Package ‘temStaR’

August 19, 2020

Title Tempered Stalbe Distribution

Version 0.814

Author Aaron Y.S. Kim [aut, cre],
Stoyan Stoyanov [aut, cre],
Minseob Kim [ctb]

Maintainer Aaron Y.S. Kim <aaron.kim@stonybrook.edu>

Description This package provides useful tools to use the multivariate normal tempered stable distribution and process

License `use_mit_license()`

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports functional,
nloptr,
pracma,
spatstat,
Matrix

Suggests functional,
nloptr,
pracma,
spatstat,
Matrix

R topics documented:

chf_NTS *chf_NTS*

Description

chf_NTS calculates Ch.F of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If a time parameter value is given, it calculates Ch.F of the NTS profess $\phi(u) = E[exp(iu(X(t+s) - X(s)))] = exp(tlog(E[exp(iuX(1))]))$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
chf_NTS(u, param)
```

Arguments

u	An array of u
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$.

Value

Characteristic function of the NTS distribution

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_NTS(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_NTS(u, ntsparam)
```

chf_stdNTS *chf_stdNTS*

Description

`chf_stdNTS` calculates Ch.F of the standard NTS distribution with parameters (α, θ, β). If a time parameter value is given, it calculates Ch.F of the standard NTS profess $\phi(u) = E[exp(iu(X(t+s) - X(s))] = exp(tlog(E[exp(iuX(1))]))$, where X is the standard NTS process generated by the standard NTS distribution with parameters (α, θ, β).

Usage

```
chf_stdNTS(u, param)
```

Arguments

`u`	An array of u
`ntsparam`	A vector of the standard NTS parameters (α, θ, β). For the standard NTS process case it is a vector of parameters $(\alpha, \theta, \beta, t)$.

Value

Characteristic function of the standatd NTS distribution

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_stdNTS(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_stdNTS(u, ntsparam)
```

dnts *dnts*

Description

`dnts` calculates pdf of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates pdf of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates pdf of the NTS profess $f(x)dx = d(P((X(t+s) - X(s)) < x))$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
dnts(xdata, ntsparam)
```

Arguments

`xdata`	An array of x
`ntsparam`	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of the standard NTS parameters (α, θ, β).

Value

Density of NTS distribution

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
x <- seq(from = -6, to = 6, length.out = 101)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
x <- seq(from = -2, to = 2, by = 0.01)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
```

```
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
x <- seq(from = -0.02, to = 0.02, length.out = 101)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')
```

---

fitnts *fitnts*

Description

`fitnts` fit parameters $(\alpha, \theta, \beta, \gamma, \mu)$ of the NTS distribution. This function using the curvefit method between the empirical cdf and the NTS cdf.

Usage

```
\code{fitnts(rawdat)}
\code{fitnts(rawdat), ksdensityflag = 1}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu))}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu)), ksdensityflag = 1}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu)), maxeval = 100, ksdensityflag
```

Arguments

`rawdat`	Raw data to fit the parameters.
`initialparam`	A vector of initial NTS parameters. This function uses the `nloptr` package. If it has a good initial parameter then estimation performs better. If users do not know a good initial parameters, then just set it as `initialparam=NaN`, that is default. The function `cffitnts()` may be helpful to find the initial parameters.
`maxeval`	Maximum evaluation number for `nloptr`. The iteration stops on this many function evaluations.
`ksdensityflag`	This function fit the parameters using the curvefit method between the empirical cdf and the NTS cdf. If `ksdensityflag = 1` (default), then the empirical cdf is calculated by the kernel density estimation. If `ksdensityflag = 0`, then the empirical cdf is calculated by the empirical cdf.

Value

Estimated parameters

Examples

```
library("quantmod")
getSymbols("^GSPC", src="yahoo", from = "2010-1-1", to = "2020-12-31")
pr <- as.numeric(GSPC$GSPC.Adjusted)
ret <- diff(log(pr))
ntsparam <-  fitnts(ret)

Femp = ecdf(ret)
x = seq(from=min(ret), to = max(ret), length.out = 100)
cemp = Femp(x)
ncdf = pnts(x, c(ntsparam))
```

```
plot(x,ncdf,type = 'l', col = "red")
points(x,cemp, type = 'l', col = "blue")
a = density(ret)
p = dnts(x,ntsparam)
plot(x,p,type = 'l', col = "red")
lines(a,type = 'l', col = "blue")
```

---

fitstdnts *fitstdnts*

Description

`fitstdnts` fit parameters (α, θ, β) of the standard NTS distribution. This function using the curvefit method between the empirical cdf and the standard NTS cdf.

Usage

```
\code{fitstdnts(rawdat)}
\code{fitstdnts(rawdat), ksdensityflag = 1}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta))}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta)), ksdensityflag = 1}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta)), maxeval = 100, ksdensityflag = 1}
```

Arguments

`rawdat`	Raw data to fit the parameters.
`initialparam`	A vector of initial standard NTS parameters. This function uses the `nloptr` package. If it has a good initial parameter then estimation performs better. If users do not know a good initial parameters, then just set it as `initialparam=NaN`, that is default.
`maxeval`	Maximum evaluation number for `nloptr`. The iteration stops on this many function evaluations.
`ksdensityflag`	This function fit the parameters using the curvefit method between the empirical cdf and the standard NTS cdf. If `ksdensityflag = 1` (default), then the empirical cdf is calculated by the kernel density estimation. If `ksdensityflag = 0`, then the empirical cdf is calculated by the empirical cdf.

Value

Estimated parameters

Examples

```
library("quantmod")
getSymbols("^GSPC", src="yahoo", from = "2010-1-1", to = "2020-12-31")
pr <- as.numeric(GSPC$GSPC.Adjusted)
ret <- diff(log(pr))
stdret <- (ret-mean(ret))/sd(ret)
stdntsparam <-  fitstdnts(stdret)

Femp = ecdf(stdret)
```

```
x = seq(from=min(stdret), to = max(stdret), length.out = 100)
cemp = Femp(x)
ncdf = pnts(x, c(stdntsparam))
plot(x,ncdf,type = 'l', col = "red")
lines(x,cemp, type = 'l', col = "blue")
a = density(stdret)
p = dnts(x,stdntsparam)
plot(x,p,type = 'l', col = "red", ylim = c(0, max(a$y, p)))
lines(a,type = 'l', col = "blue")
```

---

pnts *pnts*

Description

pnts calculates cdf of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates cdf of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates cdf of the profess $F(x) = P((X(t+s) - X(s)) < x)$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
pnts(xdata, ntsparam, dz = 2^-8, m = 2^12)
```

Arguments

xdata	An array of x
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of the standard NTS parameters (α, θ, β).

Value

Cumulative probability of the NTS distribution

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
x <- seq(from = -6, to = 6, length.out = 101)
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
x <- seq(from = -2, to = 2, by = 0.01)
```

```
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
x <- seq(from = -0.02, to = 0.02, length.out = 101)
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')
```

`qnts` *qnts*

Description

`qnts` calculates quantile of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates quantile of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates quantile of NTS profess. That is it finds x such that $u = P((X(t + s) - X(s)) < x)$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
qnts(u, ntsparam)
```

Arguments

`ntsparam`	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of standard NTS parameters (α, θ, β).
`vector`	of probabilities.

Value

The quantile function of the NTS distribution

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)

alpha <- 1.2
```

```
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)
```

---

rnts *rnts*

Description

`rnts` generates random numbers following NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it generates random numbers of standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it generates random numbers of increments of NTS profess for time interval t.

Usage

```
rnts(n, ntsparam)
```

Arguments

`n`	number of random numbers to be generated.
`ntsparam`	A vector of NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of standard NTS parameters (α, θ, β).

Value

NTS randomnumbers

Examples

```
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)
```

```
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)
```

Index