
                  type = "CTCRW",
                  response = c("x", "y"))
```

Once the model has been created, the parameters can be estimated using the method `fit`:

```
my_sde$fit()
```

Finally, the relationship between the SDE parameters and the covariates can be plotted with the method `plot_par`. The function can also generate posterior samples for the estimated relationship, to visualise uncertainty. Here, we plot the CTCRW parameters as function of temperature, with 100 posterior samples:

```
my_sde$plot_par("temp", n_post = 100)
```



As described by Michelot et al. (2021), these results suggest that this elephant tended to move less at high temperatures, with a big drop in speed over 40 degrees Celsius. Estimates and point-wise confidence intervals of the SDE parameters can also be computed for given covariate values, using the `predict_par` method.

```
# Data frame with covariate values for prediction
new_data <- data.frame(temp = c(20, 30, 40))

# Get estimates and CIs
par <- my_sde$predict_par(new_data = new_data, CI = TRUE)

# Estimates and 95% CIs for each temperature value given as input
par
```

```
$estimate
         beta    sigma
[1,] 1.234930 1.354184
[2,] 0.886193 1.024858
[3,] 1.103145 1.019070

$low
          beta     sigma
[1,] 0.7856456 0.7987292
[2,] 0.7795873 0.9644933
[3,] 0.9243158 0.8875791

$upp
         beta    sigma
[1,] 2.020229 2.284550
[2,] 1.006313 1.088361
[3,] 1.295850 1.170192
```

References

Johnson, Devin S, Joshua M London, Mary-Anne Lea, and John W Durban. 2008. "Continuous-Time Correlated Random Walk Model for Animal Telemetry Data." *Ecology* 89 (5): 1208–15.

Michelot, Théo, Richard Glennie, Catriona Harris, and Len Thomas. 2021. "Varying-Coefficient Stochastic Differential Equations with Applications in Ecology." *Journal of Agricultural, Biological and Environmental Statistics*.

Wall, Jake, George Wittemyer, Valerie LeMay, Iain Douglas-Hamilton, and Brian Klinkenberg. 2014. "Elliptical Time-Density Model to Estimate Wildlife Utilization Distributions." *Methods in Ecology and Evolution* 5 (8): 780–90.

Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with R*. CRC press.