Package ‘DtD’

February 11, 2020

Type Package
Title Distance to Default
Version 0.2.2
Maintainer Benjamin Christoffersen <boennecd@gmail.com>
License GPL-2
Encoding UTF-8
BugReports https://github.com/boennecd/DtD/issues
LazyData true
LinkingTo Rcpp, RcppArmadillo
Imports Rcpp, checkmate
Suggests knitr, rmarkdown, testthat, microbenchmark
VignetteBuilder knitr
RoxygenNote 7.0.1
SystemRequirements C++11
NeedsCompilation yes
Author Benjamin Christoffersen [cre, aut], R-core [cph], Robert Gentleman [cph], Ross Ihaka [cph]
Repository CRAN
Date/Publication 2020-02-11 08:30:02 UTC

R topics documented:

BS_call ................................................................. 2
BS_fit ................................................................. 3
BS_call

European Call Option Price and the Inverse

Description

Computes the European call option and the inverse. All vectors with length greater than one needs to have the same length.

Usage

BS_call(V, D, T., r, vol)

get_underlying(S, D, T., r, vol, tol = 1e-12)

Arguments

V numeric vector or scalar with price of the underlying asset.
D numeric vector or scalar with debt due in \( T. \).
T. numeric vector or scalar with time to maturity.
r numeric vector or scalar with risk free rates.
vol numeric vector or scalar with volatilities, \( \sigma_s \).
S numeric vector with observed stock prices.
tol numeric scalar with tolerance to \( \text{get_underlying} \). The difference is scaled if the absolute of \( S \) is large than tol as in the tolerance argument to \( \text{all.equal.numeric} \).

Value

Numeric vector or scalar with price of the underlying asset or equity price.

See Also

BS_fit

Examples

library(DtD)
set.seed(58661382)
sims <- BS_sim(
  vol = .2, mu = .03, dt = .1, V_0 = 100, T. = 1, D = rep(80, 20), r = .01)
stopifnot(with(
BS_fit

Fit Black-Scholes Parameters

Description

Function to estimate the volatility, $\sigma$, and drift, $\mu$. See vignette("Distance-to-default", package = "DtD") for details. All vectors with length greater than one needs to have the same length. The Nelder-Mead method from optim is used when method = "mle". Either time or dt should be passed.

Usage

```r
BS_fit(
  S,
  D,
  T.,
  r,
  time,
  dt,
  vol_start,
  method = c("iterative", "mle"),
  tol = 1e-12,
  eps = 1e-08
)
```

Arguments

- `S` numeric vector with observed stock prices.
- `D` numeric vector or scalar with debt due in $T$.
- `T.` numeric vector or scalar with time to maturity.
- `r` numeric vector or scalar with risk free rates.
- `time` numeric vector with the observation times.
- `dt` numeric scalar with time increments between observations.
- `vol_start` numeric scalar with starting value for $\sigma$.
- `method` string to specify which estimation method to use.
- `tol` numeric scalar with tolerance to get_underlying. The difference is scaled if the absolute of $S$ is large than tol as in the tolerance argument to all.equal.numeric.
- `eps` numeric scalar with convergence threshold.
Value

A list with the following components

- **ests**: estimates of $\sigma$, and drift, $\mu$.
- **n_iter**: number of iterations when method = "iterative" and number of log likelihood evaluations when method = "mle".
- **success**: logical for whether the estimation method converged.

Warning

Choosing tol >= eps or roughly equal may make the method alternate between two solutions for some data sets.

Examples

```r
library(DtD)
set.seed(83486778)
sims <- BS_sim(
  vol = .1, mu = .05, dt = .1, V_0 = 100, T. = 1, D = rep(80, 20), r = .01)
with(sims,
  BS_fit(S = S, D = D, T. = T, r = r, time = time, method = "mle"))
```

BS_fit_rolling

Fit Black-Scholes Parameters Over Rolling Window

Description

Function to estimate the volatility, $\sigma$, and drift, $\mu$. E.g., the window can be over a given number of months. See vignette("Distance-to-default", package = "DtD") for details.

Usage

```r
BS_fit_rolling(
  S, D, T., r, time, dt, vol_start,
  method = c("iterative", "mle"),
  tol = 1e-12, eps = 1e-08,
  grp, width, min_obs
)
```
Arguments

- **S**: numeric vector with observed stock prices.
- **D**: numeric vector or scalar with debt due in `T.`.
- **T.**: numeric vector or scalar with time to maturity.
- **r**: numeric vector or scalar with risk free rates.
- **time**: numeric vector with the observation times.
- **dt**: numeric scalar with time increments between observations.
- **vol_start**: numeric scalar with starting value for $\sigma$.
- **method**: string to specify which estimation method to use.
- **tol**: numeric scalar with tolerance to `get_underlying`. The difference is scaled if the absolute of `S` is larger than `tol` as in the tolerance argument to `all.equal.numeric`.
- **eps**: numeric scalar with convergence threshold.
- **grp**: integer vector with the group identifier (e.g., units of months).
- **width**: integer scalar with the units of `grp` to include in the rolling window.
- **min_obs**: integer scalar for the minimum number of observations required in each window.

Value

Matrix with the `grp`, number of observation in the window, parameter estimates, and `'n_iter'` as in `BS_fit`, and whether the estimation method was successful.

An error attribute is added in case other code than `optim` fails. It is a list of lists with the `grp` index where the method failed and the output from `try`.

See Also

`BS_fit`

Examples

```r
# Simulate data
set.seed(55770945)
n <- 21L * 3L * 12L # 21 trading days for 3 years w/ 12 months
sims <- BS_sim(
  vol = .1, mu = .05, dt = .1, V_0 = 100, T. = 1,
  D = runif(n, 80, 90), r = runif(n, 0, .01))
sims$month <- (1:nrow(sims) - 1L) %/% 21L + 1L

# throw out some months
sims <- subset(sims, !month %in% 15:24)

# assign parameters
grp <- sims$month
width <- 12L # window w/ 12 month width
min_obs <- 21L * 3L # require 3 months of data

# estimate results with R loop which is slightly simpler then the
```
# implementation
grps <- unique(grp)
out <- matrix(NA_real_, nrow = length(grps), ncol = 6,
              dimnames = list(NULL, c("mu", "vol", "n_iter", "success", "n_obs", "grp")))
for(g in grps){
  idx <- which(grps == g)
  keep <- which(grp %in% (g - width + 1L):g)
  out[idx, c("n_obs", "grp")]
  <- c(length(keep), g)
  if(length(keep) < min_obs)
    next
  res <- with(sims[keep, ],
              BS_fit(S = S, D = D, T. = T, r = r, time = time, method = "iterative",
                     vol_start = 1))
  out[idx, c("mu", "vol", "n_iter", "success")]
  <- rep(do.call(c, res[c("ests", "n_iter", "success")]), each = length(idx))
}

# we get the same with the R function
out_func <- with(sims, BS_fit_rolling(
                    S = S, D = D, T. = T, r = r, time = time, method = "iterative",
                    grp = month, width = width, min_obs = min_obs))

all.equal(out[, names(out) != "n_iter"],
           out_func[, names(out_func) != "n_iter"])

---

**BS_sim**

*Simulate Stock Price and Price of Underlying Asset*

**Description**

At least one of \(D\), \(r\), or \(T\) needs to have the desired length of the simulated series. All vectors with length greater than one needs to have the same length.

**Usage**

\[
\text{BS_sim}(\text{vol}, \text{mu}, \text{dt}, S_0, D, r, T.)
\]

**Arguments**

- **vol** numeric scalar with \(\sigma\) value.
- **mu** numeric scalar with \(\mu\) value.
- **dt** numeric scalar with time increments between observations.
- **V_0** numeric scalar with starting value of the underlying asset, \(S_0\).
- **D** numeric vector or scalar with debt due in \(T\).
- **r** numeric vector or scalar with risk free rates.
- **T.** numeric vector or scalar with time to maturity.
merton_ll

See Also

BS_fit

Examples

library(DtD)
set.seed(79156879)
sims <- BS_sim(
  vol = .1, mu = .05, dt = .2, V_0 = 100, T. = 1, D = rep(80, 20), r = .01)

# plot underlying
plot(sims$V)

# plot stock
plot(sims$S)

merton_ll

Compute Log-Likelihood of Merton Model

Description

Computes the log-likelihood for a given values of \( \mu \) and \( \sigma \).

Usage

merton_ll(S, D, T., r, time, dt, vol, mu, tol = 1e-12)

Arguments

S   numeric vector with observed stock prices.
D   numeric vector or scalar with debt due in \( T. \).
T.  numeric vector or scalar with time to maturity.
r   numeric vector or scalar with risk free rates.
time numeric vector with the observation times.
dt  numeric scalar with time increments between observations.
vol numeric scalar with the \( \sigma \) value.
mu  numeric scalar with the \( \mu \) value.
tol numeric scalar with tolerance to get_underlying. The difference is scaled if
      the absolute of \( S \) is larger than tol as in the tolerance argument to all.equal.numeric.

See Also

BS_fit
Examples

# we get the same if we call `optim` as follows. The former is faster and is
# recommended
set.seed(4648394)
sims <- BS_sim(
  vol = .1, mu = .05, dt = .1, V_0 = 100, T. = 1, D = rep(80, 20), r = .01)

r1 <- with(
  sims, BS_fit(S = S, D = D, T. = T, r = r, time = time, method = "mle",
               eps = 1e-8, vol_start = .2))

r2 <- optim(c(mu = 0, log_vol = log(.2)), function(par)
  -with(
    sims, merton_ll(S = S, D = D, T. = T, r = r, time = time,
                   mu = par["mu"], vol = exp(par["log_vol"]))))

all.equal(r1$n_iter, unname(r2$counts[1]))
all.equal(r1$ests[1], r2$par[1])
all.equal(r1$ests[2], exp(r2$par[2]), check.attributes = FALSE)

# the log-likelihood integrates to one as it should though likely not the
# most stable way to test this
ll <- integrate(
  function(x) sapply(x, function(S)
    exp(merton_ll(
      S = c(1, S), D = .8, T. = 3, r = .01, dt = 1/250, vol = .2,
      mu = .05)),
    lower = 1e-4, upper = 6)
  stopifnot(isTRUE(all.equal(ll$value, 1, tolerance = 1e-5)))
Index

all.equal.numeric, 2, 3, 5, 7

BS_call, 2
BS_fit, 2, 3, 5, 7
BS_fit_rolling, 4
BS_sim, 6

get_underlying, 2, 3, 5, 7
get_underlying (BS_call), 2

merton_ll, 7

optim, 3, 5

try, 5