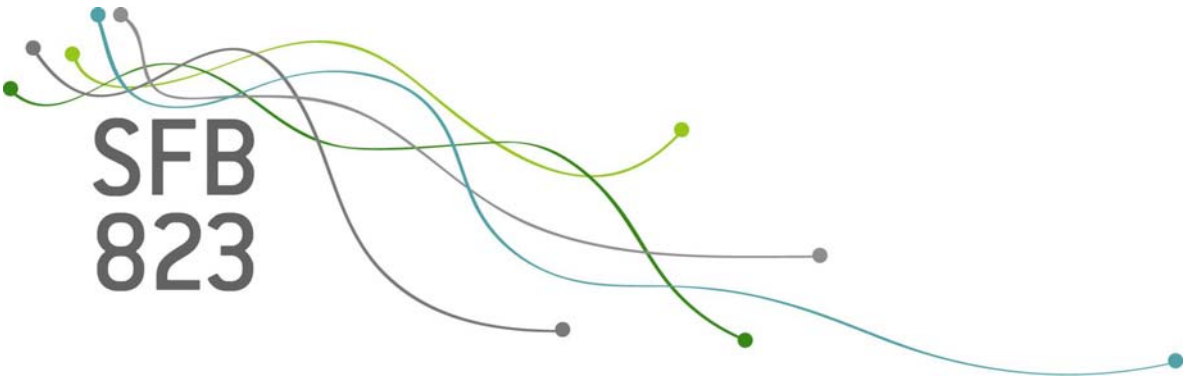
SFB 823

Discussion Paper

# BaPreStoPro: an R package for Bayesian prediction of stochastic processes

Simone Hermann

Nr. 28/2016



# BaPreStoPro: an R package for Bayesian prediction of stochastic processes

Simone Hermann*

June 6, 2016

In many applications, stochastic processes are used for modeling. Bayesian analysis is a strong tool for inference as well as for prediction. We here present an R package for a large class of models, all based on the definition of a jump diffusion with a non-homogeneous Poisson process. Special cases, as the Poisson process itself, a general diffusion process or a hierarchical (mixed) diffusion model, are considered. The package is a general tool box, because it is based on the stochastic differential equation, approximated with the Euler scheme. Functions for simulation, estimation and prediction are provided for each considered model.

*Keywords:* Bayesian estimation, Euler-Maruyama approximation, hierarchical (mixed) model, hidden Markov model, (jump) diffusion, stochastic differential equation.

## 1 Introduction

Time-continuous stochastic processes are widely used for modeling in many areas, for example in biometrical applications, see Donnet et al. (2010); Ditlevsen and Gaetano (2005), finance, see Genon-Catalot et al. (2000), material fatigue, see Hermann et al. (2016b); Hermann and Ruggeri (2016), or many others.

If one specific model is considered, maybe with an explicit solution of the stochastic differential equation (SDE), inference often is tailored to the specific model properties. This leads to specialized software implementations that often are not easy to generalize or transferable to other models. The here presented package is based on the general definition of the SDE with arbitrary drift, variance and jump high function of a jump diffusion process. This leads to a large area of application fields. The methods presented in the following, are implemented in R, see R Core Team (2015), in the package BaPreStoPro, which is available on `github.com/SimoneHermann/BaPreStoPro`.

There is a wide range of models, that will be considered. In some applications, data are collected at different individuals and, therefore, several series in possibly different time points have to be modeled. A hierarchical model might be suitable in this case, see for example Hermann et al. (2016b). Dependent on the kind of measurement, the diffusion itself might

*TU Dortmund University, Faculty of Statistics, Vogelpothsweg 87, D-44221 Dortmund, Germany, hermann@statistik.tu-dortmund.de

be observed not directly but noisy. In this case, a hidden Markov model can be suitable, see Donnet et al. (2010). There are also applications with jumps in the data series, where a jump diffusion can be of interest, see Hermann et al. (2016a); Platen and Bruti-Liberati (2010).

The models considered in the following are based on the process $\{Y_t, t \in [0,T]\}$ described by the stochastic differential equation

$$dY_t = b(\phi, t, Y_t)\, dt + s(\gamma, t, Y_t)\, dW_t + h(\theta, t, Y_t)\, dN_t, Y_0 = y_0(\phi), \tag{1}$$

where $\{W_t,\ t \in [0,T]\}$ denotes a Brownian motion and $\{N_t,\ t \in [0,T]\}$ a non-homogeneous Poisson process (NHPP) with cumulative intensity function $\Lambda_\xi(t)$, which has to be bounded on $[0,T]$.

The process $\{Y_t, t \in [0,T]\}$ can be a general diffusion with $h(\theta,t,y) = 0$ or the NHPP itself with $b(\phi,t,y) = s(\gamma,t,y) = y_0(\phi) = 0$ and $h(\theta,t,y) = 1$. We denote the whole parameter vector with $\vartheta = (\phi, \gamma^2, \theta, \xi)$ with $\phi \in \mathbb{R}^p, \gamma^2 \in (0,\infty), \theta \in \mathbb{R}$ and $\xi \in \mathbb{R}^q$.

We assume to observe the process discretely in time points $0 \le t_0 < t_1 < ... < t_n \le T$. In Hermann (2016), a Bayesian prediction approach based on the Euler approximation is presented. The approximated variables are given by

$$\begin{aligned} &Y_0 = y_0(\phi), \\ &Y_i = Y_{i-1} + b(\phi, t_{i-1}, Y_{i-1})\, \Delta_i + s(\gamma, t_{i-1}, Y_{i-1}) \sqrt{\Delta_i}\, \zeta_i + h(\theta, t_{i-1}, Y_{i-1}) \Delta N_i, \\ &\zeta_i \sim \mathcal{N}(0,1), \quad \Delta N_i \sim \text{Pois}\left(\Lambda_\xi(t_i) - \Lambda_\xi(t_{i-1})\right), \\ &\Delta_i = t_i - t_{i-1},\ i = 1, ..., n, \end{aligned} \tag{2}$$

see Platen and Bruti-Liberati (2010). In the following, we denote the vector of Euler approximated variables with $Y_{(n)} = (Y_0, Y_1, ..., Y_n)$.

In the next section, the estimation is shortly motivated for the considered models. Afterwards, the Bayesian prediction procedure is explained. In Section 4, the functions of the package are illustrated.

# 2 Models

## 2.1 Diffusion

In the following, an estimation procedure based on the Euler approximation of the process defined by the SDE (1) with $h(\theta,t,y) = 0$ is given. We here restrict to the simple case of estimation with the transition density of the Euler approximated variables in (2). Fuchs (2013) presents a data augmentation approach for the estimation based on the SDE in the case of large inter-observation times in the data.

The transition distribution is given in the Bayes model

$$\begin{aligned} Y_i | Y_{i-1}, \phi, \gamma^2 &\sim \mathcal{N}(Y_{i-1} + b(\phi, t_{i-1}, Y_{i-1})\, \Delta_i, s^2(\gamma, t_{i-1}, Y_{i-1}) \Delta_i),\ i = 1, ..., n \\ \phi | m_\phi, V_\phi &\sim \mathcal{N}(m_\phi, V_\phi), \quad V_\phi = \text{diag}(v_1^2, ..., v_p^2) \\ \gamma^2 | a_\gamma, b_\gamma &\sim \text{IG}(a_\gamma, b_\gamma). \end{aligned}$$

In the case of nonlinear function $b$, a Metropolis Hastings (MH) algorithm for $\phi$ is suitable, see Robert and Casella (2004). For generalization, only this case is considered. Parts of $\phi$,

which are linear in $b$, could be estimated with a conjugate prior, see, for example, Donnet et al. (2010).

In the case of a specific variance function $s(\gamma, t, y)$ in the form $s(\gamma, t, y) = \gamma \cdot \widetilde{s}(t, y)$, a full conditional posterior can be calculated. Examples are $\widetilde{s}(t, y) = 1$ with constant variance behavior or $\widetilde{s}(t, y) = t$ to have a variance structure dependent on the time or $\widetilde{s}(t, y) = y$ that leads to an autoregressive variance dependent on the current value of the process.

In this case, the full conditional posterior of $\gamma^2$ is given by

$$\gamma^2 | Y_{(n)}, \phi \sim \text{IG}\left(a_\gamma + \frac{n}{2}, b_\gamma + \frac{1}{2}\sum_{i=1}^{n} \frac{(Y_i - Y_{i-1} - b(\phi, t_{i-1}, Y_{i-1})\Delta_i)^2}{\widetilde{s}^2(t_{i-1}, Y_{i-1})\Delta_i}\right).$$

In the package, only the variance function $s(\gamma, t, y) = \gamma \cdot \widetilde{s}(t, y)$ is considered. In the case of a general function $s(\gamma, t, y)$, estimation of $\gamma^2$ could be conducted through a MH algorithm as well.

Sampling from the posterior of both parameters $\phi$ and $\gamma^2$ jointly, is performed with a Metropolis within Gibbs sampler, see Robert and Casella (2004).

## 2.2 Hierarchical diffusion model

We here consider the parameter $\phi$ to be a random effect and have one realisation for each series. We here restrict to the Gaussian mixture distribution, extensions would be future work.

For the special cases of the Ornstein-Uhlenbeck and the Cox-Ingersoll-Ross process with explicit representation, inference and prediction for the hierarchical diffusion model is implemented in the R package `mixedsde`, see Dion et al. (2016b) and explained in Dion et al. (2016a).

We define the Euler approximated Bayes model

$$\begin{aligned}
Y_{ij} | Y_{i-1,j}, \phi_j, \gamma^2 &\sim \mathcal{N}(Y_{i-1,j} + b(\phi_j, t_{i-1,j}, Y_{i-1,j})\,\Delta_{ij}, \gamma^2 \widetilde{s}^2(t_{i-1,j}, Y_{i-1,j})\Delta_{ij}),\ i = 1, ..., n_j \\
\phi_j | \mu, \Omega &\sim \mathcal{N}(\mu, \Omega) \text{ i.i.d.},\ j = 1, ..., J, \quad \Omega = \text{diag}(\omega_1^2, ..., \omega_p^2) \\
\mu | m_\mu, V_\mu &\sim \mathcal{N}(m_\mu, V_\mu), \quad V_\mu = \text{diag}(v_1^2, ..., v_p^2) \\
\omega_r^2 | a_{\omega,r}, b_{\omega,r} &\sim \text{IG}(a_{\omega,r}, b_{\omega,r}),\ r = 1, ..., p \\
\gamma^2 | a_\gamma, b_\gamma &\sim \text{IG}(a_\gamma, b_\gamma)
\end{aligned}$$

with $\Delta_{ij} = t_{ij} - t_{i-1,j}$ and $Y_{0j} = y_0(\phi_j), j = 1, ..., J$. Denote with $Y_{(n_j,j)}$ the $j$th observation vector and $Y_{(n\cdot,J)}$ all the observations $\{Y_{ij}\}_{i=1,...,n_j, j=1,...,J}$. The estimation procedure can also be found in Hermann et al. (2016b) for fixed $y_0 = y_0(\phi)$.

Since $Y_{0j}$ depends on the random effect, the likelihood is given by

$$p(Y_{(n\cdot,J)} | \phi_1, ..., \phi_J, \gamma^2) = \prod_{j=1}^{J} p(Y_{0j} | \phi_j) \prod_{i=1}^{n_j} p(Y_{ij} | Y_{i-1,j}, \phi_j, \gamma^2)$$

with $p(Y_{0j} | \phi_j) = \mathbb{1}_{y_0(\phi_j)}(Y_{0j})$.

We assume a conjugate normal prior distribution of $\mu$ with mean $m_\mu$ and diagonal variance matrix $V_\mu$. In Hermann et al. (2016b), the matrix representation of the posterior distribution can be found. Since $V_\mu$ and $\Omega$ are assumed to be diagonal in our model, we can calculate the

posterior distribution for each component of $\mu$. The full conditional posterior distribution is given by

$$\mu_r|\phi_1,...,\phi_J,\Omega \sim \mathcal{N}\left(\left(\frac{1}{v_r^2}+\frac{J}{\omega_r^2}\right)^{-1}\left(\frac{m_{\mu,r}}{v_r^2}+\sum_{j=1}^{J}\frac{(\phi_j)_r}{\omega_r^2}\right),\left(\frac{1}{v_r^2}+\frac{J}{\omega_r^2}\right)^{-1}\right),\ r=1,...,p$$

with $(\phi_j)_r$ being the $r$th component of $\phi_j$.

In the case of correlations between the parameters, i.e. non-diagonal matrix $\Omega$, one could take the Wishart prior distribution, which is also conjugate to the normal likelihood, see, for example, Donnet et al. (2010). Choosing the inverse gamma distribution for the diagonal elements, we get the conditional posterior distribution

$$\omega_r^2|(\phi_1)_r,...,(\phi_J)_r,\mu_r \sim \text{IG}\left(\alpha_{\omega,r}+\frac{J}{2},\beta_{\omega,r}+\frac{1}{2}\sum_{j=1}^{J}((\phi_j)_r-\mu_r)^2\right),\ r=1,...,p.$$

The full conditional posterior of $\gamma^2$ is similar to the single series model given by

$$\gamma^2|Y_{(n\cdot,J)},\phi_1,...,\phi_J \sim \text{IG}\left(a_\gamma+\sum_{j=1}^{J}\frac{n_j}{2},b_\gamma+\frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\frac{(Y_{ij}-Y_{i-1,j}-b(\phi_j,t_{i-1,j},Y_{i-1,j})\Delta_{ij})^2}{\widetilde{s}^2(t_{i-1,j},Y_{i-1,j})\Delta_{ij}}\right).$$

After choosing starting values $\phi_{j0}^*,\ j=1,...,J$, for the MH steps and $\mu_0^*,\ \Omega_0^*$ and $\gamma_0^{2*}$ for the Gibbs sampler, we draw for $k=1,...,K$ from

$$\begin{aligned}
\phi_{jk}^* &\sim p(\phi_j|Y_{(n_j,j)},\gamma_{k-1}^{2*},\mu_{k-1}^*,\Omega_{k-1}^*),\ j=1,...,J\\
\mu_k^* &\sim p(\mu|\phi_{1k}^*,...,\phi_{Jk}^*,\Omega_{k-1}^*)\\
\Omega_k^* &\sim p(\Omega|\phi_{1k}^*,...,\phi_{Jk}^*,\mu_k^*)\\
\gamma_k^{2*} &\sim p(\gamma^2|Y_{(n\cdot,J)},\phi_{1k}^*,...,\phi_{Jk}^*).
\end{aligned}$$

## 2.3 Hidden diffusion model

If the diffusion process is noisy observed, a hidden Markov model can be suitable. In this case, assume the Bayes model

$$\begin{aligned}
Z_i|Y_i,\sigma^2 &\sim \mathcal{N}(Y_i,\sigma^2),\ i=0,...,n\\
Y_i|Y_{i-1},\phi,\gamma^2 &\sim \mathcal{N}(Y_{i-1}+b(\phi,t_{i-1},Y_{i-1})\,\Delta_i,\gamma^2\widetilde{s}^2(t_{i-1},Y_{i-1})\Delta_i),\ i=1,...,n\\
\phi|m_\phi,V_\phi &\sim \mathcal{N}(m_\phi,V_\phi),\quad V_\phi=\text{diag}(v_1^2,...,v_p^2)\\
\gamma^2|a_\gamma,b_\gamma &\sim \text{IG}(a_\gamma,b_\gamma)\\
\sigma^2|a_\sigma,b_\sigma &\sim \text{IG}(a_\sigma,b_\sigma).
\end{aligned}$$

We introduce the additional short notation for the vector $Z_{(n)}=(Z_0,...,Z_n)$. The model fits in the class of state-space models in the literature and several filtering approaches for the latent variable $Y_{(n)}$ are available. In the package, the particle Gibbs sampler is implemented, see Andrieu et al. (2010).

The full conditional posterior distribution of $\gamma^2$ is exactly the same as in Section 2.1. The posterior of $\phi$ can be calculated by

$$\begin{aligned} p(\phi|Z_{(n)}, Y_{(n)}, \gamma^2, \sigma^2) &\propto p(\phi|Y_{(n)}, \gamma^2, \sigma^2) \cdot p(Z_{(n)}|\phi, Y_{(n)}, \gamma^2, \sigma^2) \\ &= p(\phi|Y_{(n)}, \gamma^2) \cdot p(Z_0|y_0(\phi), \sigma^2) \cdot \prod_{i=1}^{n} p(Z_i|Y_i, \sigma^2) \\ &\propto p(\phi) \cdot p(Y_{(n)}|\phi, \gamma^2) \cdot p(Z_0|\phi, \sigma^2), \end{aligned}$$

where $p(Z_0|\phi, \sigma^2)$ is the density of the $\mathcal{N}(y_0(\phi), \sigma^2)$ distribution.

Finally, with the inverse gamma prior for $\sigma^2$ we get the full conditional posterior

$$\sigma^2|Z_{(n)}, Y_{(n)} \sim \text{IG}\left(a_\sigma + \frac{n+1}{2}, b_\sigma + \frac{1}{2}\sum_{i=0}^{n}(Z_i - Y_i)^2\right).$$

Choose starting parameters $\theta_0^* = (\phi_0^*, \gamma_0^{2*}, \sigma_0^{2*})$ and conduct the particle Gibbs sampler for $k = 1, ..., K$ through

$$\begin{aligned} Y_{(n)}^{(k)} &\sim p(Y_{(n)}|Z_{(n)}, \phi_{k-1}^*, \sigma_{k-1}^{2*}, \gamma_{k-1}^{2*}) \\ \phi_k^* &\sim p(\phi|Z_{(n)}, Y_{(n)}^{(k)}, \sigma_{k-1}^{2*}, \gamma_{k-1}^{2*}) \\ \gamma_k^{2*} &\sim p(\gamma^2|Y_{(n)}^{(k)}, \phi_k^*) \\ \sigma_k^{2*} &\sim p(\sigma^2|Z_{(n)}, Y_{(n)}^{(k)}), \end{aligned}$$

where in the first iteration, $Y_{(n)}^{(1)}$ is drawn by an ordinary sequential Monte Carlo algorithm (SMC), because no fixed trajectory is given, and afterwards a conditional SMC is used, see Andrieu et al. (2010).

## 2.4 Hidden hierarchical diffusion model

This model is an extension of the hierarchical model in Section 2.2, which is noisy observed. We define the Euler approximated Bayes model

$$\begin{aligned} Z_{ij} &\sim \mathcal{N}(Y_{ij}, \sigma^2), \ i = 0, ..., n_j, j = 1, ..., J \\ Y_{ij}|Y_{i-1,j}, \phi_j, \gamma^2 &\sim \mathcal{N}(Y_{i-1,j} + b(\phi_j, t_{i-1,j}, Y_{i-1,j})\,\Delta_{ij}, \gamma^2\tilde{s}^2(t_{i-1,j}, Y_{i-1,j})\Delta_{ij}), \ i = 1, ..., n_j \\ \phi_j|\mu, \Omega &\sim \mathcal{N}(\mu, \Omega), \ j = 1, ..., J, \quad \Omega = \text{diag}(\omega_1^2, ..., \omega_p^2) \\ \mu|m_\mu, V_\mu &\sim \mathcal{N}(m_\mu, V_\mu), \quad V_\mu = \text{diag}(v_1^2, ..., v_p^2) \\ \omega_r^2|a_{\omega,r}, b_{\omega,r} &\sim \text{IG}(a_{\omega,r}, b_{\omega,r}), \ r = 1, ..., p \\ \gamma^2|a_\gamma, b_\gamma &\sim \text{IG}(a_\gamma, b_\gamma) \\ \sigma^2|a_\sigma, b_\sigma &\sim \text{IG}(a_\sigma, b_\sigma) \end{aligned}$$

with $Y_{0j} = y_0(\phi_j), j = 1, ..., J$. Denote with $Z_{(n_j,j)}$ the $j$th observation vector and $Z_{(n\cdot,J)}$ all the observations $\{Z_{ij}\}_{i=1,...,n_j, j=1,...,J}$.

Analogously to the single series model, estimation of the latent variables $Y_{(n_1,1)}, ..., Y_{(n_J,J)}$ is the crucial estimation step and will be conducted by the conditional SMC algorithm in each of the particle Gibbs sampler iterations.

Estimation for $\phi_1, ..., \phi_J$ is analogously to the single series model for each $j = 1, ..., J$ conditional on $\gamma^2$ and $\sigma^2$. Estimation of $\mu, \Omega$ and $\gamma^2$ are the same as for the hierarchical diffusion model in Section 2.2. The full conditional posterior of $\sigma^2$ is similar to the single series model given by

$$\sigma^2 | Z_{(n\cdot,J)}, Y_{(n\cdot,J)} \sim \text{IG}\left( a_\sigma + \sum_{j=1}^{J} \frac{n_j + 1}{2}, b_\sigma + \frac{1}{2} \sum_{j=1}^{J} \sum_{i=0}^{n_j} (Z_{ij} - Y_{ij})^2 \right).$$

## 2.5 Jump diffusion

In Hermann et al. (2016a), the SDE (1) with the special cases

$$b(\phi, t, y) = \phi y, \quad s(\gamma, t, y) = \gamma y, \quad h(\theta, t, y) = \theta y$$

is presented. Merton (1976) introduced this model with a homogeneous Poisson process (HPP), i.e. $\Lambda_\xi(t) = \xi t$, to model stock returns. The SDE has a unique strong solution, which is given by

$$Y_t = y_0 \cdot \exp\left( \phi t - \frac{\gamma^2}{2} t + \gamma W_t + \log(1 + \theta) N_t \right) \tag{3}$$

for $\theta > -1$, see Øksendal and Sulem (2005). As mentioned, inference and prediction for this process is done in Hermann et al. (2016a) for observed NHPP and in Hermann and Ruggeri (2016) for unobserved HPP. Inference for this process is implemented in the package separately.

Based on the Euler approximated variables in (2), the parameter vector $\vartheta = (\phi, \gamma^2, \theta, \xi)$ and definition $\Delta N_i = N_{t_i} - N_{t_{i-1}}$, we assume the Bayesian model

$$\begin{aligned}
Y_i | Y_{i-1}, \Delta N_i, \vartheta &\sim \mathcal{N}(Y_{i-1} + b(\phi, t_{i-1}, Y_{i-1})\Delta_i + h(\theta, t_{i-1}, Y_{i-1})\Delta N_i, s^2(\gamma, t_{i-1}, Y_{i-1})\Delta_i) \\
N_{t_i} &\sim \text{Pois}(\Lambda_\xi(t_i)), \ i = 0, ..., n \\
\phi &\sim p(\phi) \\
\gamma^2 &\sim p(\gamma^2) \\
\theta &\sim p(\theta) \\
\xi &\sim p(\xi).
\end{aligned}$$

In the package, arbitrary density functions can be chosen for each parameter. Conditional on the Poisson process variables $N_{(n)} = (N_{t_0}, ..., N_{t_n})$, the likelihood for the parameters $\phi, \gamma^2$ and $\theta$ is the product of normal distribution densities. For the likelihood of $\xi$, we define the event times of the Poisson process by $T_i = \min\{t : N_t = i\}, i = 1, ..., N_{t_n}$. The likelihood is given by

$$p(N_{(n)} | \xi) = e^{-\Lambda_\xi(t_n)} \cdot \prod_{i=1}^{N_{t_n}} \lambda_\xi(T_i),$$

see also Ríos Insua et al. (2012).

Estimation of the parameters $\phi, \gamma^2, \theta$ and $\xi$ is implemented with a Metropolis within Gibbs sampler, if the Poisson process is observed. Of course, for special cases of functions $b, s$ and

$h$, full conditional posteriors could be analytically available, possibly with conjugate priors. But, for generalization, this will not be done here.

In many cases, the Poisson process can not be directly observed and is, therefore, latent. In the following, we consider a filtering procedure for the process given in (1) for the Euler approximated variables, in the case of an unobserved Poisson process. One step is included in the Gibbs sampler for the estimation of $N_{(n)}$.

Define $\Delta\Lambda_{\xi,i} := \Lambda_\xi(t_i) - \Lambda_\xi(t_{i-1})$ and $\Delta Y_i = Y_{t_i} - Y_{t_{i-1}}$. We reduce the problem to the posterior of the independent differences $\Delta N_i \sim \text{Pois}(\Delta\Lambda_{\xi,i})$. It is

$$\Delta Y_i|\Delta N_i, Y_{i-1}, \phi, \gamma^2, \theta \sim \mathcal{N}\left(b(\phi, t_{i-1}, Y_{i-1})\Delta_i + h(\theta, t_{i-1}, Y_{i-1})\Delta N_i, s^2(\gamma, t_{i-1}, Y_{i-1})\Delta_i\right).$$

Therefore, the posterior density for $\Delta N_i$ is given by

$$\begin{aligned} &p(\Delta N_i|\Delta Y_i, Y_{i-1}, \phi, \gamma^2, \theta, \xi) \\ \propto &\frac{\exp(-\Delta\Lambda_{\xi,i})}{(\Delta N_i)!}(\Delta\Lambda_{\xi,i})^{\Delta N_i}\frac{1}{\sqrt{2\pi\Delta_i}s(\gamma, t_{i-1}, Y_{i-1})}\cdot \\ &\exp\left(-\frac{(\Delta Y_i - b(\phi, t_{i-1}, Y_{i-1})\Delta_i - h(\theta, t_{i-1}, Y_{i-1})\Delta N_i)^2}{2s^2(\gamma, t_{i-1}, Y_{i-1})\Delta_i}\right), \end{aligned}$$

which is not a density of a known distribution family. For all $i = 1, .., n$, sampling from the posterior is done with inversion method, see Devroye (1986) on a candidate set $\{0, ..., R\}, R \in \mathbb{N}$.

After choosing starting values, the Metropolis within Gibbs sampler unites all estimation steps.

## 2.6 Jump regression

Analogously to the jump diffusion, in Heeke et al. (2016), a regression model including a NHPP has been considered and is for this reason implemented in the package as well. The corresponding Bayes model is given by

$$\begin{aligned} Y_i &\sim \mathcal{N}(f(t_i, N_{t_i}, \theta), \gamma^2 s^2(t_i)) \\ N_{t_i} &\sim \text{Pois}(\Lambda_\xi(t_i)),\ i = 0, ..., n, \\ \theta|m_\theta, V_\theta &\sim \mathcal{N}(m_\theta, V_\theta) \\ \gamma^2|a_\gamma, b_\gamma &\sim \text{IG}(a_\gamma, b_\gamma) \\ \xi &\sim p(\xi) = 1. \end{aligned}$$

Here, $s^2(t)$ is a time dependent variance function. Conditional on the Poisson process variables $N_{(n)}$, we have a regression model, whose parameters can be estimated based on the likelihood, which is a product of normal distribution densities. Because of the possibly nonlinear regression function we have no closed form of the posterior of $\theta$. Therefore, a Metropolis within Gibbs sampler will be used, whereby

$$\gamma^2|Y_{(n)}, N_{(n)}, \theta \sim \text{IG}\left(a_\gamma + \frac{n+1}{2}, b_\gamma + \frac{1}{2}\sum_{i=0}^{n}\frac{(Y_i - f(t_i, N_{t_i}, \theta))^2}{s^2(t_i)}\right).$$

The Gibbs sampler unites the three estimation steps as follows

$$\begin{aligned}\theta_k^* &\sim p(\theta|Y_{(n)}, N_{(n)}, \gamma_{k-1}^{2*})\\ \gamma_k^{2*} &\sim p(\gamma^2|Y_{(n)}, N_{(n)}, \theta_k^*)\\ \xi_k^* &\sim p(\xi|N_{(n)}),\ k = 1, ..., K.\end{aligned}$$

It is future work to extend the estimation also to the case of unobserved Poisson process variables. A particle Gibbs sampler might be suitable.

In the package, a general nonlinear regression model and its hierarchical version are also implemented. The Bayes model and the prior distributions are analogously to the diffusion and the hierarchical diffusion model.

# 3 Prediction

In Hermann (2016), two Bayesian prediction methods are presented. One for a pointwise predictive distribution and one for a predictive distribution of the trajectory. For a better understanding, we shortly present the algorithms without going into theoretical details.

We assume to predict the process in time point $t^* \gg t_n$, resp. $t^* \gg t_0$. For large time distances, the Euler approximation can be inaccurate. Therefore, the idea of the algorithm is to go in $M$ steps to the interesting value and each include the predictive distribution of the last step. This means, in time points $\{t_0, t_n\} \ni \tau_0 < \tau_1 < ... < \tau_M = t^*$, predictions are calculated. Let $p(Y_{m+1}^*|Y_m^*, \vartheta)$ denote the transition density of the Euler approximated variables $Y_0^*, ..., Y_M^*$ in $\tau_0, ..., \tau_M$. In the specific model, the hierarchical and the jump models, the first step is to predict the latent variables, the random effect in the hierarchical model and the Poisson process in the jump models. In the following, $\vartheta_k^*, k = 1, ..., K$ will denote the vector of posterior and predictive samples jointly.

**Algorithm 1**
*Take samples $\vartheta_k^*,\ k = 1, ..., K$ from the posterior distribution $p(\vartheta|Y_{(n)})$, respectively from the predictive distribution of latent variables.*

*(i) Set $m = 1$ and $Y_0^{*(k)} := Y_n, k = 1, ..., K$ (prediction for the current series) or $Y_0^{*(k)} := y_0, k = 1, ..., K$ (prediction for a new series).*

*(ii) Draw $K$ samples*

$$Y_m^{*(r)} \sim \frac{1}{K}\sum_{k=1}^{K} p\left(Y_m^*|Y_{m-1}^{*(k)}, \vartheta_k^*\right),\ r = 1, ..., K.$$

*(iii) If $m = M$ stop*
*else $m = m + 1$ and go to step (ii).*

Sampling in step *(ii)* is done with inversion method, see Devroye (1986). This algorithm yields samples from the predictive distribution $p(Y_m^*|Y_{(n)}), m = 1, ..., M$ for each of the points $\tau_1, ..., \tau_M$, but no trajectories. The following algorithm is made for the case that trajectories are the value of interest. Here, it is possible not to draw exactly $M$ samples from the trajectory, but to choose a critical value $y_c$ and to stop if it is reached. In addition, it is possible to draw $L \neq K$ samples.

**Algorithm 2**
*Take the samples* $\vartheta_k^*,\ k = 1, ..., K$ *resulting from the Gibbs sampler displaying* $p(\vartheta|Y_{(n)})$*, respectively from the predictive distribution of latent variables. For* $l = 1, ..., L$ *repeat the following steps.*

*(i) Set* $m = 1$ *and* $Y_0^{*(l)} := Y_n$ *(prediction for the current series) or* $Y_0^{*(l)} := y_0$ *(prediction for a new series).*

*(ii) Draw one sample*

$$Y_m^{*(l)} \sim \frac{1}{K} \sum_{k=1}^{K} p\left(Y_m^* | Y_{m-1}^{*(l)}, \vartheta_k^*\right).$$

*(iii) If* $m = M$ *(or* $Y_m^{*(l)} \geq y_c$*) stop*
*else* $m = m + 1$ *and go to step (ii).*

In the package, two additional prediction approaches are implemented. At first, usual in frequentist estimation, the point estimations, i.e. the posterior mean, are plugged in the model definition and trajectories are simulated to form a prediction. At second, usual in Bayesian estimation, each sample from the posterior is plugged in and the process is simulated once.

# 4 Implementation

The package BaPreStoPro is structured as follows. For each of the models, there is one S4 model class as can be seen in the second column of Table 1.

| Model | Class object | |
|---|---|---|
| | `simulate` & `estimate` | `plot` & `predict` |
| diffusion | `Diffusion` | `est.Diffusion` |
| hierarchical diffusion | `mixedDiffusion` | `est.mixedDiffusion` |
| hidden diffusion | `hiddenDiffusion` | `est.hiddenDiffusion` |
| hierarchical hidden diffusion | `hiddenmixedDiffusion` | `est.hiddenmixedDiffusion` |
| NHPP | `NHPP` | `est.NHPP` |
| jump diffusion | `jumpDiffusion` | `est.jumpDiffusion` |
| jump diffusion (3) | `Merton` | `est.Merton` |
| jump regression | `jumpRegression` | `est.jumpRegression` |
| regression | `Regression` | `est.Regression` |
| hierarchical regression | `mixedRegression` | `est.mixedRegression` |

Table 1: Overview of the class names for the corresponding methods.

They all are constructed by the function `set.to.class` with the main input parameter `class.name`, denoting the name of the model class, and the input parameters `parameter`, `prior`, `start`, `b.fun`, `s.fun`, `h.fun`, `sT.fun`, `y0.fun`, `fun`, `Lambda` and `priorDensity`. `prior` is optional, if missing, it is calculated from the list entries of `parameter` so that mean and standard deviation of the prior distribution are equal to the values of `parameter`. Analogously, `start`, if missing, is set equal to `parameter`. `b.fun` defines the function $b(\phi, t, y)$ in all (jump) diffusion models, `s.fun` defines the function $s(\gamma^2, t, y)$, and `h.fun` the function $h(\theta, t, y)$ in the jump diffusion model. `sT.fun` stands for the variance

function $\widetilde{s}(t, y)$ in the (mixed) diffusion models as well as for $s^2(t)$ in the (mixed) regression models. `y0.fun` denotes the starting value function $y_0(\phi)$ for the hidden and mixed diffusion models. `fun` is the regression function $f(t, N_t, \theta)$ in the jump regression and $f(\phi, t)$ in the (mixed) regression model. `Lambda` is for all models, containing the NHPP, the cumulative intensity function $\Lambda_\xi(t)$. `priorDensity` is for the jump diffusion model a list of prior densities, for the Merton model and the NHPP a prior density for $\xi$, defaults are non-informative approaches.

For each of the model classes, a simulation method `simulate` is available. With the respective data set, method `estimate` can be applied to the model class with the additional input parameters `t`, i.e. the time vector, `data`, denoting the respective data set, which can be a vector, a matrix or a list in the respective model, and the number of Markov chain iterations, `nMCMC`. In addition, there is the possibility to choose the proposal standard deviation for the MH algorithms, `propSd`. If `adapt = TRUE`, this proposal standard deviation is just the starting one and is adapted after every 50 iterations, if the acceptance rate is smaller than 30% or larger than 60%, see for adaptive MCMC Rosenthal (2011). The proposal density itself can also be chosen between the normal and the log-normal distribution, `proposal = "normal"` or `"lognormal"`. Output of the `estimate` method is a new S4 class composed of `"est."` and the model class name, see Table 1.

This estimation class object is input parameter in the method `predict`. One of the main input parameters is `pred.alg` and denotes the prediction algorithm. Algorithm 1 (default) is named `"Distribution"`. Algorithm 2 is implemented with the `pred.alg = "Trajectory"`. As mentioned, two additional sampling prediction algorithms are implemented. With `pred.alg = "simpleTrajectory"`, the point estimates, i.e. the posterior mean, are calculated and based on these values, the process is just simulated. With `pred.alg = "simpleBayesTrajectory"`, the process is simulated with each posterior sample once. Second main input parameter is `which.series`, where one can decide between `"new"`, i.e. $\tau_0 = t_0$, and `"current"`, i.e. $\tau_0 = t_n$. An overview can be seen in Table 2.

| | Algorithm 1 | Algorithm 2 |
|---|---|---|
| $\tau_0 = t_0$ | `predict( , pred.alg = "Distribution", which.series = "new")` | `predict( , pred.alg = "Trajectory", which.series = "new")` |
| $\tau_0 = t_n$ | `predict( , pred.alg = "Distribution", which.series = "current")` | `predict( , pred.alg = "Trajectory", which.series = "current")` |

Table 2: Overview of the presented prediction methods in Section 3, implemented in method `predict`.

## Diffusion

We will illustrate the main methods of the package with an example. It will not be possible to show all implemented methods in one article, so we restrict here to a detailed explanation for the diffusion model. Usage for the other models is analogously. For two special cases, we show special features, for the mixed model and the jump diffusion.

**Example 1**
*We consider process* (1) *with* $h(\theta, t, y) = 0$, $s(\gamma, t, y) = \gamma y$ *and* $b(\phi, t, y) = \phi y$. *It is approxi-*

*mated by*

$$Y_i = Y_{i-1}(1 + \phi\,\Delta_i) + \gamma Y_{i-1}\sqrt{\Delta_i}\,\zeta_i,\ i = 1, ..., n$$

*with* $Y_0 = y_0$ *and* $\zeta_i \sim \mathcal{N}(0, 1)$. *We simulate random samples of the process with* $y_0 = 1, \phi = 2$, $\gamma = 1$ *and* $n = 50$.

We translate that to code:

```
b.fun <- function(phi, t, y) phi * y
sT.fun <- function(t, y) y
model <- set.to.class("Diffusion", parameter = list(phi = 2, gamma2 = 1),
  b.fun = b.fun, sT.fun = sT.fun)
t <- seq(0, 1, by = 0.01)
Y <- simulate(model, seed = 123, t = t, y0 = 1)
```

The simulated trajectory can be seen in Figure 1. There is an optional input parameter `mw`, default is 1, whose inverse serves as mesh width for simulating time-continuity of the process. For example, if `mw = 10`, nine points are equidistantly added between each two points of `t` and the process is simulated on the finer time grid. Afterwards, every tenth simulated point is given out as simulated series. We here restrict to `mw = 1` to avoid biased estimations.

Estimation is done by the method `estimate` as follows:

```
est <- estimate(model, t, Y, 21000)
plot(est)
```

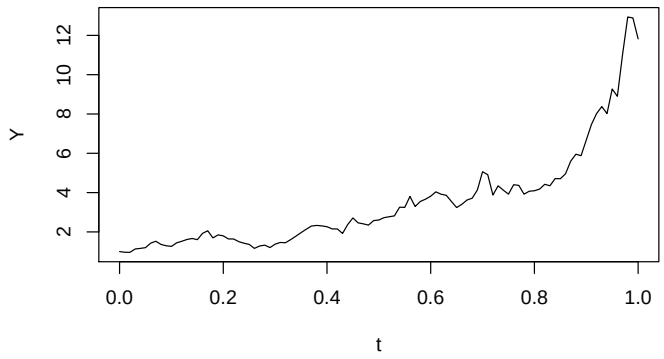



Figure 1: Trajectory created with method `simulate` for the diffusion model.

Output of the method `estimate` for class object `Diffusion` is a new class object `est.Diffusion`, which contains all information of the model as well as the data and the estimation results. In addition, it contains a proposal for the thinning rate and the burn-in phase. Both are input parameters in the following methods, but if they are missing, the proposed values are taken.

The proposed `burnIn` is calculated by dividing the Markov chains into 10 blocks and calculate the 95% credibility intervals and the respective mean. Starting in the first one, the block

is taken as burn-in as long as the mean of the current block is not in the credibility interval of the following block or vice versa. The thinning rate is proposed by the smallest lag, which leads to a chain autocorrelation of less than 80%. It is not easy to automate these choices, so it is highly recommended by the author to verify the chains manually.

The Markov chains are visualized with method `plot`, which has several options for input parameter `style`, namely `chains, acf` and `density`. The first one can be seen in Figure 2, the second one shows the autocorrelation functions for the chains and the third one the resulting posterior densities. Another important input parameter is `reduced`. If it is set to `TRUE`, the chains are reduced by the burn-in phase and the thinning rate, both optional input parameters.

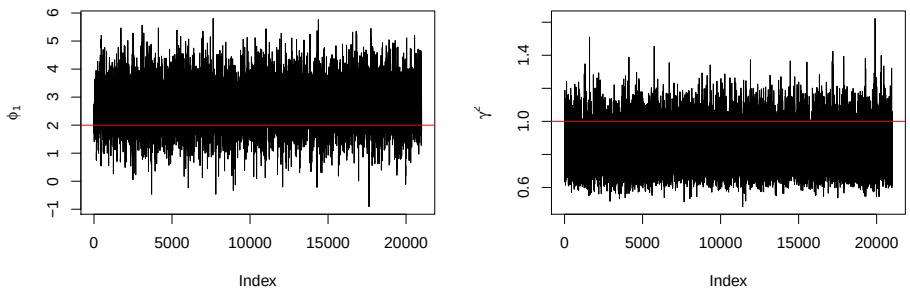



Figure 2: Markov chains produced by method `estimate` and visualized with method `plot` for the diffusion process in Figure 1.

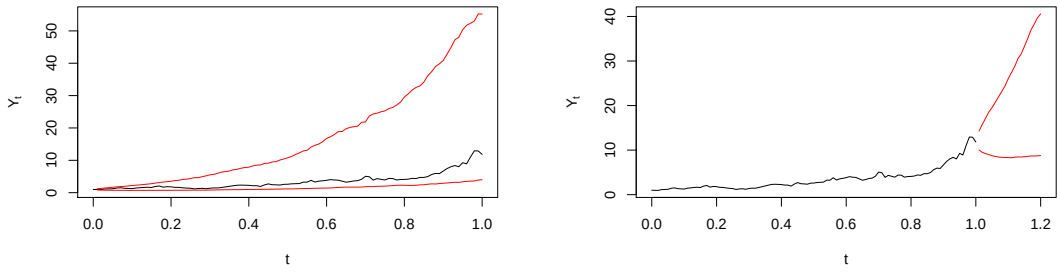
Class object `est.Diffusion` is used for method `predict`. Thinning rate and burn-in phase are also input parameters in the method `predict`.

```
burnIn <- 1000
thinning <- 2
pred <- predict(est, burnIn = burnIn, thinning = thinning,
  b.fun.mat = function(phi, t, y) phi[,1]*y)
```

Since the function `b.fun` is stored in the `est.Diffusion` object, the input parameter `b.fun.mat` is not necessary but the algorithm gets much faster with the matrix-wise function definition. If `plot.prediction = TRUE`, the 1-`level` prediction intervals are plotted with the observation series used for estimation. This leads to Figure 3(a). Default value for the input parameter `pred.alg`, i.e. the prediction algorithm, is `"Distribution"`, which denotes the pointwise prediction Algorithm 1.

One important input parameter of method `predict` is `which.series`. Default is `"new"`, which leads to prediction of a new series, starting in the first point of the observation series. The second option is `"current"`, which yields a prediction of the further development of the observed series. If no vector of time points is specified (also an input parameter, `t`), the input variable `M2pred` can be used (default: 10), which is the number of points that will be predicted with the median of the observation time distances. This can be run as follows:

```
pred.current <- predict(est, b.fun.mat = function(phi, t, y) phi[,1]*y,
  which.series = "current", M2pred = 20)
```




(a) `which.series = "new"` (b) `which.series = "current"`

Figure 3: 95% prediction intervals of method `predict`, visualized, if input parameter `plot.prediction` is `TRUE` (default).

The result can be seen in Figure 3(b). Default values for the starting points of the prediction is for `which.series = "new"` the starting point of the observed series, and for `which.series = "current"` the last observation point. If desired, this can be changed with the input parameter `y.start`, which denotes the point $Y_0^{*(k)}, k = 1, ..., K$.

Algorithm 2 is implemented with the input parameter `pred.alg = "Trajectory"`. Here, it could also be desired to choose $L \neq K$, implemented with `sample.length`. Default is $L = K$. In Hermann (2016), the two presented prediction methods are compared to two commonly used sampling procedures. For the first one, `pred.alg = "simpleTrajectory"`, the point estimates, i.e. the posterior mean, are plugged in the model definition and `sample.length`, default is the number of posterior samples $K$, trajectories are drawn. For the second one, `pred.alg = "simpleBayesTrajectory"`, each of the $K$ posterior samples is plugged in the model definition and one trajectory is drawn. If `sample.length` is specified and smaller than $K$, the first `sample.length` posterior samples are taken.

There is one additional feature of method `predict`. For comparison, a one step Euler prediction is implemented, which would be similar to Algorithm 1 with $M = 1$. In that case, sampling is not necessary and quantiles can be calculated directly. With `Euler.interval = TRUE`, for each of the time points, `level`/2 and 1-`level`/2 quantiles of the predictive distribution are calculated.

## Mixed diffusion

We simulate a model similar to Example 1, but in a mixture model. Here, we can also assume the starting point to be random, i.e. $y_0(\phi) = \phi_2$ and $b(\phi, t, y) = \phi_1 y$. Now, for similar model as before, we choose $\mu = (2, 1)$ and $\Omega = \text{diag}(1, 0.04)$. We sample $J = 50$ series as follows.

```
J <- 50
mu <- c(2, 1)
Omega <- c(1, 0.04)
phi <- cbind(rnorm(J, mu[1], sqrt(Omega[1])), rnorm(J, mu[2], sqrt(Omega[2])))
gamma2 <- 1
y0.fun <- function(phi, t) phi[2]
b.fun <- function(phi, t, y) phi[1] * y
```

```
sT.fun <- function(t, y) y

model <- set.to.class("mixedDiffusion", y0.fun = y0.fun,
  b.fun = b.fun, sT.fun = sT.fun,
  parameter = list(phi = phi, mu = mu, Omega = Omega, gamma2 = gamma2))

t <- seq(0, 1, by = 0.01)
series <- simulate(model, seed = 123, t = t, plot.series = FALSE)
```

The variable `series` is a $J \times 101$ matrix. To estimate with the whole data set, this can be also input for the parameter `data` for the method `estimate`. In this case, all series are equally long, i.e. $n_1 = ... = n_J$ and all series are observed in the same time points. Otherwise, it is possible to include a list.

```
t.list <- lapply(1:(J-1), function(i) t)
t.list[[J]] <- t[1:50]
data.list <- lapply(1:(J-1), function(i) series[i,])
data.list[[J]] <- series[J, 1:50]
est <- estimate(model, t.list, data.list, 11000)
```

For the hierarchical models, there is an additional `style` option, called `"int.phi"` meaning the credible intervals for the random effects. The point in the middle of the intervals mark the posterior median. In addition, input parameter `par2plot`, which is a logical vector, contains `TRUE` for every parameter to be plotted and `FALSE` otherwise. For comparison with starting or prior values or, in a simulation study with the chosen values, `phi` itself can be included.

```
cr <- plot(est, style = "int.phi", phi = phi, par2plot = c(T, F))
legend("bottomleft", c("true value", "posterior median", "95% credible interval"),
  col = c(2, 1, 1), lty = c(-1, -1, 1), pch = c(20, 20, -1), cex = 0.7)
sum(cr[[1]][1,] <= phi[,1] & cr[[1]][3,] >= phi[,1])
```

The result can be seen in Figure 4. Except one, all credible intervals include the simulated values.

Now, we want to make a prediction for the further development of the last, i.e. the $J$th, series.

```
pred <- predict(est, t = t[50:101], which.series = "current", ind.pred = J,
  b.fun.mat = function(phi, t, y) phi[,1]*y,
  burnIn = 1000, thinning = 2)
lines(t[51:101], series[J, 51:101], lty = 2)
```

We can see the result in Figure 5. The solid black line marks the part of the $J$th series that is used for estimation. The dotted line is the simulation part that is skipped in the estimation and prediction is made for. The red lines are the 95% prediction intervals. Output of the method `predict` is a $5\,000 \times 52$ matrix containing 5 000 samples of the predictive distribution in each of the 52 time points.

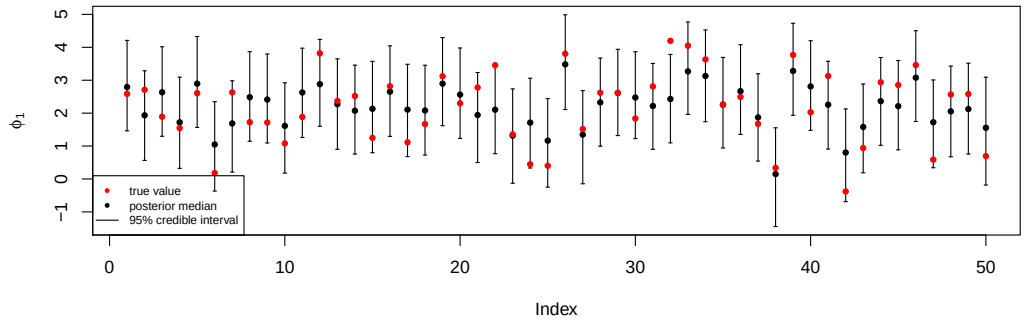



Figure 4: 95% credible intervals for the first component of the random effect in the mixed model, created with `plot(..., style = "int.phi")`.

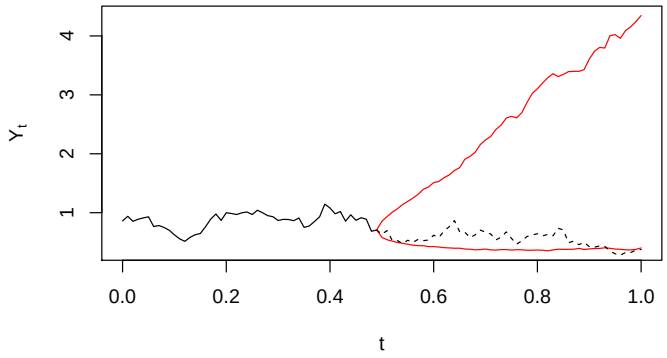



Figure 5: 95% prediction intervals for the last series of the mixed diffusion model.

## Jump diffusion

One additional example will be given by a jump diffusion. This model is only implemented for the case $\phi \in \mathbb{R}$ as one-dimensional parameter.

**Example 2**
*We consider the process* (1) *with* $b(\phi, t, y) = \phi$, $s(\gamma, t, y) = \gamma$, $h(\theta, t, y) = \theta y$ *and* $\Lambda_\xi(t) = \left(\frac{t}{\xi_2}\right)^{\xi_1}$. *It is approximated by*

$$Y_i = Y_{i-1} + \phi\,\Delta_i + \gamma\sqrt{\Delta_i}\,\zeta_i + \theta Y_{i-1}\Delta N_i,\ \ i = 1, ..., n$$

*with* $Y_0 = y_0$, $\zeta_i \sim \mathcal{N}(0,1)$ *and* $\Delta N_i \sim Pois(\Lambda_\xi(t_i) - \Lambda_\xi(t_{i-1}))$. *We fix* $y_0 = 0.5, \phi = 0.2, \gamma = 0.5, \theta = 0.1, \xi = (2, 0.2)$.

We translate the model to the language of the package:

```
b.fun <- function(phi, t, y) phi
s.fun <- function(gamma2, t, y) sqrt(gamma2)
h.fun <- function(theta, t, y) theta * y
Lambda <- function(t, xi) (t/xi[2])^xi[1]
model <- set.to.class("jumpDiffusion",
  parameter = list(phi = 0.2, gamma2 = 0.25, theta = 0.1, xi = c(2, 0.2)),
  b.fun = b.fun, s.fun = s.fun, h.fun = h.fun, Lambda = Lambda)
t <- seq(0, 1, by = 0.01)
series <- simulate(model, seed = 123, t = t, y0 = 0.5, plot.series = FALSE)
```

In the jump diffusion model, there are two possibilities: firstly, the Poisson process is not observed. This would be implemented by

```
est.hidden <- estimate(model, t, series$Y, 11000)
plot(est.hidden, par2plot = c(rep(F, 5), T), reduced = T)
lines(t, series$N, lwd = 2, col = 2)
legend("topleft", c("filtered", "simulated"), col = 1:2, lwd = 1:2)
```

As mentioned, method `plot` has input parameter `par2plot`, which contains `TRUE` for every parameter to be plotted and `FALSE` otherwise. The order for the jump diffusion model is $(\phi, \theta, \gamma^2, \xi, N_{(n)})$. This leads to Figure 6, which compares the filtered trajectories of the Poisson process with the simulated series in red.

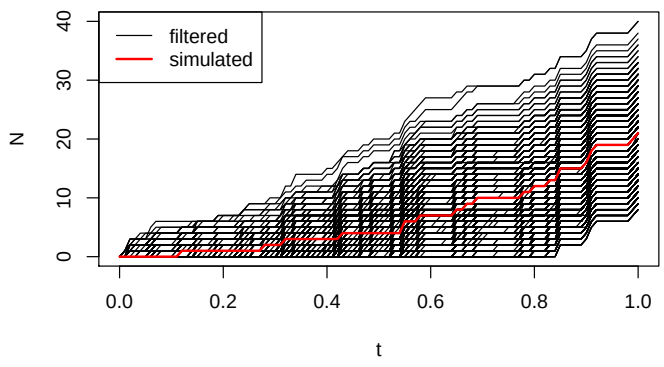



Figure 6: Estimation of the latent Poisson process variable in the jump diffusion model. In red: the simulated series.

Secondly, in the case of observed Poisson process variables, both observation vectors $Y_{(n)}$ and $N_{(n)}$ have to be joint in a list with variables `Y` and `N`.

For the prediction of the jump diffusion with the presented methods in Section 3, it can be decided for each of the processes, the latent Poisson process as well as for the jump diffusion, if Algorithm 1 or 2 is desired. We have already seen the input parameter `pred.alg`, which is here for the prediction of the jump diffusion. The additional input parameter `pred.alg.N` denotes the prediction algorithm for the Poisson process and can be chosen between `"Distribution"` and `"Trajectory"`. In the first case, pointwise sampling from the predictive distribution

of the independent differences is conducted. In the second case, Algorithm 2 is run for the event times of the process. Default is `"Trajectory"`, since the combination `pred.alg = "Trajectory"` and `pred.alg.N = "Distribution"` does not make sense and is avoided to be accidentally chosen. The parameter `Lambda.mat` again is, similar to `b.fun.mat` in the diffusion model, to fasten the sampling algorithm.

# 5 Data example

Sixty-eight replicate constant amplitude tests in aluminum alloy were carried out to investigate the fatigue crack propagation. In each of these tests, the number of cycles that leads to fixed crack lengths, was observed. See for details Virkler et al. (1979). Against the natural assumption that something is observed at fixed times, here the time is the dependent and the crack length the independent variable. Therefore, from the crack length will be treated as time vector t.

The Virkler data comes as a data frame of 164 rows and 69 columns where the first column contains the crack lengths (in mm) and the 68 following the series of observed times in load cycles up to a fixed crack length.

In Hermann et al. (2016b), a hierarchical diffusion model has been applied to the data and the most suitable function turned out to be $b(\phi, t, y) = \frac{\phi_2}{t}(\phi_1 - y)$. We here start with the starting parameter for $\mu$ given in Hermann et al. (2016b) and take the first 10 series for a pre-estimation. In the mentioned article, a constant variance function is assumed. We here try an autoregressive variance structure. For that reason, the process may not start in 0 and the first point of the series is skipped.

```
data("Virkler")

Y <- t(Virkler[-1,-1]/10000)
t <- Virkler[-1,1]
J <- nrow(Y)

b.fun <- function(phi, t, y) phi[2]/t * (phi[1]-y)
sT.fun <- function(t, y) y

mu <- c(25, 1.8)
Omega <- c(1, 0.1)
gamma2 <- 0.1

model_pre <- set.to.class("mixedDiffusion", b.fun = b.fun, sT.fun = sT.fun,
  parameter = list(phi = matrix(mu, 10, 2, byrow = TRUE),
  mu = mu, Omega = Omega, gamma2 = gamma2))
K <- 21000
est_pre <- estimate(model_pre, t, Y[1:10,], K)   # pre-estimation
```

We define the model class new based on the posterior means.

```
mu <- apply(est_pre@mu[seq(1001, K, by = 10),], 2, mean)
Omega <- apply(est_pre@Omega[seq(1001, K, by = 10),], 2, mean)
```

```
gamma2 <- mean(est_pre@gamma2[seq(1001, K, by = 10)])

model <- set.to.class("mixedDiffusion", b.fun = b.fun, sT.fun = sT.fun,
  parameter = list(phi = matrix(mu, J, 2, byrow = TRUE),
  mu = mu, Omega = Omega, gamma2 = gamma2))
```

As mentioned, since we do not define prior and starting values, the prior parameter are chosen so that mean and standard deviation are equal to the values defined in `parameter` and the starting values as well. Since we have good prior knowledge through the pre-estimation, we could also define smaller prior variances to strengthen the prior influence.

For comparison to the chosen values above, the posterior means are $\widehat{\mu} = (28.6, 1.04), \widehat{\Omega} = (11.985, 0.078)$ and $\widehat{\gamma}^2 = 5.63 \cdot 10^{-4}$, which differ widely from the starting values and the prior means from the beginning.

In addition, we assume the last series to be observed up to the 100th point and want to make predictions for the further development of the current series.

```
J <- J-10
Y <- Y[-(1:10),]

t.list <- lapply(1:(J-1), function(i) t)
t.list[[J]] <- t[1:100]
data.list <- lapply(1:(J-1), function(i) Y[i,])
data.list[[J]] <- Y[J, 1:100]

est_new <- estimate(model, t.list, data.list, K)

pred_new <- predict(est_new, t = t[100:163], which.series = "current",
  ind.pred = J, burnIn = 1000, thinning = 10,
  b.fun.mat = function(phi, t, y) phi[,2]/t * (phi[,1]-y))
lines(t[101:163], Y[J, 101:163], lty = 2)
```

In Figure 7, we see the 95% prediction intervals, which contain the corresponding observed values.

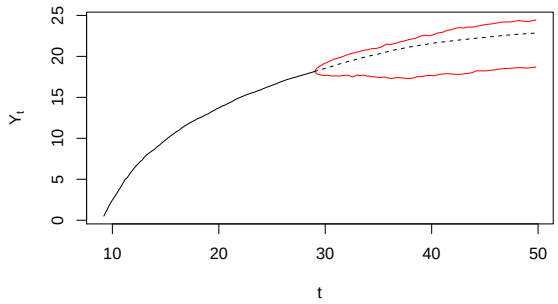



Figure 7: Prediction intervals for the last series of the Virkler data (in red), dotted lines: the predicted observations.

## 6 Discussion

The presented package BaPreStoPro yields a general tool box for estimation and prediction of stochastic processes driven by a Brownian motion and a non-homogeneous Poisson process, defined by their stochastic differential equation. Arbitrary drift, variance and jump high functions can be applied.

Currently, estimation is based on the likelihood dependent on the observation variables. If there are different, and partially large, time distances, the Euler approximation can become inaccurate. For continuous diffusions, Fuchs (2013) proposes a data augmentation approach for the estimation based on the SDE in the case of large inter-observation times in the data. This could enrich the estimation procedure of the package.

It is also thinkable to include other mixture distributions than the Gaussian for the hierarchical models or even a Bayesian nonparametric estimation approach for the mixture density. In addition, for the diffusion models, the location parameters are always assumed to have a normal prior distribution. This could be extended by a general choice of the user as done for the jump diffusion model, which also allows for a non-informative approach.

Currently, the whole package is implemented in R. In some models, as for example the hidden diffusion models with estimation based on a filtering algorithm, running times become large for a reliable evaluation, which means a high number of Markov chain iterations. It will be future work to outsource parts of the implementation to other languages like C.

## Acknowledgement

This work has been supported by the Collaborative Research Center "Statistical modeling of nonlinear dynamic processes" (SFB 823) of the German Research Foundation (DFG) in project B5 "Statistical methods for damage processes under cyclic load".

The author thanks Eric J. Tuegel for providing the data that were collected by Prof. B. M. Hillberry, published in Virkler et al. (1979).

In addition, the author wants to thank Adeline Leclercq Samson for the helpful discussions about the particle Gibbs sampler.

## References

Andrieu, C., A. Doucet, and R. Holenstein (2010). "Particle Markov Chain Monte Carlo Methods". *Journal of the Royal Statistical Society B* 72.3, pp. 269–342.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. New York: Springer.

Dion, C., S. Hermann, and A. Samson (2016a). "Mixedsde: an R Package to Fit Mixed Stochastic Differential Equations". hal-01305574.

Dion, C., A. Samson, and S. Hermann (2016b). *mixedsde: Estimation Methods for Stochastic Differential Mixed Effects Models*. R package version 1.0.

Ditlevsen, S. and A. de Gaetano (2005). "Mixed Effects in Stochastic Differential Equation". *REVSTAT - Statistical Journal* 3, pp. 137–153.

Donnet, S., J.-L. Foulley, and A. Samson (2010). "Bayesian Analysis of Growth Curves Using Mixture Models Defined by Stochastic Differential Equations". *Biometrics* 66, pp. 733–741.

Fuchs, C. (2013). *Inference for Diffusion Processes*. 1st ed. Berlin Heidelberg: Springer.

Genon-Catalot, V., T. Jeantheau, and C. Larédo (2000). "Stochastic Volatility Models as Hidden Markov Models and Statistical Applications". *Bernoulli* 6, pp. 1051–1079.

Heeke, G., S. Hermann, R. Maurer, K. Ickstadt, and C. H. Müller (2016). "Stochastic Modeling and Statistical Analysis of Fatigue Tests on Prestressed Concrete Beams under Cyclic Loadings". *SFB 823 discussion paper 25/15*.

Hermann, S. (2016). "Bayesian Prediction for Stochastic Processes based on the Euler Approximation Scheme". *SFB 823 discussion paper 27/16*.

Hermann, S. and F. Ruggeri (2016). "Modelling Wear in Cylinder Liners". *SFB 823 discussion paper 06/16*.

Hermann, S., K. Ickstadt, and C. H. Müller (2016a). "Bayesian Prediction for a Jump Diffusion Process with Application to Crack Growth in Fatigue Experiments". *SFB 823 discussion paper 30/15*.

Hermann, S., K. Ickstadt, and C. H. Müller (2016b). "Bayesian Prediction of Crack Growth Based on a Hierarchical Diffusion Model". *Applied Stochastic Models in Business and Industry*. DOI: 10.1002/asmb.2175.

Merton, R. C. (1976). "Option Pricing When Underlying Stock Returns are Discontinuous". *Journal of Financial Economics* 3, pp. 125–144.

Øksendal, B. and A. Sulem (2005). *Applied Stochastic Control of Jump Diffusions*. Berlin Heidelberg: Springer.

Platen, E. and N. Bruti-Liberati (2010). *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. First. Berlin Heidelberg: Springer.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria.

Ríos Insua, D., F. Ruggeri, and M. P. Wiper (2012). *Bayesian Analysis of Stochastic Process Models*. United Kingdom: John Wiley & Sons, Ltd.

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods*. 2nd ed. New York: Springer.

Rosenthal, J. S. (2011). "Optimal Proposal Distributions and Adaptive MCMC". Chap. Handbook of Markov Chain Monte Carlo, pp. 93–112.

Virkler, D. A., B. M. Hillberry, and P. K. Goel (1979). "The Statistical Nature of Fatigue Crack Propagation". *Journal of Engineering Materials and Technology* 101, pp. 148–153.