CRAN packages comparison

1 DiffusionRgqd

Uses the cumulant truncation procedure developed by Varughese (2013), whereby the transition density can be approximated over arbitrarily large transition horizons for a suitably general class of non-linear diffusion models.

Generalized quadratic diffusions (GQD) are the specific class of SDEs with quadratic drift and diffusion terms:

$$\begin{aligned} dX_t &= \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \text{ where} \\ \mu(X_t, t) &= G_0(t) + G_1(t)X_t + G_2(t)X_t^2, \text{ and} \\ \sigma(X_t, t) &= Q_0(t) + Q_1(t)X_t + Q_2(t)X_t^2 \end{aligned}$$

For purposes of inference the drift and diffusion terms - and consequently the transitional density - are assumed to be dependent on a vector of parameters, θ. For example, an Ornstein-Uhlenbeck model with SDE:

$$dX_t = \theta_1(\theta_2 - X_t) + \sqrt{\theta_3^2}dW_t \tag{1}$$

```
G0=function(t){theta[1]*theta[2]}
G1=function(t){-theta[1]}
Q0=function(t){theta[3]*theta[3]}
```

1.1 Constant drift, diffusion SDE

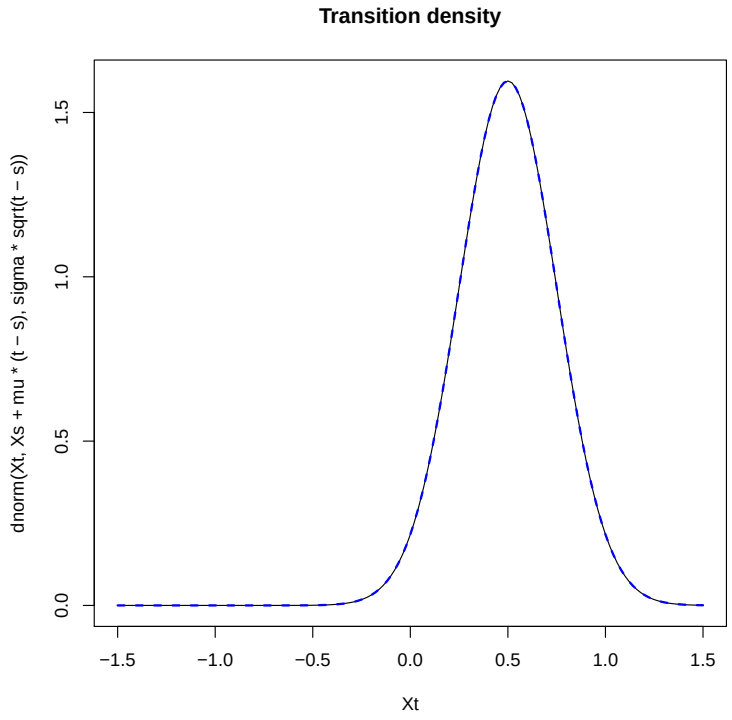
For a constant drift, diffusion SDE, with given initial condition X_s:

$$dX_t = \mu dt + \sigma dW_t \tag{2}$$

The distribution at time t of the process X_t is $\mathcal{N}(X_t, X_s + \mu(t-s), \sigma^2(t-s))$

```
library('DiffusionRgqd')

# Remove any existing coefficients
GQD.remove()

Xs <- 0                 # Initial state
Xt <- seq(-3/2,3/2,1/50)# Possible future states
s  <- 0                 # Starting time
t  <- 1                 # Final time
mu    <- 0.5            # Drift parameter
sigma <- 0.25           # Diffusion coefficient

# Define the model coefficients
G0 <- function(t){mu}
Q0 <- function(t){sigma^2}
```

```r
# Calculate the transitional density
BM <- GQD.density(Xs,Xt,s,t)

# Plot the transitional density
plot(dnorm(Xt, Xs+mu*(t-s), sigma*sqrt(t-s))~Xt, main = 'Transition density', type = 'l')
lines(BM$density[,100]~BM$Xt, col = 'blue', lty = 'dashed', lwd = 2)
```



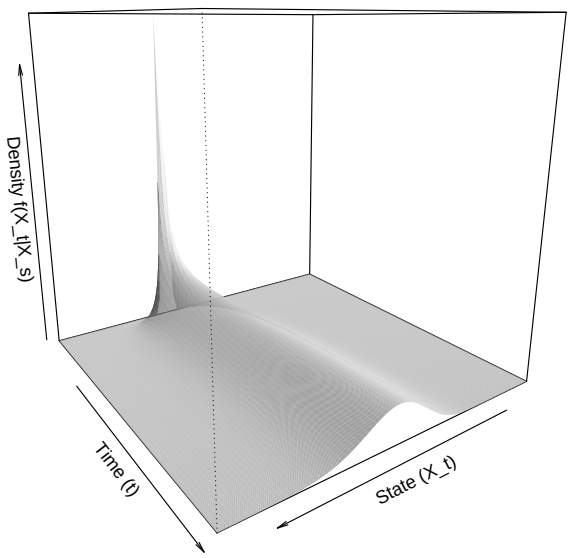
1.2 CIR process

Another example using the CIR process SDE:

$$dX_t = \theta_1(\theta_2 - X_t)dt + \theta_3\sqrt{X_t}dW_t \tag{3}$$

```r
GQD.remove()
a = 0.5; b = 5; sigma = 0.35; # Parameter values

G0 <- function(t){a*b}
G1 <- function(t){-a}
Q1 <- function(t){sigma^2}

states     <-  seq(1, 9, 1/10)# State values
initial    <-  6              # Starting value of the process
Tmax       <-  5              # Time horizon
Tstart     <-  1              # Time starts at 1
increment  <-  1/100          # Incremental time steps
```

```r
# Generate the transitional density
M <- GQD.density(Xs = initial, Xt = states, s = Tstart, t = Tmax, delt = increment)

persp(x = M$Xt, y = M$time, z = M$density, col = 'white', xlab = 'State (X_t)',ylab
 = 'Time (t)', zlab = 'Density f(X_t|X_s)', border = NA, shade = 0.5, theta = 145)
```

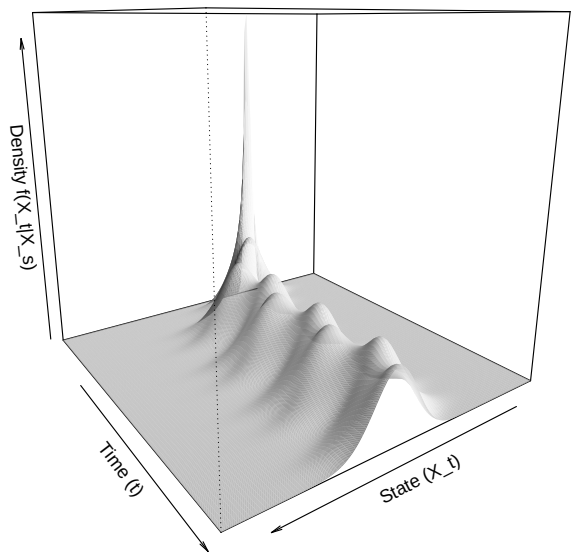


1.3 Time dependent CIR process

Consider the time-inhomogeneous CIR process for a more complicated example

$$dX_t = 2(10 + \sin(2\pi(t - 0.5)) - X_t)dt + sqrt0.25(1 + 0.75\sin(4\pi t))X_t dW_t \tag{4}$$

```r
library(DiffusionRgqd)
GQD.remove()
G0 <- function(t){2*(10+sin(2*pi*(t-0.5)))}
G1 <- function(t){-2}
Q1 <- function(t){0.25*(1+0.75*(sin(4*pi*t)))}

states    <- seq(5, 15, 1/10)
initial   <- 8
Tmax      <- 5
Tstart    <- 1
increment <- 1/100

M <- GQD.density(Xs = initial, Xt = states, s = Tstart, t = Tmax, delt = increment)
persp(x = M$Xt, y = M$time, z = M$density, col = 'white', xlab = 'State (X_t)', ylab = 'Time
```



1.4 Coupled SDEs - Prey predator model

A model that is often used to illustrate non-linear dynamics in the analysis of ODEs is that of the Lotka-Volterra model. The equations are often used to describe the dynamics of two interacting populations wherein the population growth rate of the populations are mutually influenced by the current level of the opposing population. As such the model has been used to explain oscillatory behaviour in predator-prey relationships Hoppensteadt2006 where xtxt denotes the prey population and ytyt the predator population at time t. Continuing with the predator-prey metaphor, perhaps one deficiency of the model, one might argue, is the absence of random input and subsequent effects on population levels. Indeed, under the ODE formulation the predicted population behaviour (given fixed parameters) are completely deterministic. Another deficiency might be the absence of growth inhibiting factors such as disease or over-grazing. For these purposes we may define an example of a stochastic counterpart to the Lotka-Volterra equations as:

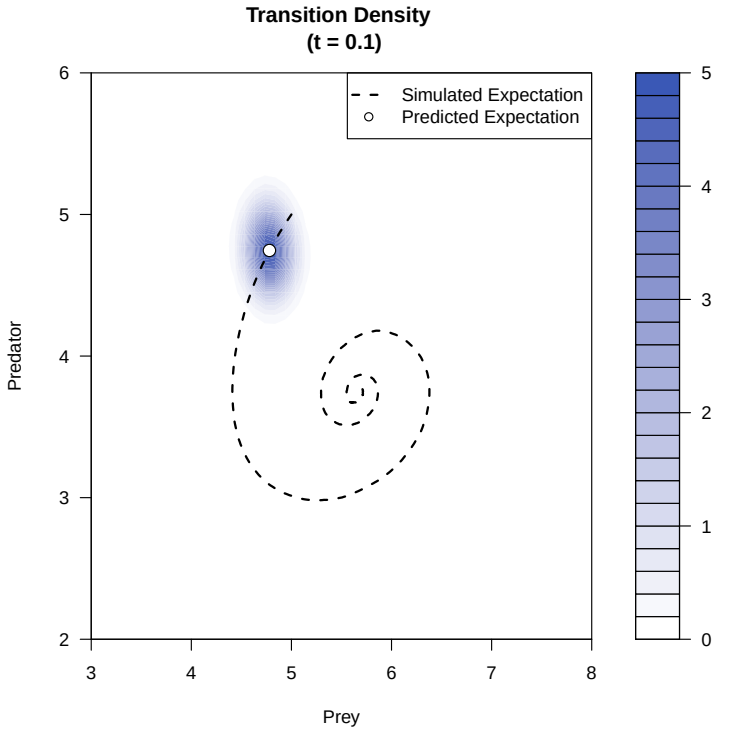
$$dX_t = (aX_t - bX_tY_t)dt + f\sqrt{X_t}dW_t^1$$
$$dY_t = (-cY_t + dX_tY_t - eY_t^2)dt + g\sqrt{Y_t}dW_t^2$$

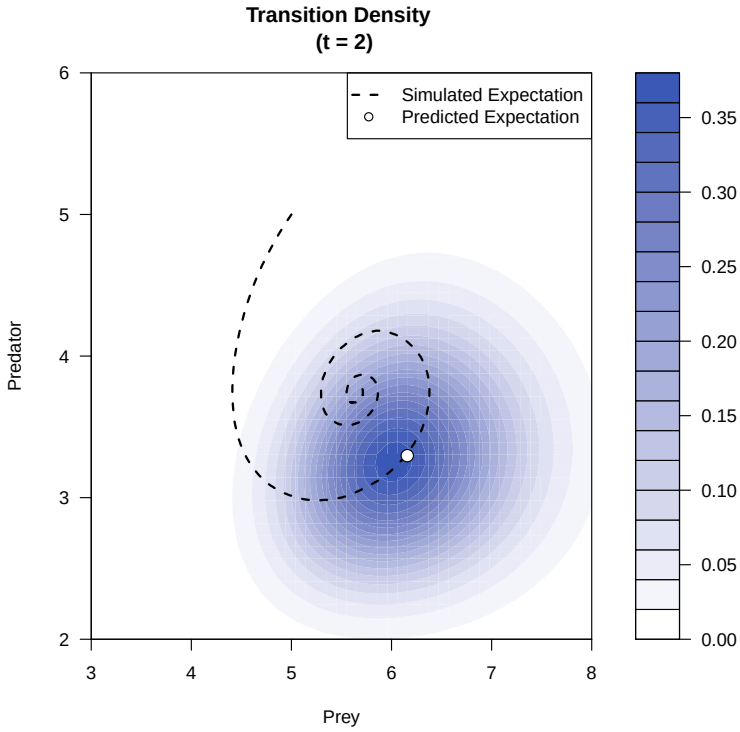
```
library(DiffusionRgqd)
# Remove any existing coefficients:
GQD.remove()

## [1] "Removed :  G0 G1 Q1"

# Define the X dimesnion coefficients:
a10 <- function(t){1.5}
```

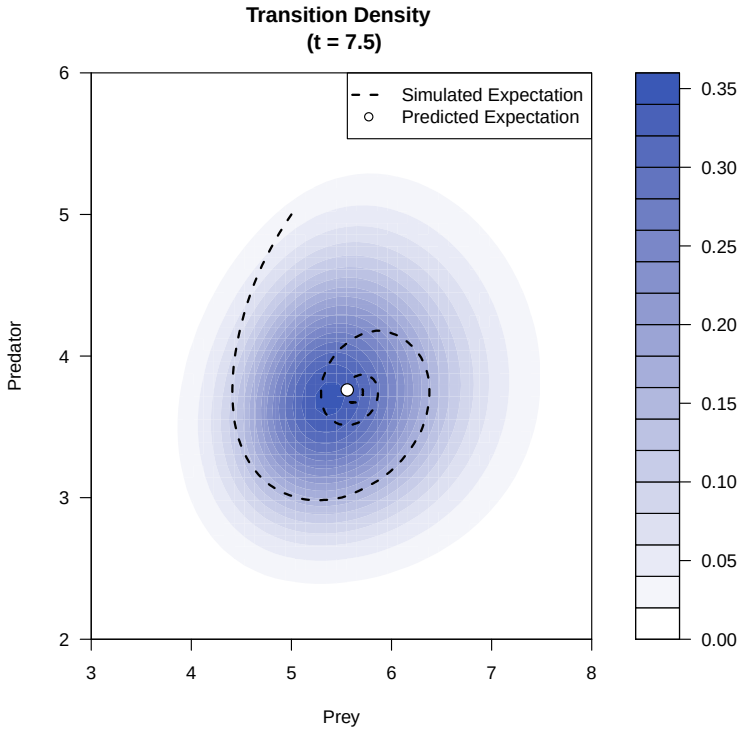
```r
a11 <- function(t){-0.4}
c10 <- function(t){0.05}
# Define the Y dimension coefficients:
b01 <- function(t){-1.5}
b11 <- function(t){0.4}
b02 <- function(t){-0.2}
f01 <- function(t){0.1}
# Approximate the transition density
res <- BiGQD.density(Xs = 5, Ys = 5, Xt = seq(3, 8, length = 50), Yt = seq(2, 6, length = 50)
```

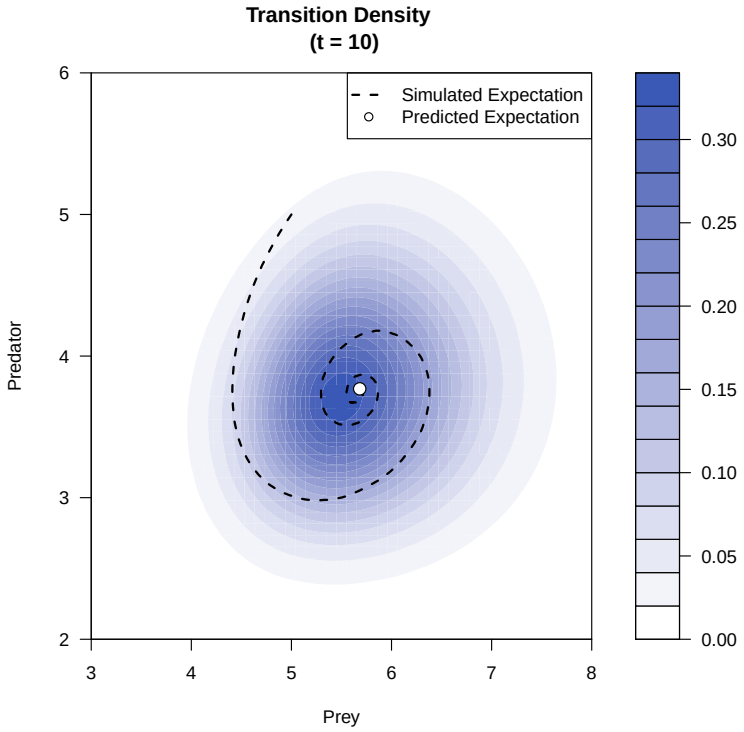
```
##
##  ================================================================
##                     GENERALIZED QUADRATIC DIFFUSON
##  ================================================================
##  ______________________ Drift Coefficients ________________________
##  a10 : 1.5
##  a11 : -0.4
##  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
##  b01 : -1.5
##  b02 : -0.2
##  b11 : 0.4
##  ____________________ Diffusion Coefficients ______________________
##  c10 : 0.05
##  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
##  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
##  ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
##  f01 : 0.1
## ================================================================
```

```
## Loading required package:  methods
```


Transition Density
(t = 0.1)
Simulated Expectation
Predicted Expectation
Predator
Prey
6
5
4
3
2
3
4
5
6
7
8
5
4
3
2
1
0


Transition Density
(t = 2)
Simulated Expectation
Predicted Expectation
Predator
Prey
6
5
4
3
2
3
4
5
6
7
8
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00



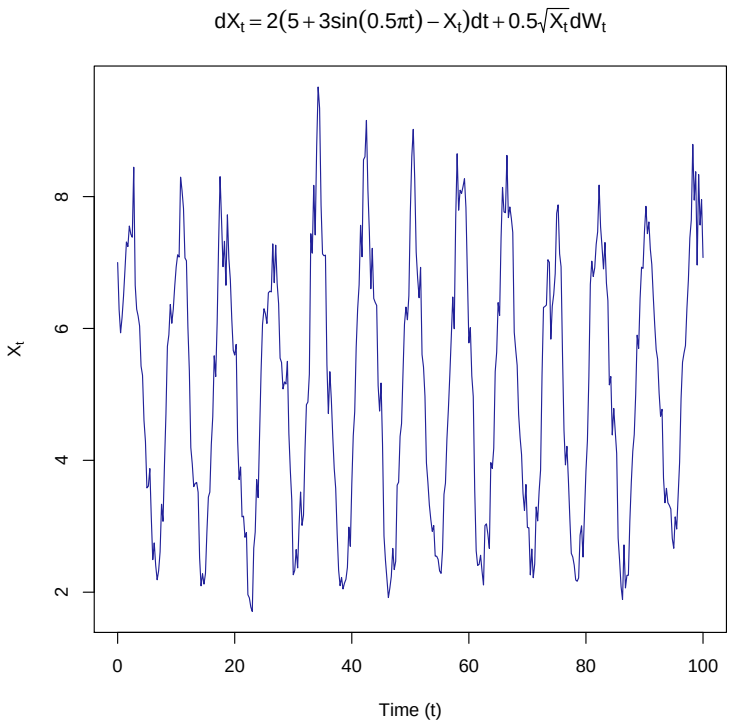


1.5 Inference on Diffusion Processes

```
library("DiffusionRgqd")
data(SDEsim1)
attach(SDEsim1)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     Xt
## The following object is masked from SDEsim3:
##
##     time
```

```
par(mfrow=c(1,1))
expr1=expression(dX[t]==2*(5+3*sin(0.5*pi*t)-X[t])*dt+0.5*sqrt(X[t])*dW[t])
plot(SDEsim1$Xt~SDEsim1$time, type = 'l', col = '#222299', xlab = 'Time (t)', ylab = expressi
```




```
GQD.remove()

G0 <- function(t){theta[1]*(theta[2]+theta[3]*sin(0.25*pi*t))}
G1 <- function(t){-theta[1]}
Q1 <- function(t){theta[4]*theta[4]}

theta   <- c(1, 10, 1, 1)               # Starting values for the chain
sds     <- c(0.25, 0.25, 0.2, 0.05)/1.5 # Std devs for proposal distributions
mesh    <- 10                           # Number of mesh points
updates <- 110000                       # Perform 110000 updates
burns   <- 10000                        # Burn 10000 updates

# Run the MCMC procedure for the model defined above:
model_1 <- GQD.mcmc(SDEsim1$Xt, SDEsim1$time, mesh, theta, sds, updates, burns)
```

The estimates are given by

```
GQD.estimates(model_1, thin = 100, burns = 10000, corrmat = TRUE)
```

2 pomp: statistical inference for partially-observed Markov processes

POMP focuses on partially observed Markov process models or hidden Markov models. They consist of an observed process Y_n from the system and an unobserved process X_n controlled by a stochastic process. The stochastic process model is determined by the density $f_{X_n|X_{n-1}}$. Following the R convention, in the pomp package the simulation of $f_{X_n|X_{n-1}}$ (the process model) is referred to as the *rprocess* whereas the evaluation of $f_{X_n|X_{n-1}}$ is referred as the *dprocess*. Similarly, for the measurement process, $f_{Y_n|X_n}$, the simulation component is *rmeasure*, while the evaluation component is called *dmeasure*.

3 Robfilter

4 Sim.DiffProc Package - PMLE

Pseudo-maximum likelihood estimators for one-dimensional SDEs. The four schemes implemented are: **Euler**, **Ozaki**, **Shoji-Ozaki**, **Kessler** method. These methods don't approximate the transition density directly but the path of the process X_t in such a way that the discretized version of the process has a usable likelihood.

Consider the general form of the SDE:

$$dX_t = f(X_t, t, \theta)dt + g(X_t, t, \theta)dWt \tag{5}$$

An assumption is made that the infinitesimal coefficients don't depend on t, so the equation can be approximated as

$$dX_t = f(X_t, \theta)dt + g(X_t, \theta)dW_t \tag{6}$$

where X_t is considered a time-homogeneous process. Using the above assumption, the transition density can be approximated as $p(\Delta t, x, y) = p(t_{i-1}, x, y, t_i)$. Given initial conditions X_0 and θ_0, the log likelihood function is

$$l_n(\theta) = \sum_{i=1}^{n} logp_\theta(\Delta t, X_{i-1}, X_i) + logp_\theta(X_0)$$

The PMLE (pseudo maximum likelihood estimator) $\tilde{\theta}_n = \arg\max h_n(\theta|X_1, \cdots, X_n)$ is estimated from a density h that doesn't belong to the family of the true conditional density but is compatible with the moments with p (??). This method applied to high frequency data, i.e. $\Delta t \to 0, n\Delta t \to \infty$.

5 HPloglik

6 abctools

7 sde (Stefano 2015)

8 yuima (Stefano 2014)

Simulation and inference for multidimensional SDEs

9 PSM package (Stig and Soren 2013)

Estimation of linear and non-linear mixed-effects models using SDEs. Also uses filtering to find smoothed estimates for the model states. It allow any multivariate non-linear time-variant model and had the ability to handle missing observations. Kalman filtering is used for linear models.

A mixed-effects model is used wherein the variation is split into intra-individual variation and inter-individual variation using a 2 stage model. The first stage model is a state space model consisting of a continuous state equation and defined by the dynamics of the system and a set of discrete measurement equations which defines the relation between the obtained observations and the system observations.

$$\begin{aligned} dx_t &= (A(\phi_i)x_t + B(\phi_i)u_t)dt + \sigma_\omega(\phi_i)d\omega_t \\ y_{ij} &= C(\phi_i)x_{ij} + D(\phi_i)u_{ij} + e_{ij} \end{aligned}$$

For a general non-linear model, the system is specified as

$$\begin{aligned} dx_t &= f(\phi_i, x_t, u_t, t)dt + \sigma(\phi_i, u_t, t)d\omega_t \\ y_{ij} &= g(\phi_i, u_{ij}, x_{ij}, t_{ij}) + e_{ij} \end{aligned}$$

where t is a continuous variable and in multivariable case, the state and measurement variables can be denoted as a vector at time t_{ij}. The noise is assumed to follow Gaussian noise $e_{ij} \in \mathcal{N}(0, S(\phi_i, u_t, t))$, where S denotes the covariance matrix. The model parameters are denoted by ϕ_i.

For the model evaluation, it requires computations of the Jacobian matrix with first order partial derivatives, $\frac{\partial f}{\partial x_t}$ and $\frac{\partial g}{\partial x_t}$.

The second stage model describes the variation of the model parameters ϕ_i between individuals,

$$\phi_i = h(\theta, \eta_i, Z_i)$$

where η_i is the multivariate random effect parameter for the i^{th} parameter and is assumed Gaussian, i.e., $\eta_i \in \mathcal{N}(0, \Omega)$

9.1 Parameter Estimation

Maximum likelihood estimator is used for parameter estimation. The full set of model parameters to be estimated are the matrices $\Sigma, \sigma_\omega, \Omega, \theta$. The likelihood function is approximated by a second-order Taylor expansion around the value of $\hat{\eta}_i$ that maximizes it.

$$-\log L(\Theta) \approx \sum_{i=1}^{N} \frac{1}{2} \log \left| \frac{-\Delta l_i}{2\pi} \right| - l_i$$

This likelihood function is evaluated using the Kalman filter which gives an exact solution for linear models. For non-linear models, the Extended Kalman filter is used which is only an approximation. There are 4 types of state and state covariance estimates available using the EKF, each of which differs in the way data is used:

1. Simulation estimate: $\hat{x}_{i(j|0)}, \hat{P}_{i(j|0)}$
   estimate of the state evolution for repeated experiments without updating based on measurements, yields a confidence band for the state evolution

2. Prediction estimate: $\hat{x}_{i(j|j-1)}, \hat{P}_{i(j|j-1)}$
   prediction required to give the conditional density for the next observation at time t_{ij} given the observations upto $t_{i(j|j-1)}$

3. Filtering estimate : $\hat{x}_{i(j|j)}, \hat{P}_{i(j|j)}$
   best estimate at time t_{ij} given the observations upto time t_{ij}

4. Smoothing estimate: $\hat{x}_{i(j|N)}, \hat{P}_{i(j|N)}$
   optimal estimate at time t_{ij} utilizing all observations both prior and after the time t_{ij}

There are 2 key objects in the packages, data object and model object. The data object contains sample times, observations and inputs while the model object contains everything else related to the model.

```
library('PSM')
PSM.template(Linear=TRUE,dimX=3,dimY=1,dimU=0,dimEta=3)

##
##
## MyModel <- vector(mode="list")
## MyModel$Matrices=function(phi) {
##   list(
##        matA=matrix(c(  ), nrow=3, ncol=3),
##        matC=matrix(c(  ), nrow=1, ncol=3)
##        )
## }
## MyModel$h = function(eta,theta,covar) {
##   phi <- theta
##   phi
## }
## MyModel$S = function(phi) {
##   matrix(c(  ), nrow=1, ncol=1)
## }
```

```
## MyModel$SIG = function(phi) {
##   matrix(c(  ), nrow=3, ncol=3)
## }
## MyModel$XO = function(Time,phi,U) {
##   matrix(c(  ), nrow=3, ncol=1)
## }
## MyModel$ModelPar = function(THETA) {
##   list(theta=list(  ),
##        OMEGA=matrix(c(  ), nrow=3, ncol=3)
##        )
## }
```

The data object is a list with the following components:

1. Y - the matrix with observations, columns hold multivariate observations at a time point, missing observations entered as NA
2. $Time$ - sample times, same number of columns in Y

```
MyData <- list()
MyData[[1]] <- list(Time=1:4,Y=matrix(c(2.1,3.2,3.4,3.7),nrow=1),covar=c(BMI=20.1))
MyData[[2]] <- list(Time=3:7,Y=matrix(c(1.9,2.1,2.0,2.9,3.5),nrow=1),covar=c(BMI=23.4))
```

Main functions for parameter estimation include:

1. PSM.estimate(Model, Data, THETA, deltaTime)
2. PSM.estimate(Model, Data, Par, CI)
3. PSM.smooth(Model, Data, THETA, subsample)
4. PSM.plot(Data, Smooth, indiv, type)
5. PSM.template(Linear,dimX,dimY,dimU,dimEta,file)

9.2 Linear Coupled SDE (Dosing in two-compartment model)

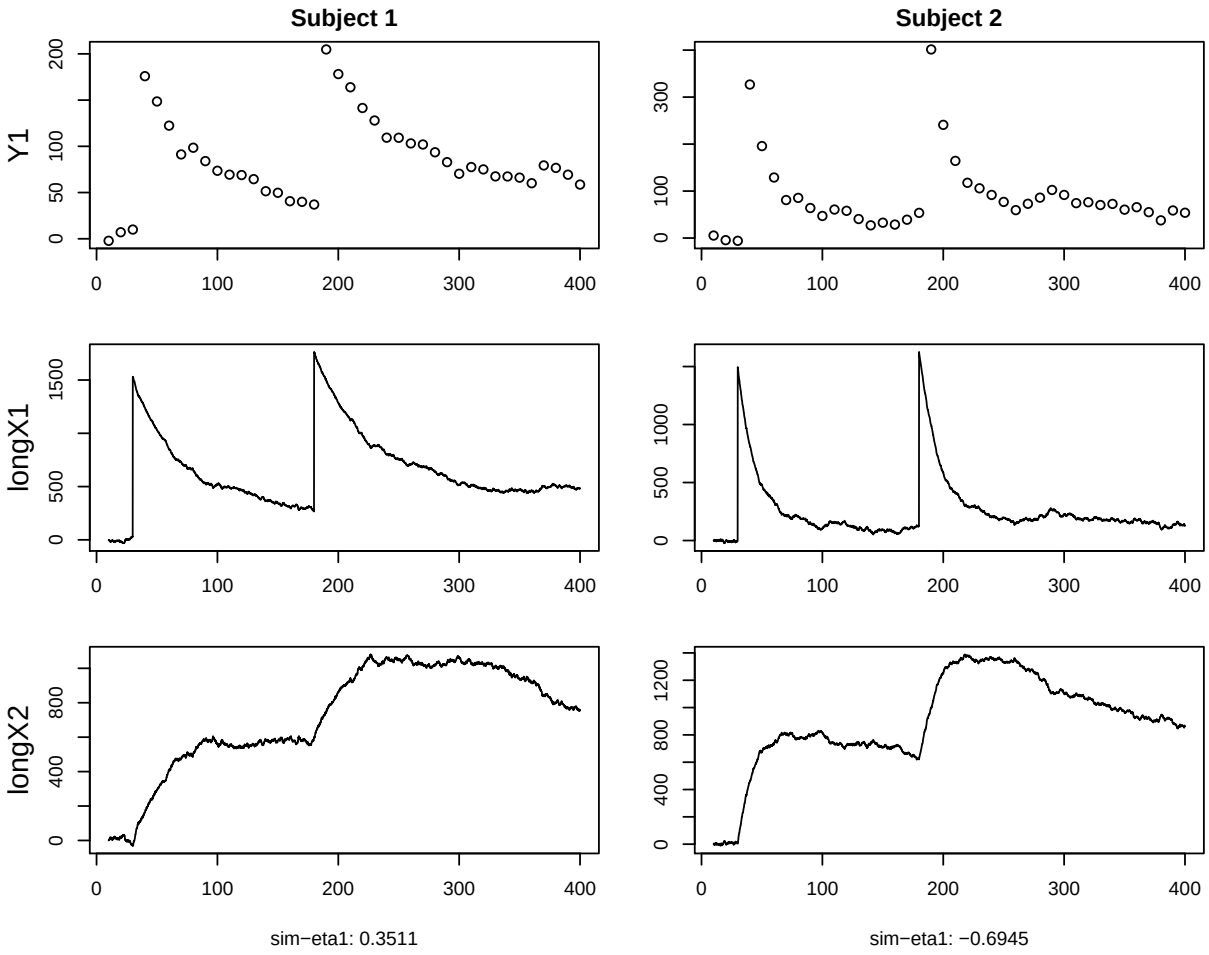
$$
\begin{aligned}
dA_1 &= \left(-\frac{CL}{V_1^i}A_1 - \frac{CL_d}{V_1^i}A_1 + \frac{CL_d}{V_2}A_2 \right) dt + \sigma_1 d\omega \\
dA_2 &= \left(\frac{CL_d}{V_1^i}A_1 - \frac{CL_d}{V_2}A_2 \right) dt - \sigma_1 d\omega \\
Y &= \frac{A_1}{V_1^i} + e
\end{aligned}
$$

```
Model.SimDose = list()

Model.SimDose$Matrices = function(phi)
{
        V1i <- phi$V1i; V2=phi$V2; CL = phi$CL; CLd = phi$CLd;
```

```
        matA <- matrix(c(-(CL+CLd)/V1i , CLd/V2 , CLd/V1i , -CLd/V2 ) ,nrow=2,byrow=T)
        matC <- matrix(c(1/V1i,0),nrow=1)
        list(matA=matA,matC=matC)
}

Model.SimDose$X0 = function(Time=Na,phi,U=Na) { matrix(0,nrow=2) }

Model.SimDose$SIG = function(phi)
{
        sig1 <- phi[["sig1"]]
        matrix(c( sig1,0,-sig1,0), nrow=2, byrow=T)
}

Model.SimDose$S = function(phi) { matrix(phi[["S"]]) }

Model.SimDose$h = function(eta,theta,covar)
{
        phi <- theta
        phi$V1i <- theta$V1*exp(eta[1])
        phi
}

Model.SimDose$ModelPar = function(THETA)
{
        V2 <- 10
        CLd <- 0.1
        list(theta=list(V1 = THETA['V1'],V2=V2,CLd=CLd,CL=THETA['CL'], sig1=THETA['sig1'], S=
        OMEGA=matrix(THETA['OMEGA1']) )
}

SimDose.THETA <- c(CL=0.05,V1 = 5, sig1 = 10 , S = 20 , OMEGA1 = .2)

N = 5
SimDose.Data <- vector(mode="list",length=N)
for (i in 1:N)
{
        SimDose.Data[[i]]$Time <- seq(from=10,by=10,to=400)
        SimDose.Data[[i]]$Dose <-list(Time = c(30,180), State = c(1, 1), Amount = c(1500,1500
}
```

The previous code is for setting up the parameters. Now the simulation and plotting can be done as follows:

```
SimDose.Data <- PSM.simulate(Model.SimDose, SimDose.Data, SimDose.THETA, deltaTime=.1)

## Simulating individual: 1, 2, 3, 4, 5, Done

PSM.plot(SimDose.Data,indiv=1:2,type=c('Y','longX','eta'))
```




```
parA <- list(LB=SimDose.THETA*.5, Init=SimDose.THETA , UB=SimDose.THETA*1.5 )
fitA <- PSM.estimate(Model.SimDose,SimDose.Data,parA,CI=T)
fitA[1:5]

## $NegLogL
## [1] 699.2027
##
## $THETA
##          CL          V1        sig1           S      OMEGA1
##  0.05023643  5.01036749  8.80832025 25.81996577  0.22403311
##
## $CI
##                      CL       V1      sig1        S      OMEGA1
## Lower CI95 0.04878348 2.732171  6.682066 15.30694 -0.05704457
## MLE        0.05023643 5.010367  8.808320 25.81997  0.22403311
```

```
## Upper CI95 0.05168939 7.288564 10.934575 36.33299  0.50511080
##
## $SD
##                 CL       V1     sig1        S   OMEGA1
## [1,] 0.0007413056 1.162345 1.084824 5.363787 0.143407
##
## $COR
##                  CL           V1         sig1             S       OMEGA1
## CL      1.00000000 -0.011979087 -0.056486460  0.055135183 -0.010157000
## V1     -0.01197909  1.000000000  0.024442660 -0.016560949 -0.008879254
## sig1   -0.05648646  0.024442660  1.000000000 -0.589445931 -0.004512447
## S       0.05513518 -0.016560949 -0.589445931  1.000000000 -0.001507466
## OMEGA1 -0.01015700 -0.008879254 -0.004512447 -0.001507466  1.000000000

SimDose.THETA

##     CL     V1   sig1      S OMEGA1
##   0.05   5.00  10.00  20.00   0.20
```

Based on the estimated parameters, it is possible to obtain a smoothes estimate of the model states

```
out <- PSM.smooth(Model.SimDose, SimDose.Data, fitA$THETA, subsample = 20)
names(out[[1]])

##  [1] "Time"    "Xs"      "Ps"      "Ys"      "Xf"      "Pf"      "Xp"
##  [8] "Pp"      "Yp"      "R"       "eta"     "negLogL"
```