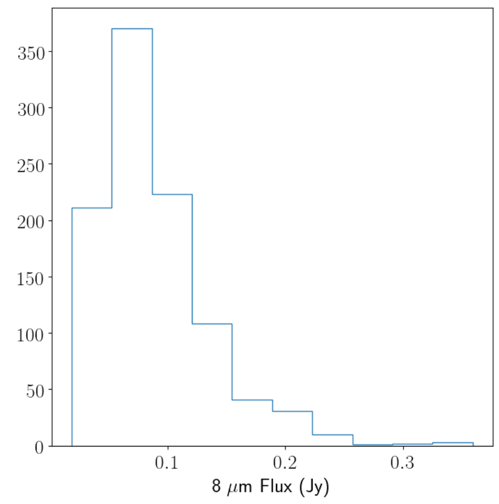
# Statistics for Astronomers: Lecture 20, 2019.05.13

Prof. Sundar Srinivasan

IRyA/UNAM

## Recall: Monte Carlo error propagation

Assume: magnitude $\sim \mathcal{N}(7.27, 0.54^2)$. Draw from this distribution $N = 1000$ times and compute the flux from each draw. Use the resulting distribution to estimate the location and scale parameters.



Distribution is severely skewed.
(Mean, median, mode) = (0.09, 0.08, 0.03) Jy.

Scale has to be evaluated in one of many ways (e.g., equal-tailed interval).

This method is extremely useful when
(a) many errors have to be simultaneously propagated and/or (b) the relationship between the variables is nonlinear (*e.g.*, the blackbody flux in terms of its parameters).

## Recall: Quadrature (Deterministic vs. stochastic)

Based on weighted averages of function evaluations at predetermined points *e.g.*: Trapezoid Rule, Simpson's Rule, Gauss quadrature.
$0^{\text{th}}$ order: $f(x)$ piecewise constant ($w(x_i) = 1$ above). Error $\sim N^{-1}$.
$1^{\text{st}}$ order: $f(x)$ piecewise linear (Trapezoid Rule). Error $\sim N^{-2}$.
$2^{\text{nd}}$ order: $f(x)$ piecewise quadratic (*e.g.*, Simpson's Rule). Error $\sim N^{-4}$.
$d$ dimensions: error $\sim$ (one-dimensional error)$^{1/d}$. More efficient techniques required!

Based on function evaluations at randomly drawn points *e.g.*: Monte Carlo, Markov Chain Monte Carlo.
The simple Monte Carlo method to evaluate $\mathbb{E}[g(x)] = \int dx\ p(x)\ g(x)$ is as follows:
Draw samples of $x$ from $p(x)$, evaluate $g(x)$, and approximate $\mathbb{E}[g(x)]$ with the average of $g(x)$:

$$\mathbb{E}[g(x)] = \int dx\ p(x)\ g(x) \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i).$$

Note that $\int_a^b dx\ g(x) = (b-a)\int_a^b dx\ p(x)\ g(x)$, where $p(x) = U[a,b] = \dfrac{1}{b-a}$.

Error $\sim \sqrt{\dfrac{Var(g(x))}{N}} \propto N^{-1/2}$. Simple Monte Carlo not very efficient! Better than Simpson only for $d > 8$!!

## Recall: Rejection sampling

Rejection sampling samples from a proposal distribution $g(x)$ instead of the target distribution $p(x)$.

$g(x)$ is such that for some $M > 1, f(x) \leq M\ g(x)$.

The general procedure for rejection sampling is as follows

1. Sample an $x$ value from the proposal distribution.
2. For this $x$ value, sample a $y$ value from $U[0, g(x)]$ (that is, find a height that is between zero and the value of the proposal distribution at this $x$ value).
3. If the sampled $y \leq f(x)$ for the corresponding $x$ value, accept this $x$ value. If not, reject it and go back to step 1.

The fraction $\nu = \dfrac{N_{\text{accepted}}}{N_{\text{total}}}$ of accepted values is such that $\int dx\ f(x) = \nu \int dx\ g(x)$.

Another example of rejection sampling: computing the value of $\pi$ using a circle inscribed in a square.

## Problems with rejection sampling

Rejection sampling uses independent draws, and is great for 1- or 2-dimensional problems.

For efficiency, need a good guess for the proposal distribution $g(x)$.

"Curse of dimensionality". *e.g.*, in order to have the same resolution $N$ along one dimension, the total number of points required $\sim N^d$. The probability of rejection increases as $d$ increases. *e.g.*, ratio of hypervolumes of hypersphere inscribed in a hypercube goes to zero as $d$ increases!

Need something better!

## Importance sampling (PSU Astrostatistics Summer School lecture notes)

Simple MC requires large samples for accurately computing $p(\mathrm{rare\ events})$. One solution – give those regions a larger weight so they are sampled more often.
In some situations, it is easier to sample from a proposal distribution $q_X(x)$ instead of sampling from the target distribution $p_X(x)$:

$$\mathbb{E}[f(X)] = \int dx\ p_X(x)\ f(x) = \int dx\ q_X(x)\ \frac{p_X(x)}{q_X(x)}\ f(x) \equiv \int dx\ q_X(x)\ w(x)\ f(x),$$

where $w(x)$ is called the importance weight function. Then,

$$\mathbb{E}_p[f(X)] = \mathbb{E}_q[w(X)f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} w(X_i)f(X_i), \text{ where } X_i \sim q_X(x).$$

Compare to simple Monte Carlo method: $\mathbb{E}_p[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i)$, where $X_i \sim p_X(x)$.

Numerical considerations: for stability, $w$ should be normalised, especially when one or both of $p_X(x)$ and $q_X(x)$ aren't.

The MC estimator for $\mathbb{E}_q[w(X)f(X)]$ is unbiased. For a smart choice of $q_X(x)$, it can also minimise variance. In fact, one of the applications of importance sampling is to reduce the variance in MC estimates.

Expectations over several different distributions $p_{X,1}(x), p_{X,2}(x), \cdots$ can be computed with one sample from $q_X(x)$.

## Importance sampling: example (PSU Astrostatistics Summer School lecture notes)

For $X \sim \mathcal{N}(0,1)$, find $p(X > 5)$ via simple Monte Carlo using $N = 1000$ samples.

$$p(X > 5) = \int_{-\infty}^{\infty} dx\ p_X(x)\ \mathbb{I}_{x>5}(x) = \mathbb{E}_p[\mathbb{I}_{x>5}(x)] \approx 10^{-7}\ (> 5\sigma \text{ event})$$

In a sample of $N = 1000$ points, we therefore expect `np.round(1000*1e-7)` $= 0$ such points! Need $N \gtrsim 10^7$ samples for accuracy!

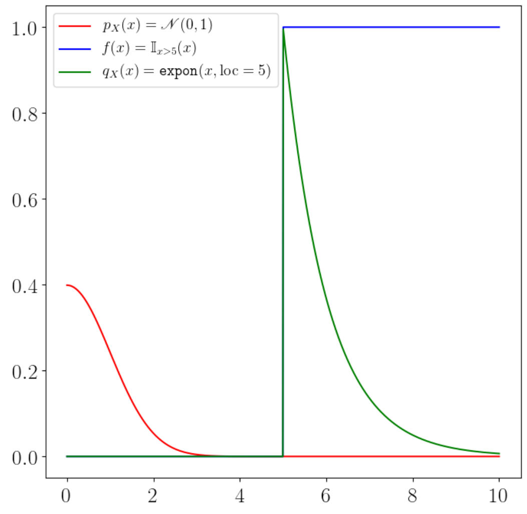
### Using importance sampling:

Step 1: Pick $q(x)$ such that it "enhances" the tail region. Since direction matters, choose $q(x)$ to be the exponential distribution with location $x = 5$.

Step 2: Draw $N = 1000$ variates from $q$ instead of $p$:
`x = scipy.stats.expon.rvs(loc = 5, size = 1000)`.

Step 3: Compute the average of $w(X)f(X)$:
`expect = (norm.pdf(x)/np.exp(5-x)).mean()`
(I get $\approx 2.91 \times 10^{-7}$).

Step 4: Verify with exact answer:
`print(1 - norm.cdf(5))`
(I get $\approx 2.87 \times 10^{-7}$).



## Some astronomical papers using importance sampling

Estimation of cosmological parameters:
Lewis & Bridle 2002, https://arxiv.org/abs/astro-ph/0205436
Trotta 2008, https://arxiv.org/abs/0803.4089

X-ray luminosity plane:
Gallo et al. (2018), http://adsabs.harvard.edu/abs/2018MNRAS.478L.132G

Extrasolar planet modelling:
Ford 2005, https://arxiv.org/abs/astro-ph/0512634.
Nelson et al. 2018, http://adsabs.harvard.edu/abs/2018arXiv180604683N.
Hsu et al. 2018, http://adsabs.harvard.edu/abs/2018AJ....155..205H.
Rajpaul et al. 2017, http://adsabs.harvard.edu/abs/2017MNRAS.471L.125R.

## The story so far

We use Monte Carlo methods in order to either sample from a distribution or compute an expectation value of a function over a distribution.

Simple MC: $\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i)$, where $X_i \sim p_X(x)$.

Problem: $p_X(x)$ may be too complicated (esp. multidimensional), and/or difficult to sample from.
Solution: rejection sampling, importance sampling – sample from a proposal distribution instead of the target distribution.

Problem: "curse of high dimensionality" – the proposal needs to be as close as possible to the target; as $d$ increases, the discrepancy increases exponentially.
Solution: Markov Chain Monte Carlo (MCMC); explore multidimensional parameter space by sampling ("travelling") along regions/zones of high probability.

## Markov Chains

Refresher: Stochastic/random process - a family/sequence of random variables.
State space - set of all possible values attained by the random variables.

In general, in a random process $\{X_0, X_1, \cdots, X_n\}$, $p(X_{n+1} = x_{n+1})$ depends on the values attained by $X_0, X_1, \cdots, X_n$.

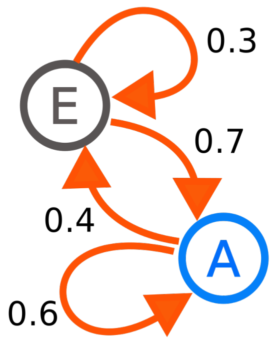
Memorylessness (Markov property): $p(X_{n+1}|X_n, \cdots, X_0) = p(X_{n+1}|X_n)$ (Given present, future is independent of the past).
The variables $X_0, X_1, \cdots, X_{n+1}$ then form a Markov Chain of order 1, and the family $\{X_0, X_1, \cdots, X_{n+1}\}$ is a Markov process.

Because of the Markov property, Markov Chains consist of dependent variables. These are useful in describing, *e.g.*, time-series data.

Members of Markov Chains can be indexed by discrete or continuous variables ("discrete-time" or "continuous-time" chains), and can attain discrete or continuous values ("discrete-space" or "continuous-space" chains. Space here refers to the state space).

Daily weather: discrete-time chain. Poisson process: continuous-time chain. Brownian motion, stock prices: continuous-space chains. Simplification of Brownian motion into discrete-space, discrete-time chain: random walk.

## Example: two-state discrete Markov Chain



(credit: User:Joxemai4/CC BY-SA 3.0)

$p(X_{j+1} = E|X_j = E) = 0.3,$
$p(X_{j+1} = A|X_j = E) = 0.7,$
$p(X_{j+1} = E|X_j = A) = 0.4,$
$p(X_{j+1} = A|X_j = A) = 0.6.$

These probabilities form the transition matrix (or, for the continuous case, kernel) for the system, such that $X_{j+1} = X_j T$
($T$ acts on the current state to give the future state).

## Markov Chains: some definitions

A time-homogeneous MC is such that the probability of a given transition is independent of time. That is, $p(X_{j+1} = b|X_j = a) = T_{ab}$ independent of $j$.

A MC is irreducible if, given enough time (steps), it is possible to get to any state starting from any other state. That is, $\exists$ some $j > 0$ such that $p(X_{j+1} = b|X_0 = a) > 0$.

A distribution $\pi$ on the state space $S$ is stationary w.r.t. the transition matrix $T$ if $\pi T = \pi$ ($\pi$ is a left-eigenvector of $T$ with eigenvalue 1). In other words, once the system attains state $\pi$, it stays there.

### Ergodic Theorem for Markov Chains

If $(X_0, X_1, \cdots, X_n)$ is an irreducible Markov Chain with stationary distribution $\pi$, then $\frac{1}{N}\sum_{i=0}^{N} f(X_i) \xrightarrow[N\to\infty]{} \mathbb{E}[f(X)]$; that is, $f(X)$ averaged over the Markov Chain is an unbiased estimate of the expectation of $f(X)$.

In addition, if the MC is aperiodic, then $P(X_n = x|X_0 = x_0) \xrightarrow[N\to\infty]{} \pi(x)$ irrespective of $x_0$.

## Markov Chain Monte Carlo

The purpose is to generate draws from a target distribution $p_X(x)$. Algorithms are framed in such a way that the Markov process asymptotically approaches a unique stationary distribution $\pi(x)$ such that $\pi(x) = p_X(x)$.

After $N$ steps (iterations), $\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i)$, where $X_i$ is the states explored at step $i$. As $N$ increases, the average converges to $\mathbb{E}[f(X)]$ due to the Ergodic Theorem.

Markov Chain: the decision as to where to advance in parameter space depends only on the current location. The next "link" in the chain is decided using a jump distribution.

Monte Carlo: Pseudorandom numbers are generated in order to sample the target distribution.

Dependent sampling: Future step depends on present step.

The algorithm for each iteration:

1. Select starting point/state (parameter value $\theta_0$).
2. Evaluate unnormalised posterior probability at this point.
3. Draw new parameter value $\theta_{j+1}$ from a proposal distribution.
4. Evaluate unnormalised posterior probability for this value.
5. Decide whether you will accept the new value.

## Metropolis-Hastings Algorithm

One of the oldest MCMC implementations.

1. Select starting point/state (parameter value $\theta_j$).
2. Evaluate unnormalised posterior probability at this point.
3. Draw new parameter value $\theta_{j+1}$ from a proposal distribution ("jump distribution" centered on current value).
   For the Metropolis algorithm, the jump distribution must be symmetric: $p(\theta_{j+1}|\theta_j) = p(\theta_j|\theta_{j+1})$. Usually, $\theta_{j+1} \sim \mathcal{N}(\theta_j, \sigma^2)$, with $\sigma$ the characteristic "step size". The results may depend on $\sigma$ (small = high acceptance rate but more iterations required, and vice versa).
4. Evaluate unnormalised posterior probability for this value.
5. Decide whether you will accept the new value accept with probability $\alpha = p(\theta')/p(\theta_j)$.
   This is implemented by comparing $\alpha$ to a uniform random variable $u \sim U(0,1)$. If $\alpha > u$, the new value is accepted. If not, the old value is retained.
   This is because $p(\alpha \geq u) = p(u \leq \alpha) \equiv F_u(\alpha) = \alpha$ for $U(0,1)$.

## Look stuff up!

AstroML implementation:
https://github.com/astroML/astroML/blob/master/astroML/plotting/mcmc.py

`emcee` (“The MCMC Hammer”): http://dfm.io/emcee/current/

Example of article that uses `emcee` to determine the most-probable combinations of parameters of a modified blackbody model for circumstellar dust around AGB stars: Dharmawardena et al. 2018 (https://arxiv.org/abs/1805.10599).