

Assignment 3, due September 30, before class

Please follow [instructions on the assignments page](#)

About this assignment: Computational experiments and simulation.

1. (*Sampling by the inverse CDF method*) Consider the force of morbidity model from exercise 2 of homework 2. Write a sampler to generate independent samples T_k by the inverse CDF method from class (or any book on Monte Carlo). Estimate the PDF of the samples from a histogram and plot the histogram vs the actual target density. The agreement should improve as you increase the number of samples. Try with various values of a and b , chosen to have different shapes of PDF. Try to find a sample size so that the code takes a few seconds to run. You don't have to report accurate timings, which would be a bad anyway because accurate timings take care to produce on a laptop where you don't control the operating system. Just get a sense of, and report, how big a sample can be if you have a few seconds of time on your laptop.
2. (*Sampling by simulation*) For this problem we will simulate the death process using small time steps. This is a computer implementation of the model used to derive the PDF of exercise 1. Choose a (small) time step Δt and define times $t_n = n\Delta t$. Define death probabilities in the interval $[t_n, t_n + \Delta t]$ by $\lambda(t_n)\Delta t$. A "person" who has survived to time t_n has this probability of dying at time t_n , which makes $T = t_n$. Start at time $t_0 = 0$ and "march" to times $t_1, t_2, \dots$, until the person dies. This method generates a sample of the random death time. Write a Python function that returns a single sample. The arguments should be $a, b, \Delta t$, and the random number generator object. The return value should be T . Make a histogram of the results to show that the distribution of this T converges to the true distribution from exercise 1 as $\Delta t \rightarrow 0$. For this, you will need to make the bin size significantly larger than Δt or else ... (try it and see...). Maybe the agreement gets better (particularly the histogram if the bins are small) if you choose a random death time in the interval $t_n, t_n + \Delta t]$ if the person dies at time t_n . Try to get an idea how Δt should depend on a and b to get good results. With such a Δt , what sample size can you make in a few seconds? *What you can learn from exercises 1 and 2.* In principle, these are different ways to generate samples from the same random variable. For this problem, the inverse CDF method should be faster. There are more complex simulations that can be sped up and made more accurate by doing some analysis like this.
3. (*Simple statistical estimation*) Write a routine to estimate the average lifespan using the sampler from exercise 1 and n independent samples. Estimate the standard deviation from the sample and find the 1σ error bar. Also estimate the probability that the lifespan is less than 50 (years). Use trial and error to find parameters a and b so that the average lifespan is 80 and the probability of less than 50 is 20%. Do not pick a and b accurately, but "ballpark" numbers that are reasonable for rich countries with good healthcare systems (e.g., Denmark).
4. (*Computational experiment with maximum likelihood*) Generate a fake dataset of n independent samples of the lifespan random variable with the parameters you found in exercise 3. Assume a is fixed (at the known value) and find the maximum likelihood estimate $\hat{b}$. You can use a routine from `scipy` to solve the single nonlinear equation (we will have more on equation solving later). Use the routine that implements Newton's method using the analytical derivative (which we used to derive the central limit theorem for the MLE). Repeat this experiment M times (independently) to estimate the mean and variance of $\hat{b}$. Compare the estimated variance to the theoretical prediction using Fisher information. You can estimate the Fisher information as in

exercise 3 using many independent samples. Choose several n values, some small enough that the deviation from gaussianity is noticeable. What's harder, try to learn the bias in $\hat{b}$ for small n . From theory (which is like the theory we did for the variance but more complicated) we know the bias is order n^{-1} , which makes it hard to detect unless n is small.