Mandatory assignment STK4505

Andreas Slaattelid

28 september 2020

Problem 1

a)

We have that clami sizes Z_i follows a log-normal distribution, this gives us the following equation for σ_i:

$$
\begin{aligned}
\frac{sd(Z_i)}{E(Z_i)} &= \left(e^{\sigma_i^2} - 1\right)^{1/2} \\
e^{\sigma_i^2} &= \left(\frac{sd(Z_i)}{E(Z_i)}\right)^2 + 1 \\
\sigma_i^2 &= \ln\left(\left(\frac{sd(Z_i)}{E(Z_i)}\right)^2 + 1\right) \\
\sigma_i &= \sqrt{\ln\left(\left(\frac{sd(Z_i)}{E(Z_i)}\right)^2 + 1\right)}
\end{aligned}
$$

We get the following equation for ξ_i:

$$
\begin{aligned}
E(Z_i) &= e^{\xi_i + \sigma_i^2} \\
\xi_i + \sigma_i^2/2 &= \ln E(Z_i) \\
\xi_i &= \ln E(Z_i) - \sigma_i^2/2
\end{aligned}
$$

```
generate_sigma = function(sd, ex){
  sigma_sq = log((sd/ex)^2 + 1)
  sigma_sq
}

sds = c(1.0, 3.0, 5.0)

sigmas_sq = rep(0, length(sds))

for (i in 1:length(sds)){
  sigmas_sq[i] = generate_sigma(sds[i], 2)
}
sigmas_sq
```

```
## [1] 0.2231436 1.1786550 1.9810015
```

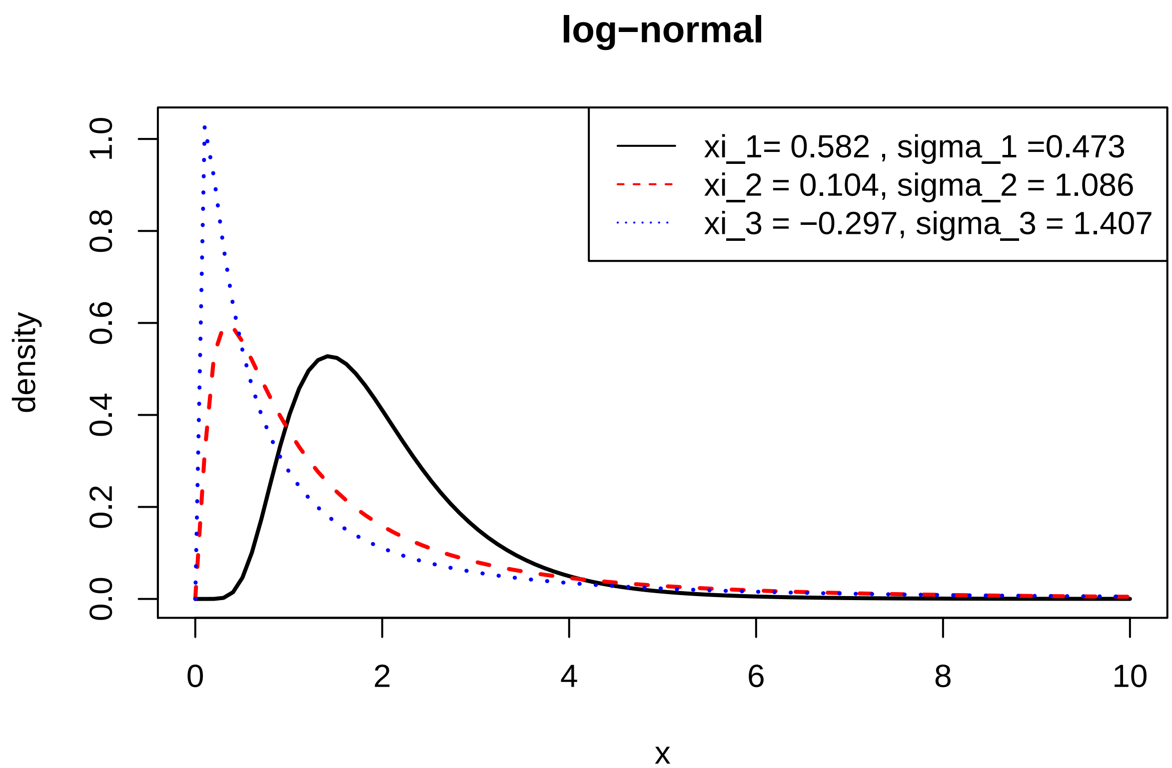
```r
sigmas = sqrt(sigmas_sq)
sigmas
```

```
## [1] 0.4723807 1.0856588 1.4074805
```

```r
#Now we want to calculate the xi's:
xi = log(2) - sigmas_sq/2
xi
```

```
## [1]  0.5815754  0.1038197 -0.2973536
```

The plotting of the corresponding densities are:



b)

```r
reserve = function(xi, sigma, lambda, m){
  N = rpois(m, lambda)
  X = 1:m*0 #generate m zeros
  for (i in 1:m){
    Z = rlnorm(N[i], xi, sigma)
    X[i] =sum(Z)
  }
  lower_reserve = sort(X)[0.95*m] #95% reserve
  upper_reserve = sort(X)[0.99*m] #99% reserve
  cbind(lower_reserve, upper_reserve)
}
```

```
J = 1000; mu = 0.01; T=1
lambda = J*mu*T
for (i in 1:3){
  print(reserve(xi[i], sigmas[i], lambda, 10000))
}
```

```
##      lower_reserve upper_reserve
## [1,]      32.32429      38.51872
##      lower_reserve upper_reserve
## [1,]      41.20256      57.58849
##      lower_reserve upper_reserve
## [1,]      49.39053      81.64788
```

c)

We now have a contract with reduced risk, where the deductable is $a = 0.5$, and the maximum responsibility is $b = 3.0$, this gives us that the indivudual losses are modelled by:

$$H(z) = \begin{matrix} 0 & , z \leq 0.5 \\ z - 0.5 & , 0.5 < z \leq 3.5 \\ 3.0 & , z > 3.5 \end{matrix}$$

So now we just need to sum over the H's this gived us the modified code:

```
reserve_d = function(xi, sigma, lambda, a, b, m){
  N = rpois(m, lambda)
  X = 1:m*0 #generate m zeros
  for (i in 1:m){
    Z = rlnorm(N[i], xi, sigma) #the claim sizes
    H = pmin(pmax(Z-a, 0), b)   #the claim sizes reduced according to contract
    X[i] =sum(H)
  }
  lower_reserve = sort(X)[0.95*m] #95% reserve
  upper_reserve = sort(X)[0.99*m] #99% reserve
  cbind(lower_reserve, upper_reserve)
}

J = 1000; mu = 0.01; T=1
lambda = J*mu*T

for (i in 1:3){
  print(reserve_d(xi[i], sigmas[i], lambda, 0.5, 3.0, 10000))
}
```

```
##      lower_reserve upper_reserve
## [1,]      23.40579      27.93614
##      lower_reserve upper_reserve
## [1,]      18.77553      23.25497
##      lower_reserve upper_reserve
## [1,]      16.30646      20.39107
```

Here we see drastically differences between the reserves, in the case where we have a contract modelled by H, there is clearly a difference in reserves. This is due to the upper bound b on the losses. This makes it impossible to have higher losses than b.

Problem 2

a)

Using the formula for the put option in the book, we get the following code:

```
price_put = function(sigma, r, rg, v0, T=1){
  a = (log(1+rg) - r*T + (sigma**2)*(T/2))/sigma*sqrt(T)
  price = ((1+rg)*exp(-r*T)*pnorm(a) - pnorm(a-sigma*sqrt(T)))*v0
  price
}

price_put(sigma = 0.25, r = 0.04, rg = 0.06, v0=1, T=1)
```

```
## [1] 0.1098784
```

b)

The Monte Carlo method yields:

```
monte_carlo_price = function(sigma, r, rg, v0, T=1, m){
  eps = rnorm(m); #drawing m normal variables.
  R = exp(r*T-sigma**2*(T/2) + sigma*sqrt(T)*eps) - 1
  price = exp(-r*T)*mean(pmax(rg-R,0))*v0
  price
}
sigma_p = c(0.25, 0.30, 0.35)

for (i in 1:length(sigma_p)){
  print(monte_carlo_price(sigma_p[i],r= 0.04, rg=0.06, v0=1,T= 1, m=100000))
}
```

```
## [1] 0.1097889
## [1] 0.129414
## [1] 0.1496775
```

```
monte_carlo_price(0.25, 0.04, 0.06, 1, 1, 100000)
```

```
## [1] 0.1097039
```

For the value $\sigma_1 = 0.25$ we see that the Monte Carlo estimate are close to the one of where we use the formula exactly.

c)

We have two assets which we regard as a correlated normal pair:

$$
\begin{aligned}
X_1 &= \xi_{q1} + \sigma_1\epsilon_1, \quad \epsilon_1 = \eta_1 \\
X_2 &= \xi_{q2} + \sigma_2\epsilon_2, \quad \epsilon_2 = \rho\eta_1 + \sqrt{1-\rho^2}\eta_2
\end{aligned}
$$

$$
\begin{aligned}
\xi_{qj} &= r - \frac{1}{2}\sigma_j^2 \; j = 1,2 \\
\epsilon_j &\sim N(0,1) \; j = 1,2 \\
\eta_j &\sim N(0,1) \; j = 1,2
\end{aligned}
$$

Their returns are:

$$R_j = \exp\left(\xi_{qj}T + \sigma_j\sqrt{T}\epsilon_j\right) - 1,\ j = 1, 2$$

The asset we will regard is $R = w_1R_1 + w_2R_2 = 0.5(R_1 + R_2)$

```
monte_carlo_corr = function(sigma1, sigma2, r, rg, v0, T=1, rho, m){
  eps1 = rnorm(m);
  R1 = exp(r*T-sigma1**2*(T/2) + sigma1*sqrt(T)*eps1) - 1 #first asset simulated.

  eta2 = rnorm(m) #drawing random number
  eps2 = rho*eps1 + sqrt(1-rho^(2))*eta2
  R2 = exp(r*T-sigma2**2*(T/2) + sigma2*sqrt(T)*eps2) - 1

  R = 0.5*(R1 + R2) # the weights are equal
  price = exp(-r*T)*mean(pmax(rg-R,0))*v0
  price
}
rho = c(-0.9, -0.5, 0.0, 0.5, 0.9)

for(i in 1:length(rho)){
  print(monte_carlo_corr(sigma1 = 0.25, sigma2 = 0.35, r=0.04, rg=0.06, v0=1, T=1, rho[i], 10000))
}
```

```
## [1] 0.04992955
## [1] 0.07350584
## [1] 0.09794728
## [1] 0.11367
## [1] 0.1284929
```

If we regard the assets as for example stocks, $\rho = -0.9$ could be interpeted as that there is a negative, linear relationship between the stock prices. So when stock num.1 goes up, then stock num.2 goes down. this leads to a cheaper price for the option due to the negative linear relationship. For $\rho = 0.9$ we have a linear relation ship, where both move in the same direction. This means that when stock num.1 goes up, so will stock.2 go up. if we think in terms of put options, the ideal situation is when the stock falls, so both stock.1 and stock.2 will fall simontanously, hence e higher price for the put makes sense.