Random Numbers and Simulation

- **Generating random numbers:** Typically impossible/unfeasible to obtain truly random numbers
- Programs have been developed to generate pseudo-random numbers:
- Values generated from a complicated deterministic algorithm, which can pass any *statistical test* for randomness
- They *appear* to be independent and identically distributed.
- Random number generators for common distributions are built into R.
- For less common distributions, more complicated methods have been developed (e.g., Accept-Reject Sampling, Metropolis-Hastings Algorithm)
- STAT 740 covers these.

(Monte Carlo) Simulation

Some Common Uses of Simulation

1. Optimization (Example: Finding MLEs)

2. Calculating Definite Integrals (Ex: Finding Posterior Distributions)

3. Approximating the Sampling Distribution of a Statistic (Ex: Constructing CIs)

- 1. Finding the x that maximizes (or minimizes) a complicated function $h(x)$ can be difficult analytically
- Situation even tougher if $\mathbf{x}$ is multidimensional
- Find $\mathbf{x}$ to maximize $h(x_1, x_2, \ldots, x_p)$

OTHER OPTIONS:

- Simple Stochastic Search: If the maximum is to take place over a bounded region, say $[0,1]^p$, then:
  Generate many uniform random observations in that region, plug each into $h(\cdot)$, and pick the one that gives the largest $h(\mathbf{x})$.
- *Advantage:* Easy to program.
- *Disadvantage:* Very slow, especially for multidimensional problems. Requires much computation.

Example: Maximize $h(x_1, x_2) = (x_1^2 + 4x_2^2)e^{1-x_1^2-x_2^2}$ over $[-3, 3]^2$.

More advanced: Gradient Methods, which use derivative information to determine which area of the region to search next.

- Rule: "go up the slope"
- Disadvantage: Can get stuck on *local* maxima

Simulated Annealing: Tries a sequence of $\mathbf{x}$ values: $\mathbf{x}_0, \mathbf{x}_1, \ldots$

- If $h(\mathbf{x}_{i+1}) \geq h(\mathbf{x}_i)$, "move" to $\mathbf{x}_{i+1}$.
- If $h(\mathbf{x}_{i+1}) < h(\mathbf{x}_i)$, "move" to $\mathbf{x}_{i+1}$ with a certain probability which depends on $h(\mathbf{x}_{i+1}) - h(\mathbf{x}_i)$.

R functions that perform optimization

`optim()`

`optimize()` $\leftarrow$ one-dimensional optimization

Example: `optim(par = c(0,0), fn=my.fcn,`
`control=list(fnscale=-1), maxit=100000)`

Nelder-Mead optimization

Other choices: `method="CG"`, `method="BFGS"`, `method="SANN"`

Calculating Definite Integrals

In statistics, we often have to calculate difficult definite integrals (Posterior distributions, expected values)

$$I = \int_a^b h(x)\, dx$$

(here, $\mathbf{x}$ could be multidimensional)

Example 1: Find:

$$\int_0^1 \frac{4}{1+x^2}\, dx$$

Example 2: Find:

$$\int_0^1 \int_0^1 (4 - x_1^2 - 2x_2^2)\, dx_2\, dx_1$$

Hit-or-Miss Method

Example 1:

$$h(x) = \frac{4}{1+x^2}$$

- Determine c such that $c \geq h(x)$ across entire region of interest. (Here, $c = 4$)
- Generate n random uniform (X_i, Y_i) pairs, X_i's from $U[a, b]$ (here, $U[0, 1]$) and Y_i's from $U[0, c]$ (here, $U[0, 4]$)
- Count the number of times (call this m) that the Y_i is less than the $h(X_i)$
- Then $I \approx c(b-a)\frac{m}{n}$

[This is (height)(width)(proportion in shaded region)]

Classical Monte Carlo Integration

$$I = \int_a^b h(x)\,dx$$

- Take n random uniform values $U_1, \ldots, U_n$ (could be vectors) over $[a, b]$

Then

$$I \approx \frac{b-a}{n} \sum_{i=1}^{n} h(U_i)$$

Expected Value of a Function of a Random Variable

Suppose X is a random variable with density f.

Find $E[h(X)]$ for some function h, e.g.,

$$E[X^2]$$

$$E[\sqrt{X}]$$

$$E[\sin(X)]$$

- Note $E[h(X)] = \int_{\mathcal{X}} h(x)f(x)\,dx$ over whatever the support of f is.
- Take n random values $X_1, \ldots, X_n$ from the distribution of X (i.e., with density f)
- Then

$$E[h(X)] \approx \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

Examples

Example 3: If X is a random variable with a $N(10, 1)$ distribution, find $E(X^2)$.

Example 4: If Y is a beta random variable with parameters $a = 5$ and $b = 1$, find $E(-\log_e Y)$.

- Some more advanced methods of integration using simulation (Importance Sampling)
- Note: R function `integrate()` does numerical integration for functions of a *single* variable (*not* using simulation techniques)
- `adapt()` in the "`adapt`" package does multivariate numerical integration

Approximating the Sampling Distribution of a Statistic

To perform inference based on sample statistics, we typically need to know the sampling distribution of the statistics.

Example: $X_1, \ldots, X_n \sim iid\ N(\mu, \sigma^2)$.

$$T = \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

has a $t(n-1)$ distribution.

If σ^2 known,

$$Z = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

has a $N(0,1)$ distribution.

Then we can use these sampling distributions for inference (CIs, hypothesis tests).

What if the data's distribution is not normal?

1. Large sample: Central Limit Theorem
2. Small sample: Nonparametric procedures based on permutation distribution

- If population distribution is known, can approximate sampling distribution with simulation.
- Repeatedly (m times) generate random sample of size n from population distribution.
- Calculate statistic (say, S) each time.
- The empirical distribution of S-values approximates its true distribution.

Example 1: $X_1, \ldots, X_4 \sim Expon(1)$

- What is the sampling distribution of $\bar{X}$?
- What is the sampling distribution of sample midrange?

- What if we don't know the exact population distribution (more likely)?
- Can use *bootstrap methods*: Resample (randomly select n values from the original sample, with replacement). These "bootstrap samples" together mimic the population.
- For each of the, say, m bootstrap samples, calculate the statistic of interest.
- These m values will approximate the sampling distribution.

Example 2: Observe $7, 9, 13, 12, 4, 6, 8, 10, 10, 7$ from an unknown population type.

- Bootstrap sampling built into R in the "`boot`" package.
  Try `library(boot); help(boot)` for details.
- If you know the *form* of the population distribution, but not the parameters, a *parametric* bootstrap can be used.
- Simple bootstrap CIs have some drawbacks
- More complicated "bias-corrected" bootstrap methods have been developed