Distance to default package

Benjamin Christoffersen

April 28, 2018

This package is provides fast functions to work with the Merton's distance to default model. We will only briefly cover the model here. See e.g., Lando (2009) for a more complete coverage. Denote the observed market values by S_t and unobserved asset values by V_t. We assume that V_t follows a geometric Brownian motion

$$dV_t = \mu V_t \,\mathrm{d}t + \sigma V_t \,\mathrm{d}W_t$$

We assume that we observe the assets over increaments of dt in time. Thus, we will let V_k the value at $t_0 + k \cdot dt$. Thus,

$$V_{k+1} = V_k \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma W_t\right)$$

We further let r denote the risk free rate, D_t denote debt due at time $t+T$. Then

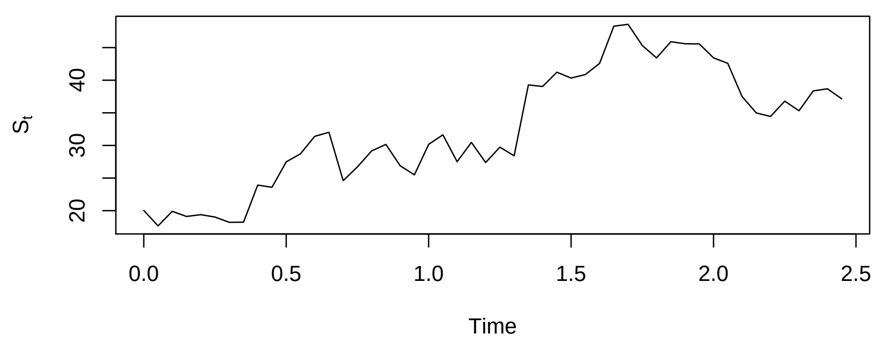
$$\begin{aligned} C(V_t, D_t, T, \sigma, r) &= V_t N(d_1) - D_t \exp\left(-rT\right) N(d_1 - \sigma\sqrt{T}) \\ d_1 &= \frac{\log(V_t) - \log D_t + \left(r + \frac{1}{2}\sigma^2\right) T}{\sigma\sqrt{T}} \qquad (1) \\ S_t &= C(V_t, D_t, T, \sigma, r) \qquad (2) \end{aligned}$$

where C is a European call option. T is the time to maturity, D_t is the debt to due at time $T+t$, and r is the risk free rate. Common choices tend to be $T = 1$ year and D_t as the short term debt plus half of the long term debt. d_1 in equation (1) is the so-called distance to default. Equation (2) can be computed with the `BS_call` function. Further, the `get_underlying` can be used to invert the equation (2)

```r
library(DtD)
(S <- BS_call(100, 90, 1, .1, .3))

## [1] 22.51

get_underlying(S, 90, 1, .1, .3)

## [1] 100

options(digits = 4)
```

To illustrate the above then we can simulate the underlying and transform the data into the stock price as follows

```
# assign parameters
vol <- .1
mu  <- .05
dt  <- .05
V_0 <- 100
t.  <- (1:50 - 1L) * dt
D   <- c(rep(80, 27), rep( 70, length(t.) - 27))
r   <- c(rep( 0, 13), rep(.02, length(t.) - 13))

# simulate underlying
set.seed(seed <- 83992673)
V <- V_0 * exp(
  (mu - vol^2/2) * t. + cumsum(c(
    0, rnorm(length(t.) - 1, sd = vol * sqrt(dt)))))

# compute stock price
S <- mapply(BS_call, V, D, T = 1, r, vol)
plot(t., S, type = "l", xlab = "Time", ylab = expression(S[t]))
```



Despite that the model assume a constant risk free rate than we let it vary in this example. We end by plotting the stock price. Further, we can confirm that we the same underlying after transforming back

```
all.equal(V, get_underlying(S, D, 1, r, vol))

## [1] TRUE
```

We could also have used the simulation function in the package

```r
set.seed(seed) # use same seed
sims <- BS_sim(
  vol = vol, mu = mu, dt = dt, V_0 = V_0, D = D, r = r, T. = 1)

isTRUE(all.equal(sims$V, V))

## [1] TRUE

isTRUE(all.equal(sims$S, S))

## [1] TRUE
```

1 Drift and volatility estimation

There are a few ways to estimate the volaltilty, σ, and drift, μ. This package only includes the iterative procedure and maximum likelihood method covered in Duan (1994); Vassalou, Xing (2004); Duan et al. (2004); Bharath, Shumway (2008). We denote the former as the "iterative" method and the latter as the MLE. We have not implemented the method in Crosbie (2003) as it will be based on two meassurements and may be quite variable (as mentioned in Crosbie, 2003).

The parameters can be estimeted with the `BS_fit` function. The iterative method is used in the following call

```r
# simulate data
set.seed(52722168)
sims <- BS_sim(
  vol = .2, mu = .05, dt = 1/252, V_0 = 100, r = .01, T. = 1,
  # simulate firm that grows partly by lending
  D = 80 * (1 + .03 * (0:(252 * 4)) / 252))

# the sims data.frame has a time column. We need to pass this
head(sims$time, 6)

## [1] 0.000000 0.003968 0.007937 0.011905 0.015873 0.019841

# estimate parameters
it_est <-
  BS_fit(S = sims$S, D = sims$D, T. = sims$T, r = sims$r, time = sims$time)
it_est

## $ests
##     mu    vol
## 0.1717 0.1948
##
## $n_iter
## [1] 10
```

```
##
## $success
## [1] TRUE
```

The volatility is quite close the actual value while the mean is drift. This may be due to the fact that the likelihood is flat in the drift. We can also estimate the parameters when there unequal time gaps in the data set

```
# drop random rows
sims <- sims[sort(sample.int(nrow(sims), 100L)), ]

# the gap lengths are not equal anymore
range(diff(sims$time))

## [1] 0.003968 0.190476

# estimate parameters
it_est <- BS_fit(
  S = sims$S, D = sims$D, T. = sims$T, r = sims$r, time = sims$time,
  method = "iterative")
it_est

## $ests
##       mu      vol
## -0.01562  0.18432
##
## $n_iter
## [1] 10
##
## $success
## [1] TRUE
```

The maximum likelihood estimator is obtained by maximizing the observed log likelihood

$$
\begin{aligned}
L(\mu, \sigma, \vec{S}) \propto & -n \log\left(\sigma^2 dt\right) - \sum_{k=1}^{n} \frac{\left(\log \frac{C^{-1}(S_k, \sigma)}{C^{-1}(S_{k-1}, \sigma)} - \left(\mu - \sigma^2/2\right) dt\right)^2}{\sigma^2 dt} \\
& - 2 \sum_{k=1}^{n} \left(\log C^{-1}\left(S_k, \sigma\right) + \log\left|C'\left(C^{-1}\left(S_k, \sigma\right), \sigma\right)\right|\right)
\end{aligned} \tag{3}
$$

where C^{-1} is the inverse of the call option price in equation (2) and implicitly depend on D_t, T, and r. Notice that we need to use dt in (3) and the time to maturity, T, in C and C^{-1}. The last term in equation (3) follows from the change of variable

$$\begin{aligned}
X &= h^{-1}(Y) \\
f_Y(y) &= f_X\left(h^{-1}(y)\right)\left|\left(h^{-1}\right)'(y)\right| \\
&= f_X\left(h^{-1}(y)\right)\left|\frac{1}{h'\left(h^{-1}(y)\right)}\right|
\end{aligned} \tag{4}$$

where f denotes a density and the subscript denotes which random variable the density is for. We can estimate the parameters with the MLE method as follows

```
mle_est <- BS_fit(
  S = sims$S, D = sims$D, T. = sims$T, r = sims$r, time = sims$time,
  method = "mle")
mle_est

## $ests
##     mu    vol
## 0.1849 0.1824
##
## $n_iter
## [1] 55
##
## $success
## [1] TRUE
```

The result are usually very similar although they need not to as far as I gather

```
it_est$est - mle_est$est

##        mu       vol
## -0.200526  0.001936
```

The iterative method is quite a lot faster though

```
library(microbenchmark)
with(sims,
  microbenchmark(
    iter = BS_fit(
      S = S, D = D, T. = T, r = r, time = time, method = "iterative"),
    mle = BS_fit(
      S = S, D = D, T. = T, r = r, time = time, method = "mle"),
    times = 5))

## Unit: milliseconds
##  expr   min     lq   mean median     uq    max neval cld
##  iter 17.24  17.35  19.93  20.67  22.06  22.32     5  a
##   mle 96.04 102.62 103.23 102.94 106.40 108.15     5   b
```

References

Bharath Sreedhar T., Shumway Tyler. Forecasting Default with the Merton Distance to Default Model // The Review of Financial Studies. 2008. 21, 3. 1339–1369.

Crosbie Peter. Modeling Default Risk. December 2003.

Duan Jin-Chuan, Gauthier Geneviève, Simonato Jean-Guy. On the Equivalence of the KMV and Maximum Likelihood Methods for Structural Credit Risk Models. 2004.

Duan JinChuan. Maximum Likelihood Estimation Using Price Data Of The Derivative Contract // Mathematical Finance. 1994. 4, 2. 155–167.

Lando David. Credit risk modeling: theory and applications. 2009.

Vassalou Maria, Xing Yuhang. Default Risk in Equity Returns // The Journal of Finance. 2004. 59, 2. 831–868.