Rare-event simulation: Assignment 1

Patrick Laub

February 12, 2020

Due: Wednesday, February 26, 1:30 pm

Submission by email, send completed Jupyter notebook.

[ ]:
```
# numpy is the 'Numerical Python' package
import numpy as np

# Numpy's methods for pseudorandom number generation
import numpy.random as rnd

# scipy is the 'Scientific Python' package
# We'll use the stats package to get some p.d.f.s & c.d.f.s
from scipy import stats

import matplotlib.pyplot as plt
```

1 Random number generation

1) Simulate $R = 10^5$ $\mathsf{Pareto}(10, 100)$ random variables, where our definition of $\mathsf{Pareto}(\alpha, \lambda)$ has the p.d.f.

$$f(x) = \begin{cases} \frac{\alpha}{\lambda}\left[1 + \frac{x}{\lambda}\right]^{-(\alpha+1)} & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Plot a histogram of these points against this p.d.f., like in the lectures.

[ ]:
```
rng = rnd.default_rng(1)
R = 10**5
α = 10.0
λ = 100.0

paretos = ...
```

2) Simulate $R = 10^5$ $\mathsf{LogNormal}(3, 0.1)$ random variables, where our definition of $\mathsf{LogNormal}(\mu, \sigma^2)$ has the p.d.f.

$$f(x) = \begin{cases} \frac{1}{x\sigma\sqrt{2\pi}} \exp\{-\frac{(\log(x)-\mu)^2}{2\sigma^2}\} & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Plot the histogram of your samples against p.d.f. above.

```
[ ]: R = 10**5
     μ = 3
     σ = np.sqrt(0.1)

     lognormals = ...
```

3) Find $\overline{\mu}$, $\overline{\sigma}$ such that $X \sim \mathsf{LogNormal}(\overline{\mu}, \overline{\sigma}^2)$ has $\mathbb{E}[X] = m$ and $\mathbb{V}[X] = v$.

Simulate $R = 10^5$ lognormal random variables from the $\mathsf{LogNormal}(\overline{\mu}, \overline{\sigma}^2)$ where $m = 150$ and $v = 500$.

Plot a histogram of the simulated points, and overlay a vertical line at $x = m = 150$ to check that the simulated points are near the desired mean.

For style points: Find $\overline{\mu}$, $\overline{\sigma}$ using the `solve` function from the `sympy` 'symbolic python' package.

```
[ ]: R = 10**5
     m = 150
     v = 500

     lognormals = ...
```

```
[ ]: print("Sample mean:", lognormals.mean())
     print("Theoretical mean:", m)
     assert np.isclose(lognormals.mean(), m, rtol=0.01)
```

```
[ ]: print("Sample var:", lognormals.var())
     print("Theoretical var:", v)
     assert np.isclose(lognormals.var(), v, rtol=0.01)
```

2 Crude Monte Carlo

We model an insurer's risk reserve process R_t as

$$R(t) = u + pt - \sum_{i=1}^{N_t} U_i$$

where $u \geq 0$, $p > 0$, N_t is a Poisson process with intensity λ and $U_i \overset{\text{i.i.d.}}{\sim} \mathsf{Gamma}(r, m)$.

The only possible times when the insurer's reserve can become negative is at the times $T_1, T_2, \ldots$ when the claims arrive. Define

$$R_i = R(T_i), \quad i = 1, 2, \ldots,$$

which is reserve calculated at the time when the i-th claim arrives (and including this claim).

1) Estimate

$$\mathbb{P}(\text{Ruin before } n\text{-th arrival}) = \mathbb{P}(\min\{R_1, \ldots, R_n\} < 0)$$

using crude Monte Carlo.

N.B. With $T_0 \equiv 0$, then we have $T_i - T_{i-1} \overset{\text{i.i.d.}}{\sim} \mathsf{Exponential}(\lambda)$ for $i = 1, 2, \ldots$

Constants:

```
[ ]: R = 10**6
     u = 1
     p = 0.30
     λ = 0.5
     n = 50
     r = 1
     m = 0.5


     ...
```

2) Conditioned on a bankruptcy event, what is the expected value of the reserve process (it will be negative)?

```
[ ]: ...
```

3) Conditioned on a bankruptcy event, when does it occur? Plot a histogram of the claim number which causes bankruptcy ($\in \mathbb{N}$).

```
[ ]:
```

3 Importance sampling

1) Consider estimating $\mathbb{P}(\max\{X_1, X_2\} > 5)$ where $X_i \overset{\text{i.i.d.}}{\sim} \mathsf{Exponential}(1)$.

Use importance sampling, where each $X_i \overset{\text{i.i.d.}}{\sim} \mathsf{Exponential}(\lambda)$ for some $\lambda \in (0, 1)$.

Do this where λ takes values in a grid of points between 0 and 1. Plot the IS estimates and the IS variances, like in the lecture slides (complete with horizontal line for the true value, and the vertical line at the minimal variance λ).

```
[ ]: R = 10**6
     λs = np.linspace(0.01, 1, 25)
     ellHats = np.zeros_like(λs)
     sigmaHats = np.zeros_like(λs)
```

```
...
```

2) Redo the financial example from the lecture (option pricing with the sum of correlated log-normals) but apply any importance sampling you like. Show the price estimate with a 95% confidence interval, but make sure your choice of importance distribution gives you a smaller confidence interval then crude Monte Carlo.

Here is a crude Monte Carlo estimate. Some of the problem specification constants may be changed (compare to the lecture), use these values.

[ ]:
```
# Problem constants
n = 2
r = 0.05
T = 1
K = 4
ρ = -0.5
σ2 = 1/10
R = 10**6

# Mean vector and covariance matrix
μ = np.arange(1, n+1) / 10
Σ = σ2 * ( (1-ρ) * np.eye(n) + ρ * np.ones(n) )

# Simulating the index value at T
rng = rnd.default_rng(1)
normals = rng.multivariate_normal(μ, Σ, size=R)
Xs = np.exp(normals)
Ss = Xs.sum(axis=1)

# Calculating the MC estimate and CIs
ests = np.exp(-r*T) * np.maximum(Ss - K, 0)
ellHatCMC = ests.mean()
sigmaHatCMC = ests.std()
widthCICMC = 1.96 * sigmaHatCMC / np.sqrt(R)

print(f"CMC option price: {ellHatCMC} (+- {widthCICMC})")
print(f"Fraction of simulated options which payed out: {np.mean(Ss - K >
  ↪0)}")
```

N.B. To calculate pdf of this original distribution at the points in some array `x` use `stats.multivariate_normal.pdf(x, μ, Σ)`

[ ]:
```
...


print(f"IS option price: {ellHat} (+- {widthCI})")
```