Package 'GaDiFPT'

February 19, 2015

Type Package

Title First Passage Time Simulation for Gaussian Diffusion Processes

Version 1.0

Date 2015-01-16

Author A. Buonocore, M.F. Carfora

Maintainer Maria Francesca Carfora <f.carfora@iac.cnr.it>

Description In this package we consider Gaussian Diffusion processes and smooth thresholds. After evaluating the mean of the process to check the subthreshold regimen hypothesis, the FPT density function is reconstructed via the numerical quadrature of the integral equation in (Buonocore 1987); first passage times are also generated by the method in (Buonocore 2014) and results are compared. The timestep of the simulations can iteratively be refined. User should provide the functional form for the drift and the infinitesimal variance in the script 'userfunc.R' and for the threshold in the script 'userthresh.R'. All the parameters required by the implementation are to be set in the script 'userparam.R'. Example scripts for common drifts and thresholds are given.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-17 19:43:09

R topics documented:

`GaDiFPT-package` *First Passage Time Simulation for Gauss Diffusion Processes*

Description

We consider Gaussian Diffusion processes and smooth thresholds. After evaluating the mean of the process to check the subthreshold regimen hypothesis, the FPT density function is reconstructed; first passage times are also generated by an algorithm based on the hazard rate method and results are compared. The timestep of the simulations can iteratively be refined. User should provide the functional form for the drift and the infinitesimal variance of the considered diffusion process along with the functional form for the threshold. Example scripts for common drifts and thresholds are given.

Details

Package:	GaDiFPT
Type:	Package
Version:	1.0
Date:	2015-01-16
License:	GPL-2

The **GaDiFPT** package allows to approximate efficiently the f.p.t. density for a diffusion process through a continuous time-dependent boundary. First, the coefficients of the diffusion process under consideration and the related boundary must be defined in a script built according to the structure of the examples reported in the `demo` folder. Parameters setting the initial time and state, the final time and the initial resolution are also to be given in the same script. Then, the `vectorsetup` function will be used to calculate mean and covariance for the specified process. A plot of both the estimated mean of the process (with confidence interval) and the boundary is produced for the user to check the subthreshold regimen. Thus, the function `FPTdensity_byint` constructs a numerical approximation of the First Passage Time density function. Plots of the FPT density, of the FPT distribution function and of the instantaneous firing rate are produced. Then, the function `FPTsimul` generates a sample of first passage times and compares the related histogram to the approximated FPT density. The scripts `Wiener` and `OrnUhl` in the `demo` folder constitute skeleton examples of use of the package for simulation of the classical Wiener and Ornstein-Uhlenbeck processes through different boundaries. More compex examples are also given in `OrnUhlCurrent` and `Logistic` scripts.

Author(s)

A. Buonocore, M. F. Carfora

Maintainer: Maria Francesca Carfora <f.carfora@iac.cnr.it>

References

Buonocore, A., Caputo, L., Pirozzi, E., and Carfora, M.F., A simple algorithm to generate firing times for Leaky Integrate-and-Fire neuronal model. *Math Biosci Eng 11*, 1 (2014), 1–10.

Buonocore, A., Nobile, A.~G., and Ricciardi, L.M., A new integral equation for the evaluation of first-passage-time probability densities. *Adv Appl Prob 19* (1987), 784–800.

`builtfunc` *Functions characterizing the Gaussian Diffusion process X(t) and the considered threshold*

Description

`builtfunc` evaluates all the functions describing a generic Gaussian Diffusion process X(t) (drift, infinitesimal variance, mean, variance and first derivative of the transition density); it also evaluates the threshold function and its time derivative, both required in the evaluation of the kernel function of the Volterra integral equation for the FPT pdf (Buonocore 1987).

Usage

```
a(t)
b(t)
cc(t)
S(t)
Sp(t)
a1(x, t)
a2(t)
mdt(t,y,tau)
vdt(t,tau)
fdt(x,t,y,tau)
psi (t, y, tau)
```

Arguments

`x`	current value of the process X(t)
`t`	current time
`y`	previous value of the process X at time tau
`tau`	previous time

Value

`a1`	gives the drift of the process as a(t)*x + b(t)
`a2`	gives the infinitesimal variance of the process as cc(t)
`S`	gives the threshold
`Sp`	gives the threshold time derivative

`mdt`	gives the mean of the transition pdf of the process
`vdt`	gives the variance of the transition pdf of the process
`fdt`	gives the transition pdf of the process
`psi`	gives the kernel for evaluating the FPT pdf of the process via numerical integration of the Volterra integral equation

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.


delta <- 0.5
time.vec <- seq(0, by=delta, 100)

# linear threshold
Scost <- 6
Sslope <- 0.2
# user provided function (see examples in demo folder)
SSS <- function(t) {
  SSS <- Scost + Sslope*t
}
Slin <- S(time.vec)
plot(time.vec,Slin,type='l',xlab='time',ylab='threshold',main='linear threshold')

# periodic threshold
S0 <- 0
S1 <- 2
Sfr <- 0.5
# user provided function (see examples in demo folder)
SSS <- function(t) {
  SSS <- S1*cos(Sfr*t+S0)
}
Sper <- S(time.vec)
plot(time.vec,Sper,type='l',xlab='time',ylab='threshold',main='periodic threshold')
```

`diffusion` *Diffusion processes*

Description

`diffusion` creates an object of class “diffusion” from the set of provided values. `is.diffusion`checks if its argument is an object of class “diffusion”. `print` shows an object of class “diffusion”.

Usage

```
diffusion(text)
is.diffusion(obj)
## S3 method for class 'diffusion'
print(x, ...)
```

Arguments

`text`	a character vector of length two, containing the infinitesimal mean and infinitesimal variance of the process
`obj`	an R object to be tested
`x`	an object of class “diffusion”
`...`	additional arguments potentially passed (currently none is considered).

Value

`diffusion` returns an object of class “diffusion” that defines a family of diffusion processes. It is a two-component list:

`mean`	character of length 1 with the mathematical expression of the infinitesimal mean of the process;
`var`	character of length 1 with the mathematical expression of the infinitesimal variance of the process.

`is.diffusion` returns `TRUE` or `FALSE` depending on whether its argument is an object of class “diffusion” or not. `print.diffusion` shows a brief description of the process reporting the functional form of its infinitesimal mean and variance.

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
## Creating a "diffusion" object representing a Wiener process
Wiener <- diffusion(c("mu","sigma2"))

## Creating a "diffusion" object representing
## an Ornstein-Uhlenbeck process with an injected current

OrnUhlCur <- diffusion(c("-x/theta + mu + i0*exp(-(t-t0)/theta1)","sigma2"))

## testing diffusion objects

is.diffusion(Wiener)
is.diffusion(OrnUhlCur)
```

examples — *Example scripts and user provided functions for the Gaussian Diffusion Process*

Description

For a generic Gaussian Diffusion process X(t) the drift can be written as a(t)*X(t)+b(t) and the infinitesimal variance as cc(t)^2. User should provide the functional form for a(t),b(t) and cc(t) in the main script. The threshold to be crossed has to be also provided through the function S(t) and its derivative Sp(t) in the same script. Examples of such a script are given for a Wiener process with or without drift through different thresholds (scripts `Wiener`, `Wiener1`, `WienerDrift`) for an Ornstein Uhlenbeck process also in presence of an additional current (scripts `OrnUhl`, `OrnUhlCurrent`) and for a more complex process with time-varying coefficients (`Logistic`).

Usage

```
Wiener.R
```

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

# Wiener process through a periodic boundary: the process is built, its FPT pdf
# evaluated by numerical quadrature and a train of crossing times is simulated.
# Results are shown and saved to a file


library(GaDiFPT)

cat('###################################################################### \n')
cat('#####              First Passage Time Simulation             ##### \n')
cat('#####                  for the Wiener process                ##### \n')
cat('#####                through a periodic boundary             ##### \n')
cat('###################################################################### \n \n \n')

Wiener1 <- diffusion(c("mu","sigma2"))

mu <- 0.0
sigma2 <- 1.0

S0 <- 10
S1 <- 3
Sfr <- 0.005
```

```
Stype <- "periodic"

t0 <- 0.0
x0 <- 0.0
Tfin <- 1000
deltat <- 0.5
N <- floor((Tfin - t0)/deltat)
M <- 3000

quadflag <- 1
RStudioflag <- TRUE

param <- inputlist(mu,sigma2,Stype,t0,x0,Tfin,deltat,M,quadflag,RStudioflag)

fileout <- "results_Wiener1.out"

aaa <- function(t) {
  aaa <- 0.0 + 0.0*t
}

bbb <- function(t) {
 bbb <- mu + 0.0*t
}

SSS <- function(t) {
  SSS <- S0+S1*cos(Sfr*t)
}

SSSp <- function(t) {
  SSSp <- -S1*Sfr*sin(Sfr*t)
}

mp <- numeric(N+1)
up <- numeric(N+1)
vp <- numeric(N+1)
app <- numeric(N)

tempi <- seq(t0, by=deltat, length=N+1)

dum <- vectorsetup(param)
mp <- dum[,1]
up <- dum[,2]
vp <- dum[,3]

splot <- S(tempi)
mp1 <- mp - sqrt(2*sigma2)
mp2 <- mp + sqrt(2*sigma2)
matplot(tempi, cbind(mp,mp1,mp2,splot),type="l",lty=c(1,2,2,1),lwd=1,
        main="mean of the process vs. threshold",xlab="time(ms)",ylab="")
legend("bottomright",c("mean","threshold"),
       lty=c(1,1),col=c("black","blue"))
```

```
Nmax <- which.min(abs(mp[2:(N+1)]-mp[1:N]))

  N1 <- N
  if (quadflag == 0)   N1 <- max(c(Nmax,N/4))

  N1p1 <- N1+1

answer <- FPTdensity_byint(param,N1)
plot(answer)

spikes <- FPTsimul(answer,M)
histplot(spikes,answer)

res_summary(answer,M,fileout)
```

FPTdensity_byint *Evaluation of the FPT density and distribution functions*

Description

The FPT density g0 and distribution function gg0 are evaluated up to a fixed time T on N1max gridpoints by numerical integration of the Volterra integral equation given in Buonocore 1987. Note that this time may not correspond to the final time Tfin when full reconstruction of the FPT density by quadrature is not required (quadflag set to 0 in the input parameters list).

Usage

```
FPTdensity_byint(obj,n1max)
```

Arguments

`obj`	An “inputlist” class object yielding all the input parameters
`n1max`	Total number of gridpoints in the evaluation procedure

Value

Values are returned as an object of class “FPTdensity” yielding the timegrid and the corresponding values of the FPT density and FPT distribution.

Author(s)

A. Buonocore, M.F. Carfora

References

Buonocore, A., Nobile, A.G., and Ricciardi, L.M., A new integral equation for the evaluation of first-passage-time probability densities. *Adv Appl Prob 19* (1987), 784–800.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## Continuing the Wiener() example:


Nmax <- which.min(abs(mp[2:(N+1)]-mp[1:N]))
N1 <- N
if (quadflag == 0)   N1 <- max(c(Nmax,N/4))
N1p1 <- N1+1
answer <- FPTdensity_byint(param,N1)
plot(answer)
```

`FPTsimul` *Simulation of FPT by the Hazard Rate Method*

Description

`FPTsimul` generates M spikes (times of first boundary crossing) for the Gaussian Diffusion proces X(t) by implementing the Hazard Rate Method, based on the instantaneous firing rate, defined as the ratio between the probability density function and the survival function of the process. The asymptotic firing rate is also estimated to shorten computations. `histplot` then produced an histogram of the simulated times and plot it against the approximated FPT density as obtained by `FPTdensity_byint`

Usage

```
FPTsimul(obj,M)
histplot(obj1,obj)
```

Arguments

`obj`	An "FPTdensity" class object yielding the numerical approximation of the FPT density and the FPT distribution on a given timegrid
`M`	The number of crossing times to be simulated
`obj1`	A vector of simulated crossing times as obtained by a call to `FPTsimul`.

Value

`FPTsimul` returns a vector containing the simulated crossing times; `histplot` produces an histogram plot of these crossing times with the approximated FPT density superimposed

Author(s)

A. Buonocore, M.F. Carfora

References

A. Buonocore, L. Caputo, E. Pirozzi, M.F. Carfora, A Simple Algorithm to Generate Firing Times for Leaky Integrate-and-Fire Neuronal Model, *Math Biosci Eng*,11, 1-10 (2014).

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## Continuing the Wiener() example:


spikes <- FPTsimul(answer,M)
histplot(spikes,answer)
```

`inputlist` *User provided parameters*

Description

`inputlist` creates an object of class “inputlist” from the given set of provided values. `print` shows an object of class “inputlist”.

Usage

```
inputlist(m,s,Sty,tini,xini,Tend,delta,Ntime,quadfl,RSfl)
## S3 method for class 'inputlist'
print(x, ...)
```

Arguments

`m`	gives the constant part of the infinitesimal drift of the process
`s`	gives the infinitesimal variance of the process
`Sty`	gives the functional type of the considered threshold
`tini`	gives the initial time in milliseconds
`xini`	gives the value of the resting potential of the process
`Tend`	gives the final time in milliseconds
`delta`	gives the timestep in milliseconds
`Ntime`	gives the total number of crossing times to be simulated
`quadfl`	is a flag denoting the requirement for full reconstruction of the FPT density by numerical integration of the Volterra equation
`RSfl`	is a technical flag to manage opening and closing of plot windows in case RStudio interface is used
`x`	an object of class “inputlist”
`...`	additional arguments potentially passed (currently none is considered).

Value

`inputlist` returns an object of class “inputlist” yielding the user-provided parameters as a named list. `print.inputlist` shows a brief summary of the user provided parameters.

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
## Creating a list of parameters for the Wiener process

mu <- 0.0
sigma2 <- 1.0
Stype <- "constant"
t0 <- 0.0
x0 <- 0.0
Tfin <- 4000
deltat <- 1.0
M <- 1000
quadflag <- 1
RStudioflag <- TRUE

# building an object of \dQuote(inputlist) class and printing a summary of its parameters

param <- inputlist(mu,sigma2,Stype,t0,x0,Tfin,deltat,M,quadflag,RStudioflag)

print(param)
```

`plot.FPTdensity` *Plotting Method for FPTdensity objects*

Description

This function creates the plots of the approximate density function and the approximate distribution function for the FPT problem at hand. It also creates an additional plot for the corresponding hazard rate function, representing the instantaneous rate of the crossing occurrence at a certain time, conditional on its not occurring before that time.

Usage

```
  ## S3 method for class 'FPTdensity'
plot(x, ...)
```

Arguments

`x`	an object of class “FPTdensity”, a result of a call to `FPTdensity_byint` function
`...`	additional arguments potentially passed (currently none is considered).

Author(s)

A. Buonocore, M.F. Carfora

res_summary *User provided parameters*

Description

`res_summary` writes

Usage

```
res_summary(obj,Nspikes,fileout)
```

Arguments

`obj`	an object of class “FPTdensity”, a result of a call to `FPTdensity_byint` function yielding the approximated values of the FPT density and distribution
`Nspikes`	the total number of simulated crossing times
`fileout`	a character indicating the name of the file where the results have to be written.

Value

`res_summary` writes on file a table with the timesteps and the corresponding values of the FPT density and distribution; it also writes the sequence of the generated crossing times. It also evaluates statistics of the crossing time (mean, standard deviation and median) and send the report to the command window.

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
## examples are shown as part of the 'examples.Rd' ones
```

vectorsetup — *Setup of the mean and covariance vectors for the Gaussian Diffusion process*

Description

`vectorsetup` evaluates the vectors mp (mean of the process) and the two covariance factors up and vp (i.e. covariance of the process is given by up*vp) in the interval [t0, Tfin] with timestep deltat

Usage

```
vectorsetup(obj)
```

Arguments

`obj`	An “inputlist” class object yielding all the input parameters

Value

Values are returned as a matrix (mp,up,vp)

Author(s)

A. Buonocore, M.F. Carfora

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## Continuing the Wiener() example:


####       INITIALIZATION OF VECTORS

tempi <- numeric(N+1)
mp <- numeric(N+1)
up <- numeric(N+1)
vp <- numeric(N+1)

# dummy vector
app <- numeric(N)

####    EVALUATION OF MEAN AND COVARIANCE OF THE PROCESS

tempi <- seq(t0, by=deltat, length=N+1)

dum <- vectorsetup(param)
```

```
mp <- dum[,1]
up <- dum[,2]
vp <- dum[,3]

## plot of S and m

splot <- S(tempi)
mp1 <- mp - sqrt(2*sigma2)
mp2 <- mp + sqrt(2*sigma2)
matplot(tempi, cbind(mp,mp1,mp2,splot),type="l",lty=c(1,2,2,1),lwd=1,
        main="mean of the process vs. threshold",xlab="time(ms)",ylab="")
legend("bottomright",c("mean","threshold"),
       lty=c(1,1),col=c("black","blue"))
```

Index