Replication materials for: 'DiffusionRgqd: An R Package for Performing Inference and Analysis on Time-Inhomogeneous Quadratic Diffusion Processes'

Etienne A.D. Pienaar

December, 2015

Contents

Introduction

This file contains replication materials for the examples section of the Package paper as submitted to the Journal of Statistical Software (JSS). Although the code snippets of the paper doesn't contain any comments, the code presented here is commented for clarification. For readability, this document provides the replication materials only, and thus contains no R output. As such, this document should be most useful when read in conjunction with the paper.

Example 7.1: Generate the transitional density of a time-inhomogeneous GQD

```
# Load the package:
library("DiffusionRgqd")

# Remove any pre-existing models:
GQD.remove()

# Define the model wihin the GQD-framework:
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)))}

# Some peripheral parameters:
states    <- seq(5, 15, 1 / 10)
initial   <- 8
Tmax      <- 5
Tstart    <- 1
increment <- 1 / 100

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

# Plot the transitional density:
library("rgl")
open3d(windowRect = c(50, 50, 690, 690), zoom=0.95)
persp3d(x = M$Xt,y = M$time,z = M$density, col = 3, box = F,
  xlab = 'State (X_t)', ylab = 'Time(t)',zlab = 'Density f(X_t|X_s)')
```

Example 7.2: Time-inhomogeneous Jacobi diffusion.

```
# Remove pre-existing models and define the model
GQD.remove()
a <- 0.5; b <- 0.6; cc <- 1; X0 <- 0.5;

# Define the model:
G0 <- function(t){a * (b * (1 + 0.25 * sin(pi * t)))}
G1 <- function(t){-a}
Q1 <- function(t){cc}
Q2 <- function(t){-cc}

# Approximate the transitional density using various truncation orders:
states <- seq(0.001, 0.999, 0.001)
res1 <- GQD.density(X0, states, 0, 2, 0.01, Dtype='Beta', Trunc=c(4, 4))
res2 <- GQD.density(X0, states, 0, 2, 0.01, Dtype='Beta', Trunc=c(6, 6))
res3 <- GQD.density(X0, states, 0, 2, 0.01, Dtype='Beta', Trunc=c(8, 8))

# Set up a simulation algortihm for comparing the approximate trans. dens.
delt <- 0.001
N    <- 100000
d    <- 0
X    <- rep(X0,N)
when <- c(0, 0.25, 0.5, 1, 1.75)

for(i in 2:2000)
{
 X <- pmax(pmin(X + (G0(d)+ G1(d) * X) * delt
   + sqrt(Q1(d) * X + Q2(d) * X^2) * rnorm(length(X), sd = sqrt(delt)),
   1), 0)

 d <- d + delt
 if(any(when == round(d, 3)))
 {
   index <- which(res1$time == round(d, 3))
   hist(X, col = '#F7F7F7', freq = F, breaks = 30,
   main = paste0('Transitional Density at t  =  ', round(d, 3)),
     ylim = c(0, 3))
   lines(res1$density[, index] ~ res1$Xt, col = '#1B7837', lty = 'dotdash',lwd=1)
   lines(res2$density[, index] ~ res2$Xt, col = '#D92120', lty = 'solid',lwd=1)
   lines(res3$density[, index] ~ res3$Xt, col = '#5289C7', lty = 'dashed',lwd=2)
 }
}

# Now plot the transitional density using a built-in function:
GQD.plot(res2)
```

Example 7.3 Bivariate non-linear dynamics - The stochastic Lotka-Volterra equations.

```
# Remove pre-existing models:
GQD.remove()

# Define the model:
a10 <- function(t){1.5}
a11 <- function(t){-0.4}
c10 <- function(t){0.05}

b01 <- function(t){-1.5}
b11 <- function(t){0.4}
b02 <- function(t){-0.2}
f01 <- function(t){0.1}

# Approximate the transitional density:
res <- BiGQD.density(Xs = 5, Ys = 5, Xt = seq(3, 8, length = 50),
  Yt = seq(2, 6, length = 50), s = 0, t = 10, delt = 0.01)


# Visualize the evolution of the transitional density:
data(SDEsim3)
attach(SDEsim3)
library("colorspace")
colpal=function(n){rev(sequential_hcl(n,power=1,l=c(40,100)))}
time.index <- c(10, 300, 750, 1000) +1
for(i in time.index)
{
 filled.contour(res$Xt, res$Yt, res$density[, , i],
 main = paste0('Transition Density \n (t = ', res$time[i], ')'),
 color.palette = colpal,
   xlab = 'Prey', ylab = 'Predator', plot.axes =
  {
    lines(my ~ mx, col = 'black', lty = 'dashed', lwd = 2)
    points(res$cumulants[5, i] ~ res$cumulants[1, i], bg = 'white',
      pch = 21, cex=1.5)
    axis(1); axis(2);
    legend('topright', lty = c('dashed', NA), pch = c(NA, 21),
      lwd = c(2, NA), legend = c('Simulated Expectation',
      'Predicted Expectation'))
   })

}
```

Example 7.4 Maximum likelihood estimation - Stochastic volatility models

```
# Source data using the Quandl package:
library("Quandl")
quandldata1 <- Quandl("YAHOO/INDEX_GSPC", collapse = "weekly",
start_date = "1990-01-01", end_date = "2015-01-01",  type = "raw")
St <- rev(quandldata1[, names(quandldata1) == 'Close'])
time1 <- rev(quandldata1[, names(quandldata1) == 'Date'])

quandldata2 <- Quandl("YAHOO/INDEX_VIX", collapse = "weekly",
start_date = "1990-01-01", end_date = "2015-01-01",  type = "raw")
Vt <- rev(quandldata2[, names(quandldata2) == 'Close'])

# Fit the Heston Model:
GQD.remove()
a00 <- function(t){theta[1]}
a01 <- function(t){-0.5 * theta[2] * theta[2]}
c01 <- function(t){theta[2] * theta[2]}
d01 <- function(t){theta[2] * theta[5] * theta[6]}
b00 <- function(t){theta[3]}
b01 <- function(t){-theta[4]}
e01 <- function(t){theta[2] * theta[5] * theta[6]}
f01 <- function(t){theta[5] * theta[5]}

#Give some starting parameters and calculate MLEs:
X    <- cbind(log(St), (Vt / 100)^2)
time <- cumsum(c(0, diff(as.Date(time1)) * (1 / 365)))
theta.start <- c(0, 1, 1, 0.5, 1, 0)
model_1     <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)

# Print the parameter estimates:
GQD.estimates(model_1)

# Define a new model:
GQD.remove()
a00 <- function(t){theta[1]}
a02 <- function(t){-0.5 * theta[2] * theta[2]}
c02 <- function(t){theta[2] * theta[2]}
d02 <- function(t){theta[2] * theta[5] * theta[6]}
b00 <- function(t){theta[3]}
b01 <- function(t){-theta[4]}
e02 <- function(t){theta[2] * theta[5] * theta[6]}
f02 <- function(t){theta[5] * theta[5]}

# Fit the revised model:
theta.start <- c(0, 1, 1, 1, 1, 0)
model_2     <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)

# Compare aic statistics for the two models:
GQD.aic(list(model_1, model_2))
```

```r
# Fit another model:
GQD.remove()
a02 <- function(t){-0.5 * theta[1] * theta[1]}
c02 <- function(t){theta[1] * theta[1]}
d02 <- function(t){theta[1] * theta[4] * theta[5]}
b00 <- function(t){theta[2]}
b01 <- function(t){-theta[3]}
e02 <- function(t){theta[1] * theta[4] * theta[5]}
f02 <- function(t){theta[4] * theta[4]}

theta.start <- c(1, 1, 1, 1, 0)
model_3 <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)

# Fit another model:
GQD.remove()
a00 <- function(t){theta[1]}
a10 <- function(t){theta[7]}
a02 <- function(t){-0.5 * theta[2] * theta[2]}
c02 <- function(t){theta[2] * theta[2]}
d02 <- function(t){theta[2] * theta[5] * theta[6]}
b00 <- function(t){theta[3]}
b01 <- function(t){-theta[4]}
e02 <- function(t){theta[2] * theta[5] * theta[6]}
f02 <- function(t){theta[5] * theta[5]}

theta.start <- c(0, 1, 1, 1, 1, 0, 0)
model_4 <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)

# Fit another model:
GQD.remove()
a00 <- function(t){theta[1]}
a02 <- function(t){-0.5 * theta[2] * theta[2]}
c02 <- function(t){theta[2] * theta[2]}
d02 <- function(t){theta[2] * theta[5] * theta[6]}
b00 <- function(t){theta[3]}
b10 <- function(t){theta[7]}
b01 <- function(t){-theta[4]}
e02 <- function(t){theta[2] * theta[5] * theta[6]}
f02 <- function(t){theta[5] * theta[5]}
theta.start <- c(0, 1, 1, 1, 1, 0, 0)
model_5 <- BiGQD.mle(X, time, mesh = 10, theta = theta.start)

# Compare all of the models' AIC statistics:
GQD.aic(list(model_1, model_2, model_3, model_4, model_5))

# Have a look at model_4's parameter estimates:
GQD.estimates(model_4)
```

Example 7.5 Model selection for 2D diffusions via DIC

```
data(SDEsim4)
attach(SDEsim4)
plot(Xt~time, type = 'l', col = 'blue', ylim = c(0, 25),
  main = 'Simulated Data', xlab = 'Time (t)', ylab = 'State')
lines(Yt~time, col = 'red')

GQD.remove()
a00 <- function(t){theta[1] * theta[2]}
a10 <- function(t){-theta[1]}
c11 <- function(t){theta[3] * theta[3]}
b00 <- function(t){theta[4] * theta[5]+theta[7] * sin(0.25 * pi * t)}
b01 <- function(t){-theta[4]}
f01 <- function(t){theta[6] * theta[6]}

priors <- function(theta){dnorm(theta[1], 1, 5) * dnorm(theta[4], 1, 5)}

mesh    <- 10
updates <- 150000
burns   <- 50000
X <- cbind(Xt, Yt)
th <- c(5, 5, 5, 5, 5, 5, 5)
par.sds <- c(0.22, 0.30, 0.02, 0.11, 0.04, 0.01, 0.21)
m1 <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns)


GQD.estimates(m1, thin = 200)


M  <- 4
SaveOutput  <- list()
M.counter   <- 1
Tot.counter <- 1
kill.count  <- 1
while((M.counter <= M) & (kill.count < 20))
{
  th <- runif(7, 1, 10)
  m1 <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns,
                   Tag = paste0('Model_A_run_', M.counter))
  if(!m1$failed.chain)
  {
    SaveOutput[[Tot.counter]]  <-  m1
    Tot.counter <- Tot.counter+1
    M.counter   <- M.counter +1
    kill.count  <- 1
  }else
  {
    kill.count <- kill.count+1
  }
}

GQD.remove()
```

```
a00 <- function(t){theta[1] * theta[2]}
a10 <- function(t){-theta[1]}
c00 <- function(t){theta[7] * (1+sin(0.25 * pi * t))}
c10 <- function(t){theta[3] * theta[3]}

b00 <- function(t){theta[4] * theta[5]+theta[8] * sin(0.25 * pi * t)}
b01 <- function(t){-theta[4]}
f01 <- function(t){theta[6] * theta[6]}

priors <- function(theta){dnorm(theta[1], 1, 5) * dnorm(theta[4], 1, 5)}

par.sds    <- c(0.17, 0.29, 0.07, 0.10, 0.04, 0.01, 0.83, 0.19)
M.counter  <- 1
kill.count <- 1
while((M.counter <= M) & (kill.count < 20))
{
  th <- runif(8, 1, 10)
  m2 <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns,
                   Tag = paste0('Model_B_run_', M.counter))

  if(!m2$failed.chain)
  {
    SaveOutput[[Tot.counter]]  <- m2
    Tot.counter <- Tot.counter+1
    M.counter   <- M.counter +1
    kill.count  <- 1
  }else
  {
    kill.count <- kill.count+1
  }
}

GQD.remove()
a00 <- function(t){theta[1] * theta[2]}
a10 <- function(t){-theta[1]}
a01 <- function(t){theta[7]}
c11 <- function(t){theta[3] * theta[3]}

b00 <- function(t){theta[4] * theta[5]+theta[8] * sin(0.25 * pi * t)}
b01 <- function(t){-theta[4]}
f01 <- function(t){theta[6] * theta[6]}

priors <- function(theta){dnorm(theta[1], 1, 5) * dnorm(theta[4], 1, 5)}
par.sds  <- c(0.18, 0.95, 0.02, 0.10, 0.03, 0.01, 0.23, 0.18)
M.counter  <- 1
kill.count <- 1
while((M.counter <= M)&(kill.count < 20))
{
  th <- runif(8, 1, 5)
  m3 <- BiGQD.mcmc(X, time, mesh, th, par.sds, updates, burns,
                   Tag = paste0('Model_C_run_', M.counter))
  if(!m3$failed.chain)
  {
```

```
    SaveOutput[[Tot.counter]]  <-  m3
    Tot.counter <- Tot.counter+1
    M.counter   <- M.counter +1
    kill.count  <- 1
  }else
  {
    kill.count <- kill.count+1
  }
}

GQD.dic(SaveOutput)

GQD.estimates(SaveOutput[[12]], 200)
```

Example 7.6 Scalar first passage times

```
# Compare our scheme to an existing package:

# Use the fptApprox package to calculate a FPT density:
library("fptdApprox")

OU <- diffproc(c("alpha * x + beta", "sigma^2",
  "dnorm((x-(y * exp(alpha * (t-s)) - beta * (1 - exp(alpha * (t-s)))
  / alpha)) / (sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha))),
  0, 1) / (sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha)))",
  "pnorm(x,  y * exp(alpha * (t-s)) - beta * (1 - exp(alpha * (t-s))) /
  alpha, sigma * sqrt((exp(2 * alpha * (t-s)) - 1) / (2 * alpha)))"))

# Calculate the density:
res1 <- Approx.fpt.density(OU,  0,  10,  3, "5+0.25 * sin(2 * pi * t)",
  list(alpha = -0.5, beta = 0.5 * 5, sigma = 1))


# Now do the same using DiffusionRgqd:
# Define the model (after accounting for the static-barrier transform):
GQD.remove()
G0 <- function(t)
{
  0.5 * 5 - 0.5 * pi * cos(2 * pi * t)-0.5 * 0.25 * sin(2 * pi * t)
}
G1 <- function(t){-0.5}
Q0 <- function(t){1}
# Calculate the FPT density:
res2 <- GQD.TIpassage(Xs = 3, B = 5, s = 0, t = 10, delt = 1 / 200)

# PLot the resulting densities:
plot(res1$y ~ res1$x, type = 'l', col = '#BBCCEE', xlab = 'Time(t)',
  ylab = 'FPT Density', main =  'DiffusionRgqd vs. fptdApproximate.',
  lwd = 2)
lines(res2$density ~ res2$time, col = '#222299', lwd = 2, lty = 'dashed')
legend('topright', lty = c('solid', 'dashed'), col = c('#BBCCEE', '#222299'), legend =
  c('fptdApproximate', 'DiffusionRgqd'), lwd = c(2, 2), bty = 'n')

# Note, although the paper uses a stored dataset for illustrating the simulated
# FPT density, we show here how to simulate such a density:
theta <- 0.5
mu <- function(X, t)
{
  theta[1] * (10+0.2 * sin(2 * pi * t)+0.3 * sqrt(t) * (1+cos(3 * pi * t))) * X-theta[1] * X^2
}
sigma <- function(X, t){sqrt(0.1) * X}
simulate.fpt = function(N, S, B, delt)
{
X = rep(S, N)
Ndim = N
time.vector = rep(0, N)
t = 1
```

```
k = 0
while(Ndim >= 1)
{
 X = X+mu(X, t) * delt+sigma(X, t) * rnorm(Ndim, sd = sqrt(delt))
 I0 = X<B
 X = X[I0]
 count = sum(!I0)
 Ndim = length(X)
 t = t+delt
 if(count>0)
 {
 time.vector[k+1:count] = t
 k = k+count
 }
}
return(time.vector)
}

# res.sim <- simulate.fpt(500000, 8, 12, 1 / 2000)

# Calculate the FPT density using a parameterised model:
GQD.remove()
G1 <- function(t)
{
  theta[1] * (10+0.2 * sin(2 * pi * t)+0.3 * prod(sqrt(t),
  1+cos(3 * pi * t)))
}
G2 <- function(t){-theta[1]}
Q2 <- function(t){0.1}
res3 = GQD.TIpassage(8, 12, 1, 4, 1 / 100, theta = c(0.5))


# Vary the model parameter to see its effect on the FPT density:
plot(res3$density ~ res3$time, type = 'l', ylim = c(0, 1.0),
  main = 'First Passage Time Density', ylab = 'Density', xlab = 'Time',
  cex.main = 0.95)
data(SDEsim5)
lines(SDEsim5$density ~ c(SDEsim5$mids-diff(SDEsim5$mids)[1] / 2),
  type = 's', lty = 'solid', lwd = 1)

library("colorspace")
colpal = function(n){rev(sequential_hcl(n, power = 0.8, l = c(40, 90),
  h = c(-61, -10)))}
th.seq = seq(0.1, 0.5, 1 / 20)
for(i in 2:length(th.seq))
{
  res3 = GQD.TIpassage(8, 12, 1, 4, 1 / 100, theta = c(th.seq[i]))
  lines(res3$density ~ res3$time, type = 'l', col = colpal(10)[i])
}
lines(res3$density ~ res3$time, type = 'l', col = colpal(10)[i], lwd = 2)
legend('topright', legend = th.seq, col = colpal(10), lty = 'solid',
  cex = 0.75, lwd = c(rep(1, 8), 2), title = expression(theta[1]))
```