View options

Show slides

Show questions

Show responses

Response

Submitted

Saved

```
print(np.mean(X, axis=0))
```

What is the expected runtime of this algorithm? Well, it will be the number of 1s in the array, plus one, unless we have X_n =1, in which we case we will know that we don't have to generate an additional geometric random variable.

So this has expectation: $E[\sum X_i + 1|X_n = 0]P(X_n = 0) + E[\sum X_i|X_n = 1]P(X_n = 1) = p(n-1) + 1 = pn + (1-p)$. This suggests a small optimization: if $p < 0.5$, we proceed as before; if $p > 0.5$, use q = (1-p) as the parameter for the geometric random variable, and swap the 1s and 0s in the above algorithm.

Sampling a Binomial Random Variable

In the last section, we showed a method for generating n independent Bernoulli(p) random variables. Clearly, we can use the method to generate a Binomial(n,p) by taking their sum.

However, we could also directly exploit the inverse cdf transformation to sample Binomials. There is, of course, no closed-form formula for the binomial cdf, so we will have to do something like this:

- Step 1: $U \sim Unif(0, 1)$
- Step 2: $i = 0$, let $pr = \binom{n}{i} p^i * (1-p)^{n-i}$, F = pr
- Step 3: If $U \leq F$ return X = i
- Step 4: $i = i + 1$, $pr = pr + \binom{n}{i} p^i * (1-p)^{n-i}$, $F = F + pr$
- Step 5: go to step 3

Using this method, we make X + 1 comparisons, giving us an expected number of iterations of $np + 1$.