julia> Lecture Five: Stochastic Methods

Course: Computational Bootcamp

Name: John Higgins

Date: July 1, 2024

julia> Today's goals

1. Understand a variety of stochastic methods and how to implement them
2. Understand the usefulness of stochastic methods in solving certain difficult problems

julia> Random variables

- Life is filled with heterogeneity and uncertainty
- Our model also must account for this
  - Agents of different types distributed in some way
  - Agents are hit with shocks that affect decisions and outcomes
- We often want to calculate statistics that depend on this randomness:
  - Expected Continuation Values
  - Probability of making a given decision
- Problem:
  - Distributions often don't have simple closed form solutions
  - Calculations of these statistics can be challenging

julia> Solution

- Let our computers calculate numeric approximations of these values
- We can do this by performing simulations from the known random distribution

julia> Example: Birthday Problem

- Question: If there are N people in a room, what is the probability that at least two of them share the same birthday?
  - Assume all days are equally likely
  - Ignore leap years
- Closed form expression exists, but is a messy combination of binomials (yucky)
- Can perform simulations to arrive at a good approximation

julia> Example: Distance of points in a cube

- Question: If two points are chosen at random within a (three dimensional) unit cube, what is the average distance between them?
- Close form solution is an integral in six dimensions:

$$E[D] = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}\, dx_1\, dx_2 \ldots dy_3$$

- Don't know about you, but I don't want to do this :)
- Rather than using pencil or paper, we can use simulations to get a much better approximation

julia> Economic example: college choice

> In period one, the individual chooses to go to college based on expected value:

$$V_1 = \max_{s \in \{0,1\}} E_\varepsilon[V_2(s, \varepsilon)]$$

> In period two, the agent chooses whether to supply labor:

$$V_2 = \max_{s \in \{0,1\}} \{\log(c) + \alpha(1 - \ell)\}$$

$$c = \begin{cases} B, \ell = 0 \\ \exp(\beta_0 + \beta_1 s + \varepsilon, \ell = 1 \end{cases} \qquad \varepsilon \sim N(0, \sigma^2)$$

julia> Why is this hard?

- We need to know the expected continuation value

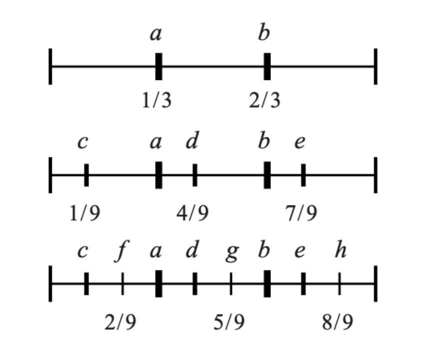
$$V_1 = \max_{s \in \{0,1\}} E_\varepsilon[V_2(s, \varepsilon)]$$

- The shock ε will affect our wage and whether we decide to work or not
  - If ε is too low, we will decide not to work
- Need to solve for the cutoff value ε for each schooling decision, and then take a conditional expectation
- Or we can just ask our computer to do some simulations to approximate the expectation

julia> Non-random sequences

- Simulating random numbers can be a good way to approximate expectations
- But, it requires a large number of simulations to obtain good coverage of the distribution
- Non-random sequences can obtain better coverage of the distribution
- Ex: Halton sequence with base 3:

$$1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9, 1/27, \ldots$$



julia> Why should you care about Halton sequences?

> Halton sequences
  1. Will appear to be random for many purposes
  2. Offer better coverage (i.e. more uniform) than random draws
  3. Require smaller sample size to achieve the same performance as a random sample

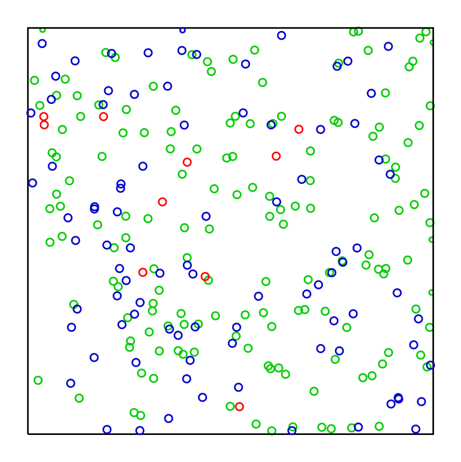
> Example: Bhat (2001) finds that using 100 Halton draws for a mixed logit model *outperforms* taking 1000 standard random draws: lower bias and standard error

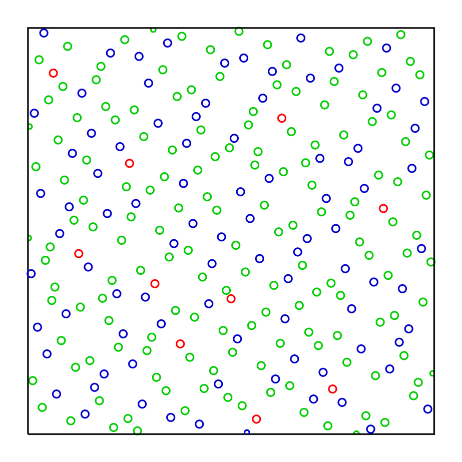
> Tip: when drawing Halton shocks, it is good practice to generate a long sequence, discard the initial values (due to their high correlation), shuffle the values, and draw samples from the remaining values

> For higher dimensional integrals, you'll want to use shuffled/scrambled Halton sequences

julia> Example

- Consider drawing random numbers (x, y) where both are i.i.d. uniformly distributed
- ''Random'' Numbers (left) and Halton Sequences (Right)





julia> Quadrature: Approximating integrals

> We are interested in approximating an integral (such as an expectation):

$$\int_a^b g(x)\,dx$$

> To do this, we choose a set of n nodes (x_i) and weights (w_i) and calculate the following instead:

$$\int_a^b g(x)\,dx \approx \sum_{i=1}^{n} w_i g(x_i)$$

> In essence: take a weighted sum of the function values instead of evaluating the true integral

> The intuition: like the Riemann sums in Calculus, we can approximate an integral as a weighted sum of function values

julia> Quadrature examples

> Okay, so how are these nodes and weights determined?

> The choice of (w_i, x_i) depends on the integral (https://en.wikipedia.org/wiki/Gaussian_quadrature)

> Gauss-Legendre Quadrature:

$$\int_{-1}^{1} g(x)\, dx$$

> Gauss-Hermite Quadrature (good for normal distribution):

$$\int_{-\infty}^{\infty} f(x) \exp(-x^2)\, dx$$

> FastGaussQuadrature.jl provides nodes and weights