Automne 2012

IFT–3245
Démonstration 7, 19 novembre 2012.

---

1. Exercice 1, Chapitre 6.
2. Illustration des intervalles de confiance construit avec une normale ou une student avec l'exemple `queueevent_ci.cpp`.
3. Illustration avec SSJ des intervalles de confiance pour des variables aléatoires discrètes.

   The value-at-risk x_p corresponds to the $(1-p)$-quantile of the loss function. We cannot use the normal approximation for computing a confidence interval, because $1-p$ is too close to 1. We therefore use the binomial confidence interval as follows. We first initialize j and k to the value $q = \lceil n(1-p) \rceil$, and the level of confidence to 0. We then iterative decrease j or increase k until the level becomes greater than or equal to $1-\alpha$. Because we alternatively decrease j and increase k upon successive iterations, q is approximately $(j+k)/2$. Listing 1 presents the program computing the interval using this method. Figure **??** presents the output of the program. If we were asked to estimate x_p for $p = 0.001$ rather than $p = 0.01$, we would need more than $1,000$ observations. Otherwise, the returned quantile will be the maximal loss value, and the program will not be able to get an upper bound for the confidence interval.

Listing 1 – Confidence interval for the value-at-risk

```
import java.util.Arrays;
import umontreal.iro.lecuyer.probdist.*;
import umontreal.iro.lecuyer.randvar.*;
import umontreal.iro.lecuyer.rng.*;

public class VarPlain {
   int N          = 1000;          // Number of simulation runs
   int m          = 5;             // Number of assets
   double T       = 0.1;           // Time horizon
   double sigma2 = 0.3 * 0.3;// sigma^2
   double sigma= 0.3 ; // sigma : volatility
   double r = 0.05; // Drift parameter
   double s0= 50;// Price of assets at time 0
   double bsigma = Math.sqrt(T); // sqrt of T
   double alpha= 0.05; // Confindence level of CI.
   double p = 0.01; // probability

   // Random stream for brownian motion
   RandomStream streamBrownian = new RandMrg();
```

```
public void quantile() {
   double b;  // Value of B(T)
   double[] loss = new double[N];  // Loss for each simulation
   for (int n = 0; n < N; n++) {
      loss[n] = 0.0;
      for (int j = 0; j < m; j++) {
         b = NormalDist.inverseF01 (streamBrownian.nextDouble())
            * bsigma;
         loss[n] += s0
            * (1.0 - Math.exp ((r - sigma2 / 2 ) * T + sigma * b));
      }
   }
   Arrays.sort (loss);
   // compute quantile
   int q = (int) Math.ceil (N * (1.0 - p));
   double xp = loss[q];
   double delta = NormalDist.inverseF01 (1.0 - alpha / 2.0)
      * Math.sqrt (N * (1.0 - p) * p);
   int j = (int) Math.floor (N * (1.0 - p) + 1.0 - delta);
   int k = (int) Math.floor (N * (1.0 - p) + 1.0 + delta);
   if (j < 0 || k >= N)
      System.out.println ("Could not find a normal confidence interval");
   else {
      System.out.println("Value of x_(" + p + ") = " + xp);
      System.out.println("95% normal confidence interval : [" + loss[j] +
                         " , "+ loss[k] + "]");
   }

   BinomialDist bdist = new BinomialDist (N, 1.0 - p);
   j = k = q;
   double level = 0;
   boolean left = true;
   while (level < 1.0 - alpha) {
      if (j == 0 && k == N - 1)
         break;
      if (j > 0 && (left || k == N - 1)) {
         --j;
         level += bdist.prob (j);
         left = false;
      }
      if (k < N - 1 && (!left || j == 0)) {
         level += bdist.prob (k);
         ++k;
         left = true;
      }
   }
   if (level < 1.0 - alpha)
      System.out.println ("Could not find a binomial confidence interval"
   else {
      System.out.println("Value of x_(" + p + ") = " + xp);
```

```
            System.out.println("95% binomial confidence interval : [" + loss[j]+
                               " , "+ loss[k] + "]");
        }
    }

    public static void main (String[] args) {
        VarPlain v = new VarPlain();
        v.quantile();
    }
}
```

```
Value of x_(0.01) = 21.74
95% normal confidence interval : [20.20 , 27.10]

Value of x_(0.01) = 21.74
95% binomial confidence interval : [20.03 , 27.10]
```