2DI90
Probability &
Statistics
Computer Simulations

The knowledge and tools we developed so far allow us to construct models for many processes where one can use randomness as a very useful building block:

• Modeling the arrival of customer requests to a webserver (Poisson process)
• Modeling the cloud coverage/rain
• Modeling the propagation of a virus/worm in a network

Sometimes these models are very complex, and the only way to say something about them is by simulation (the LLN and the CLT play a key role here).

Finally, simulations can provide valuable qualitative insight about the models devised - you can “see” the model in action, and assess if it appears to behave as the real world.
Computer Simulations

The main purpose of simulation is to compute quantities of interest whose direct computation is either complicated, risky, time consuming and expensive, or impossible.

**Historical Example:** To develop safe nuclear energy (and unfortunately also nuclear bombs) it was necessary to study physical branching processes (modeling chain reactions). This was virtually impossible to do analytically, and computers were used to evaluate critical masses, and other quantities of interest.
The MonteCarlo Method

From the Law of Large Numbers we know that probabilities can often be evaluated as long-run averages of independent random variables. Therefore if we can generate many samples from the appropriate random variables we can easily compute probabilities. This is the basic idea behind MonteCarlo methods!

The name of these methods comes from Europe’s most famous casino, in Monaco

MonteCarlo was the codename chosen by the researchers of the Manhattan project for a broad class of simulation techniques...

Gambling always involves long-run averages of wins and losses, and it has always attracted the interest of mathematicians, who developed many optimal gambling strategies...
The MonteCarlo Method

The class of methods known as MonteCarlo is rather broad, as consequently there are many researchers that have been associated with their creation. However, one can single out a few names:

Simulation of Random Variables

The first step to be able to simulate complex processes is to generate single independent random variables from an arbitrary distribution. This is going to be our focus for now.

Most statistics software have build-in random number generators able to return samples from a wide range of common random variables. However, in most computer languages you can only generate uniform random variables. It turns out this is sufficient to generate any random variable you might want!

A good, albeit advanced, reference on this topic is the book by Luc Devroye “Non-Uniform Random Variate Generation”, published by Springer and available free of charge at

http://luc.devroye.org/rnbookindex.html
Random Number Generators

Generating truly independent random numbers using a computer is not a trivial task – computers are deterministic machines!!! Typically one generates sequences of pseudo-random numbers.

**Definition:** Sequence of Pseudo-Random Numbers
A sequence of pseudo-random numbers is a deterministic sequence of numbers that “looks” and “feels” like a sequence of independent random variables. Furthermore this sequences should pass the most common randomness tests (e.g. Kolmogorov test).

**Example:**
\[ X_1, X_2, \ldots \] computed recursively by
\[ X_n = (a_1 X_{n-1} + a_2) \mod a_3 , \]
for carefully choosen \( a_1, a_2, a_3 \) (group theory informs this choice).

The choice of “seed” \( X_1 \) is the only place where randomness might enter the picture (e.g. choose \( X_1 \) according to the computer timer when you start the simulation).
Random Number Generators

Obviously the sequence in the example is not really random, but it “looks” like a random sequence. In most software packages this is how “random” numbers are generated, and you can choose the seed in any way you like. This is enough for most purposes...

... but if you really want something more sophisticated you can use a device that measures some physical quantity considered to be random (e.g. thermal noise).

We will assume for now on that we have a random number generator \texttt{rand()} capable of generating independent samples from a continuous uniform distribution over \([0,1]\) (each call to the function returns an independent sample).
Discrete Random Variables - Bernoulli

Suppose we want to generate i.i.d. samples from an arbitrary discrete random variable using the random number generator at hand. Let’s start simple:

Example: Simulating a Bernoulli random variable

Let $U \sim \text{Unif}(0,1)$. We can transform $U$ so that the result is a Bernoulli random variable:

$$X = \begin{cases} 1 & \text{if } U < p \\ 0 & \text{if } U \geq p \end{cases}$$

$$P(X = 1) = P(U < p) = \int_0^p 1\,du = p$$

$$P(X = 0) = P(U \geq p) = \int_p^1 1\,du = 1 - p$$

function Bernoulli(p)
    if rand() < p
        return 1
    else
        return 0;
end

(Pseudo-code)
Discrete Random Variables - Binomial

As i.i.d. Bernoulli random variables are the building block of many other random variables we can just use the function we devised to generate other random variables:

Example: Simulating a Binomial random variable (Bin(n,p))

Suppose we want to generate samples from a binomial random variable with parameters $n$ and $p$. This random variable has the same distribution as the sum of $n$ Bernoulli random variables therefore we can use the following code:

```python
function Binomial(n,p)
    x=0;
    for k=1 to n
        x=x+Bernoulli(p);
    endfor
    return x;
```
Geometric and Negative Binomial

Since geometric/negative binomial random variables are just the time until the first/$r^{th}$ success of a sequence of Bernoulli trials it is also easy to generate such random variables.

Example: Simulating geometric random variable (Geom($p$))

```
function Geom(p)
    x=1;
    while Bernoulli(p)==0
        x=x+1;
    endwhile;
    return x;
```

Example: Simulating negative binomial r.v. (NegBin($r,p$))

```
function NegBin(r,p)
    x=0;
    for k=1:r
        x=x+Geom(p);
    endfor;
    return x;
```
Arbitrary Discrete Distributions

The technique used for the Bernoulli distribution can easily be generalized to arbitrary discrete random variables. For the Bernoulli distribution we divided the interval [0,1] in two parts (as Bernoulli random variables take only two values). In general we need to break the interval in more parts.

Let $X$ be an arbitrary discrete random variable taking values $x_1, x_2, \ldots$ with probability respectively $p_1, p_2, \ldots$

$$p_i = P(X = x_i), \quad i = 1, 2, \ldots .$$

Recall that $\sum_i p_i = 1$, so we can partition the unit interval into smaller intervals of length $p_i$. 
Arbitrary Discrete Distributions

Let $X$ be an arbitrary discrete random variable taking values $x_1, x_2, \ldots$ with probability respectively $p_1, p_2, \ldots$

\[ p_i = P(X = x_i), \quad i = 1, 2, \ldots \]

\[ A_1 = [0, p_1] \]
\[ A_2 = [p_1, p_1 + p_2] \]
\[ A_3 = [p_1 + p_2, p_1 + p_2 + p_3] \]
and so on...

Now let $U \sim \text{Unif}(0, 1)$ and

\[
X = \begin{cases} 
  x_1 & \text{if } U \in A_1 \\
  x_2 & \text{if } U \in A_2 \\
  x_3 & \text{if } U \in A_3 \\
  \vdots & \text{and so on...}
\end{cases}
\]

Clearly $P(X = x_i) = P(U \in A_i) = p_i$, and so we have the desired distribution for $X$. 
Arbitrary Discrete Distributions

Example: Generating a Poisson r.v. with parameter $\lambda$ ($\text{Poisson}(\lambda)$)

Recall that if $X \sim \text{Poisson}(\lambda)$ then

$$P(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}, \quad k \in \mathbb{N}_0$$

function Poisson(lambda)
    u=rand();
    cp=exp(-lambda);
    k=0;
    factorial_k=1;
    while cp<u
        k=k+1;
        factorial_k=factorial_k*k;
        cp=cp+exp(-lambda)*lambda^k/factorial_k;
    endwhile;
    return k;

Note that here we use only one sample from the uniform distribution, unlike the code we used for the negative binomial for instance.
The Inverse Probability Transform

To generate samples from continuous distributions we will essentially use a similar technique. First we state a what is perhaps a surprising result! Let’s first state a simple version of the result:

**Proposition:**

Let $F$ be a continuous cumulative distribution function (c.d.f.) and assume $F$ is invertible over the interval $(0,1)$. That is, there is a function $F^{-1}$ such that for $x \in (0,1)$

$$F(F^{-1}(x)) = x \quad \text{and} \quad F^{-1}(F(x)) = x.$$ 

(i) If $U \sim \text{Unif}[0,1]$ then $X = F^{-1}(U)$ is a random variable with cumulative distribution function $F$.

(ii) The random variable $F(X)$ is uniformly distributed on $[0,1]$. 
The Inverse Probability Transform

Proof: Let $X = F^{-1}(U)$. Note that for any $x \in \mathbb{R}$

$$
P(X \leq x) = P(F^{-1}(U) \leq x)
= P(F(F^{-1}(U)) \leq F(x))
= P(U \leq F(x))
= F(x)
$$

So $X$ has cumulative distribution $F$.

The second statement follows by the same type of argument

$$
P(F(X) \leq u) = \begin{cases} 
0 & \text{if } u \leq 0 \\
1 & \text{if } u \geq 1 \\
?? & \text{if } 0 < u < 1
\end{cases}
$$

For $0 < u < 1$

$$
P(F(X) \leq u) = P(F^{-1}(F(X)) \leq F^{-1}(u))
= P(X \leq F^{-1}(u))
= F(F^{-1}(u))
= u
$$

so $F(X)$ has the c.d.f. of a uniform continuous distribution over $[0, 1]$.
Generating Continuous Random Variables

To generate continuous random variables all we need is to be able to compute the inverse c.d.f. (not always an easy task). Let’s see an example.

**Example:** Simulating an exponential r.v. (Exp($\lambda$))

Recall that for an exponential random variable the probability density function (p.d.f.) is

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ \lambda e^{-\lambda x} & \text{if } x \geq 0 \end{cases}$$

Therefore

$$F(x) = \int_{-\infty}^{x} f(t) dt = \begin{cases} 0 & \text{if } x < 0 \\ 1 - e^{-\lambda x} & \text{if } x \geq 0 \end{cases}$$

To apply the inverse transform method we need to invert $F(x)$, that is, we need to solve for $x$ the equation

$$u = F(x)$$
Exponential Random Variables

\[ F(x) = \int_{-\infty}^{x} f(t) \, dt = \begin{cases} 0 & \text{if } x < 0 \\ 1 - e^{-\lambda x} & \text{if } x \geq 0 \end{cases} \]

\[ u = F(x) \iff u = 1 - e^{-\lambda x} \]
\[ \iff 1 - u = e^{-\lambda x} \]
\[ \iff \ln(1 - u) = -\lambda x \]
\[ \iff -\frac{1}{\lambda} \ln(1 - u) = x \]

Using the proposition we can safely say that

\[ X = -\frac{1}{\lambda} \ln(1 - U) \]

is a random variable with exponential distribution with parameter \( \lambda \).

**Remark:** Note that \( 1 - U \) is also a uniform random variable over the range \([0, 1]\), therefore

\[ X_1 = -\frac{1}{\lambda} \ln(U) \]

is also an exponential random variable with parameter \( \lambda \).
Exponential Random Variables

Pseudo-code: function Exponential(lambda)
    return -1/lambda \ln(rand())

Example: Simulating an Erlang r.v.'s

Recall that the Erlang random variable with parameters \( \lambda > 0 \) and \( r \in \mathbb{N} \) can be obtained by summing \( r \) independent exponential random variables. With that in mind we can generate Erlang random variables with the following pseudo-code

```plaintext
function Erlang(lambda,r)
    x=0;
    for k=1:r
        x=x+Exponential(lambda);
    endfor
    return x
```
The Inverse Probability Transform*

The inverse probability transform result is actually more powerful, and can be used to generate random variables with arbitrary distributions.

**Theorem:**

Let $F$ be an arbitrary cumulative distribution function (c.d.f.) and define the generalized inverse as

$$F^{-1}(u) = \inf\{x : F(x) \geq u, 0 < u < 1\}.$$ 

(i) If $U \sim \text{Unif}[0, 1]$ then $X = F^{-1}(U)$ is a random variable with cumulative distribution function $F$.

(ii) If $F$ is a continuous c.d.f. then $F(X)$ is uniformly distributed on $[0, 1]$.

* - this result and proof is included in the slides for your reference, but is not covered in the course.
Proof*

Note that, for any $x \in \mathbb{R}$

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(\inf\{y : F(y) \geq U\} \leq x)$$

We will check that the statements

$$\inf\{y : F(y) \geq u\} \leq x \quad \text{and} \quad u \leq F(x)$$

are equivalent:

(i)

$$u \leq F(x) \Rightarrow x \in \{y : F(y) \geq u\} \quad \text{(as } F \text{ is monotone non-decreasing)}$$

$$\Rightarrow \inf\{y : F(y) \geq u\} \leq x$$

(ii)

$$\inf\{y : F(y) \geq u\} \leq x \Rightarrow x \in \{y : F(y) \geq u\}$$

$$\Rightarrow F(x) \leq u$$

* - this result and proof is included in the slides for your reference, but is not covered in the course
Proof*

Note that, for any $x \in \mathbb{R}$

\[
P(X \leq x) = P(F^{-1}(U) \leq x)
= P(\inf\{ y : F(y) \geq U \} \leq x)
= P(U \leq F(x))
= F(x)
\]

So $X = F^{-1}(U)$ has cumulative distribution $F(\cdot)$.

The proof of the second statement uses the equivalence proved in the previous slide. Also, since $F$ is continuous its range contains the interval $(0, 1)$ and for $u \in (0, 1)$ we have $F(F^{-1}(u)) = u$.

\[
P(F(X) \leq u) = P(\inf\{ y : F(y) \geq u \} \geq X)
= P(F^{-1}(u) \geq X)
= P(X \leq F^{-1}(u))
= F(F^{-1}(u))
= \begin{cases} 
0 & \text{if } u < 0 \\
1 & \text{if } u > 1 \\
u & \text{if } 0 \leq u \leq 1 
\end{cases}
\]

* - this result and proof is included in the slides for your reference, but is not covered in the course
The Rejection Method

Sometimes it is not possible to compute the easily C.D.F., and therefore computing its inverse is likely to be even harder. However, we often have access to the density of the random variable. In that case we can use what is known as the rejection method.

**Theorem:**

Let a pair of random variables \((X, Y)\) have uniform distribution over the region

\[
A = \{(x, y) : 0 \leq y \leq f(x)\},
\]

for some probability density function \(f\). Then \(X\) is a random variable with density \(f\).
Proof *

Note that the set $A$ has area 1, because $f$ is a probability density function:

$$\int \int_A 1 \, dx \, dy = \int \int_0 f(x) \, 1 \, dy \, dx = \int f(x) \, dx = 1.$$  

So we conclude that the joint density of $(X, Y)$ is simply

$$f_{XY}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in A \\ 0 & \text{otherwise} \end{cases}$$

The marginal density of $X$ is, as we have seen before

$$f_X(x) = \int f_{XY}(x, y) \, dy = \int_0^1 f(x) \, 1 \, dy = f(x),$$

concluding the proof.

* - This proof is not covered in the course
The Rejection Method

If we want to generate a random variable $X$ with density $f$ we can use the previous result. Let’s make an extra assumption that the density $f$ is bounded and has finite support (meaning the set where the density if different than zero is bounded). In other words

Assume we have numbers $a, b$ and $c$ such that

(i) $0 \leq f(x) \leq c$ for all $x \in \mathbb{R}$

(ii) $f(x) = 0$ if $x < a$ or $x > b$

All we have to do now is to generate points uniformly at random in the purple area!!!
The Rejection Method

(i) Find $a, b, c$ such that for $a \leq x \leq b$ we have $0 \leq f(x) \leq c$ and for all other values of $x$ we have $f(x) = 0$.

(ii) Generate two independent standard uniform random variables $U$ and $V$.

(iii) Define $X = a + (b - a)U$ and $Y = cV$. Note that $(X, Y)$ are uniform over the rectangle in the figure below.

(iv) If $Y > f(X)$ reject the sample and return to step (ii). Otherwise return the sample $X$.

It is easy to see that the sample we return has the desired distribution, and that the algorithm will eventually return a sample (with probability 1).
Example - The Beta Distribution

Beta random variables are particularly useful for inference using Bayesian Networks. The corresponding distribution has two parameter $\alpha > 0$ and $\beta > 0$. It takes values in the interval $[0, 1]$ and has density

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1 - x)^{\beta-1} \quad \text{for } 0 \leq x \leq 1.$$ 

Clearly $f(x) \leq \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$ and so we can use the following pseudo-code to generate samples from the Beta distribution.

```plaintext
function Beta(alpha, beta)
    a=0; b=1; c=Gamma(alpha+beta)/Gamma(alpha)/Gamma(beta);
    repeat
        u=rand();
        v=rand();
        x=a+(b-a)*u;
        y=c*v;
        until y<=Gamma(alpha+beta)/Gamma(alpha)/Gamma(beta)...
            *x^(alpha-1)*(1-x)^(beta-1);
    return x
```
Remarks

The rejection method can be made much more general, and it is able to deal with unbounded densities with unbounded support. It can also be used to generate random vectors efficiently.

It forms the basis of many Monte-Carlo approaches, and the famous Metropolis-Hastings algorithm, widely used to solve complicated statistical problems involving weather prediction (think [www.buienradar.nl/](http://www.buienradar.nl/)). You can learn more about these in Luc Devroye’s book referenced a few slides back.

Now that we can generate samples from random variables we can start using those to solve some problems !!!
Estimating Probabilities

This is one of the most common problems one needs to address. What is the probability of a certain event (e.g. what is the probability the cache of the router is full)?

Let $X$ be a random variable, to estimate $p = P(X \in A)$ where $A$ is an arbitrary set. Suppose we can generate $n$ independent and identically distributed samples from the distribution of $X$, denoted by $X_1, \ldots, X_n$. The Law of Large Numbers tells us that

$$\hat{p} = \frac{\text{number of } i \text{'s such that } X_i \in A}{n},$$

is a good estimator of $p$, in the sense that, as $n$ grows

$$\hat{p} \to p.$$ 

How large should we pick $n$ so that we get an accurate estimate?
Estimating Probabilities

Note that \( Y = n\hat{p} \) is a binomial random variable \( Y \sim Bin(n, p) \). As we have seen

\[
\mathbb{E}[Y] = np \quad \text{and} \quad V(Y) = np(1 - p) .
\]

This means that \( \hat{p} \) is such that

\[
\mathbb{E}[\hat{p}] = p \quad \text{and} \quad V(\hat{p}) = \frac{p(1 - p)}{n} .
\]

Clearly the variability of \( \hat{p} \) is decreasing fast, indicating that \( \hat{p} \) should be getting closer to \( p \) with high probability. To get a hold on this we need some extra results.

**Theorem: Chebyshev’s inequality**

Let \( X \) be an arbitrary random variable, with mean \( \mathbb{E}[X] = \mu \). Then, for \( \varepsilon > 0 \),

\[
P(|X - \mu| \geq \varepsilon) \leq \frac{V(X)}{\varepsilon^2} .
\]
Estimating Probabilities

**Theorem: Chebyshev’s inequality**

Let $X$ be an arbitrary random variable, with mean $\mathbb{E}[X] = \mu$. Then, for $\varepsilon > 0$,

$$P(|X - \mu| \geq \varepsilon) \leq \frac{\text{Var}(X)}{\varepsilon^2}.$$  

Applying this result to $\hat{p}$ we get

$$P(|\hat{p} - p| \geq \varepsilon) \leq \frac{p(1 - p)}{n\varepsilon^2} \leq \frac{1}{4n\varepsilon^2},$$

where the last inequality follows from the fact that $p(1 - p) \leq 1/4$.

Unfortunately this bound is too conservative most of the times, and for this particular case a better bound exists:

**Theorem: Chernoff Bound/Hoeffding’s Ineq.**

$$P(|\hat{p} - p| \geq \varepsilon) \leq 2e^{-2n\varepsilon^2}, \varepsilon > 0.$$
Choosing the simulation sample size $n$

**Theorem:** Chernoff Bound/Hoeffding’s Ineq.

$$P(|\hat{p} - p| \geq \varepsilon) \leq 2e^{-2n\varepsilon^2}, \varepsilon > 0.$$  

Suppose we want to guarantee that with probability larger than $1 - \alpha$ the error of our Monte-Carlo estimate is smaller than $\varepsilon$. All we need to do is solve for $n$ the inequality

$$2e^{-2n\varepsilon^2} \leq \alpha.$$  

This implies that

$$n \geq \frac{1}{2\varepsilon^2} \ln \left( \frac{2}{\alpha} \right).$$

It is important to keep in mind that using a MonteCarlo method you’ll always incur on a random error that can be very large, although this event happens with very small probability!!!
Choosing \( n \) using the CLT

The Chernoff bound is still conservative, and the Central Limit Theorem can be much better, albeit being approximate...

Recall the approximation of a Binomial random variable by a normal distribution (for large \( n \)). Since \( n\hat{p} \sim \text{Bin}(n,p) \) we know that

\[
\frac{n\hat{p} - np}{\sqrt{np(1-p)}} = \frac{\hat{p} - p}{\sqrt{p(1-p)/n}} \approx N(0,1).
\]

This implies that

\[
P(|\hat{p} - p| \geq \epsilon) = P(\hat{p} - p > \epsilon \text{ or } \hat{p} - p < -\epsilon)
\]

\[
= P(\hat{p} - p > \epsilon) + P(\hat{p} - p < -\epsilon)
\]

\[
= P\left(\frac{\hat{p} - p}{\sqrt{p(1-p)/n}} > \frac{\epsilon}{\sqrt{p(1-p)/n}}\right)
\]

\[
+ P\left(\frac{\hat{p} - p}{\sqrt{p(1-p)/n}} < -\frac{\epsilon}{\sqrt{p(1-p)/n}}\right)
\]

\[
\approx P\left(Z > \frac{\epsilon}{\sqrt{p(1-p)/n}}\right) + P\left(Z < -\frac{\epsilon}{\sqrt{p(1-p)/n}}\right)
\]

33
Choosing $n$ using the CLT

Therefore

$$P(|\hat{p} - p| \geq \epsilon) \approx P\left(Z > \frac{\epsilon}{\sqrt{p(1-p)/n}}\right) + P\left(Z < -\frac{\epsilon}{\sqrt{p(1-p)/n}}\right)$$

$$= \ 1 - \Phi\left(\frac{\epsilon}{\sqrt{p(1-p)/n}}\right) + \Phi\left(-\frac{\epsilon}{\sqrt{p(1-p)/n}}\right)$$

$$= \ 2\Phi\left(-\frac{\epsilon}{\sqrt{p(1-p)/n}}\right)$$

To choose $n$ so that with probability $1 - \alpha$ the error of our Monte-Carlo estimate is smaller than $\epsilon$ we need to solve for $n$ the inequality

$$2\Phi\left(-\frac{\epsilon}{\sqrt{p(1-p)/n}}\right) \leq \alpha.$$
Choosing $n$ using the CLT

To choose $n$ so that with probability $1 - \alpha$ the error of our Monte-Carlo estimate is smaller than $\epsilon$ we need to solve for $n$ the inequality

$$2\Phi\left(-\frac{\epsilon}{\sqrt{p(1-p)/n}}\right) \leq \alpha.$$

Let $z_\alpha$ be the unique number such that $\Phi(z_\alpha) = 1 - \alpha$ (this is called the $\alpha$-quantile). Notice also that $\Phi(-z_\alpha) = \alpha$.

Then

$$n \geq p(1 - p) \left(\frac{z_\alpha/2}{\epsilon}\right)^2.$$

Since we don't know $p$, it seems that we might have a problem, but if we notice that $p(1 - p) \leq 1/4$ for any $0 \leq p \leq 1$ then can simply take

$$n \geq \left(\frac{z_\alpha/2}{2\epsilon}\right)^2.$$

Provided this bound is larger than 20 or so the approximation is very accurate...
Example

Let’s use what we have learned to estimate the area of the unit circle.

Let \((X, Y)\) be independent uniform random variables over the set \([-1, 1]^2\). Since the area of \([-1, 1]^2\) is 4 the joint density of \(X\) and \(Y\) is simply

\[
f_{XY} = \begin{cases} 
\frac{1}{4} & \text{if } x \in [-1, 1]^2 \\
0 & \text{otherwise}
\end{cases}.
\]

Let

\[
A = \{(x, y) : x^2 + y^2 \leq 1\}
\]

be the unit circle.

Then

\[
P((X, Y) \in A) = \int \int_A f_{XY}(x, y)\,dy\,dx = \frac{1}{4} \times \text{Area of the unit circle}.
\]
Example

```c
function Pi(n)
    c=0;
    for k=1:n
        x=-1+2*rand();
        y=-1+2*rand();
        if (x^2+y^2<=1)
            c=c+1;
        endif
    endfor
    return 4*c/n;
```

Area \(= 4P((X,Y) \in A) = \pi??\)

How large should \(n\) be so that with probability 0.99 we have the value of the area accurate to the 2\(^{th}\) decimal place?

Error in the area should be smaller than 0.005 with probability \(1-\alpha\), where \(\alpha = 0.01\). This implies that the error in \(\hat{p}\) should be smaller than \(\epsilon = 0.005/4\). Using our previous derivation we should take

\[
n \geq \frac{1}{2\epsilon^2} \ln \left( \frac{2}{\alpha} \right) = 1695461.6
\]
Example

How large should $n$ be so that with probability 0.99 we have the value of the area accurate to the $2^{\text{th}}$ decimal place?

$$n \geq \frac{1}{2\epsilon^2} \ln \left( \frac{2}{\alpha} \right) = 1695461.6$$

(From Chernoff’s bound, with $\alpha = 0.01$ and $\epsilon = 0.005/4$)

We can also try the normal approximation. In this case $z_{0.005} = 2.576$ and so

$$n \geq \left( \frac{z_{\alpha/2}}{2\epsilon} \right)^2 = 1061724.16,$$

which is a bit smaller.

This is obviously not the best way to estimate the area of the unit circle, but serves as illustration of the method.
A Real Life Example

A supercomputer facility is shared by 250 independent subscribers. Each day, each subscriber uses the facility with probability 0.3. The number of tasks sent by each active user has a geometric distribution with parameter 0.15, and each task takes a Erlang(3,10) computer time (in minutes).

This computer doesn’t do any distributed computations, therefore tasks are processed in sequence. What is the probability that the requested computer time in one day is less than 24 hours?

It is impossible to calculate this analytically!!! But we can obtain an estimate by MonteCarlo simulation of the system. Let’s estimate the desired probability with an error margin of +/- 0.01 with probability larger than 0.99.
Shared Supercomputer

- The number of active users is a random variable \( X \sim \text{Bin}(250, 0.3) \).
- Let \( i = 1, 2, \ldots, X \) denote each of the active users. Each of them requests a total of \( T_i \) computer time.
- User \( i \) sets-up \( Y_i \sim \text{Geom}(0.15) \) tasks, each taking \( T_{i,j} \sim \text{Erlang}(3, 10) \) minutes. So
  \[
  T_i = T_{i,1} + \cdots + T_{i,Y_i}.
  \]
- Let \( T = T_1 + \cdots + T_X \) denote the total requested computer time. We want to estimate the probability
  \[
P(T < 24 \text{ hrs}) = P(T < 1440).
  \]

We can therefore generate samples of \( T \) and estimate this probability by an average. How many samples do we need?

In this case \( \epsilon = 0.01 \) and \( \alpha = 0.01 \). Using the normal approximation we get

\[
n \geq \left( \frac{z_{\alpha/2}}{2\epsilon} \right)^2 = \left( \frac{2.575}{0.02} \right)^2 = 16576.6
\]
Shared Supercomputer

```
n=16577;
c=0;
for k=1:n
    T=0;
    X=Binomial(250,0.3);
    for l=1:X;
        Y=Geom(0.15);
        for m=1:Y
            T=T+Erlang(3,10);
        endfor;
    endfor;
    if T<1440
        c=c+1;
    endif;
endfor;
return c/n
```

Running this code once yields the result 0.1719.

Just to check we can run the code more times. Below is a list of the outcomes:

0.1719; 0.1688; 0.1719; 0.1693; 0.1695
0.1751; 0.1746; 0.1710; 0.1765; 0.1759

- In conclusion, we are quite confident that \( P(T < 24 \text{ hrs}) \approx 0.17 \),

so the computer is most of the times overloaded.

**Programming remark:** you should avoid for-loops like the plague!!!
Most high-level languages allow you to be much more efficient by using vectorized constructions for instance...
Estimating Expectations

Other than probabilities we are often interested in estimating expected values and standard deviations. This follows essentially the same principles we have seen...

Let $X$ be a random variable, to estimate $\mathbb{E}[X]$ we can generate $n$ independent and identically distributed samples from the distribution of $X$, denoted by $X_1, \ldots, X_n$. The Law of Large Numbers tells us that

$$\bar{X} = \frac{X_1 + \cdots + X_n}{n},$$

is a good estimator of $\mathbb{E}[X]$. In particular

$$\mathbb{E}[\bar{X}] = \mathbb{E}[X] \quad \text{and} \quad \text{V}(\bar{X}) = \frac{\text{V}(X)}{n}.$$  

How large should $n$ be to ensure the estimate is accurate? The CLT comes again to the rescue, but as we will see things will get more complicated...
Choosing the Sample Size

The CLT tells us that

\[
\frac{\bar{X} - \mathbb{E}[X]}{\sqrt{\text{Var}(X)/n}} \approx \mathcal{N}(0, 1)
\]

This implies that

\[
P(|\bar{X} - \mathbb{E}[X]| \geq \epsilon) = P(\bar{X} - \mathbb{E}[X] > \epsilon \text{ or } \bar{X} - \mathbb{E}[X] < -\epsilon)
\]

\[
= P(\bar{X} - \mathbb{E}[X] > \epsilon) + P(\bar{X} - \mathbb{E}[X] < -\epsilon)
\]

\[
= P\left(\frac{\bar{X} - \mathbb{E}[X]}{\sqrt{\text{Var}(X)/n}} > \frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right)
\]

\[
+ P\left(\frac{\bar{X} - \mathbb{E}[X]}{\sqrt{\text{Var}(X)/n}} < -\frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right)
\]

\[
\approx P\left(Z > \frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right) + P\left(Z < -\frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right)
\]

\[
= 1 - \Phi\left(\frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right) + \Phi\left(-\frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right)
\]

\[
= 2\Phi\left(-\frac{\epsilon}{\sqrt{\text{Var}(X)/n}}\right)
\]
Choosing the Sample Size

To choose \( n \) so that with probability \( 1 - \alpha \) the error of our Monte-Carlo estimate is smaller than \( \epsilon \) we need to solve for \( n \) the inequality

\[
2\Phi\left(-\frac{\epsilon}{\sqrt{V(X)/n}}\right) \leq \alpha.
\]

Recall that we defined \( z_\alpha \) as the unique number such that \( \Phi(z_\alpha) = 1 - \alpha \) (this is called the \( \alpha \)-quantile). Notice also that \( \Phi(-z_\alpha) = \alpha \). Then

\[
n \geq V(X) \left(\frac{z_\alpha/2}{\epsilon}\right)^2.
\]

The problem is that we don’t know \( V(X) \) !!! An (approximate) way to deal with this is to replace it by an estimate of the variance, given by

\[
\hat{V}(X) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2.
\]
Choosing the Sample Size

Since we don’t know $V(X)$, we cannot choose the sample size beforehand, so the problem becomes quite delicate. A way to proceed is to run an experiment for some moderate value of $n$, compute an estimate for the variance, and check if the formula

$$n \geq \hat{V}(X) \left( \frac{z_{\alpha/2}}{\epsilon} \right)^2,$$

where

$$\hat{V}(X) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

is still satisfied. If this is the case we should be okay. Otherwise increase $n$ and try again.

Keep in mind that this approach is approximate, and only makes sense if $n$ is large enough. In most cases provided $n$ is larger than 50 the approximation is quite reasonable...
Shared Computer Again

A supercomputer facility is shared by 250 independent subscribers. Each day, each subscriber uses the facility with probability 0.3. The number of tasks sent by each active user has a geometric distribution with parameter 0.15, and each task takes a \( \text{Gamma}(10,3) \) distributed computer time (in minutes).

This computer doesn’t do any distributed computations, therefore tasks are processed in sequence.

What is the expected computer time requested in one day?

What we want now is simply to estimate \( \mathbb{E}[T] \). This can be done by generating an independent and identically distributed sequence of random variables \( T_i \) and computing the estimate

\[
\bar{T} = \frac{T_1 + \cdots + T_n}{n}.
\]
n=100;
E=0;
for k=1:n
    T=0;
    X=Binomial(250,0.3);
    for l=1:X;
        Y=Geom(0.15);
        for m=1:Y
            T=T+Erlang(3,10);
        endfor;
    endfor;
E=E+T;
endfor;
return E/n

Running this code once yields the result $E[T]=1668$.

Just to check we can run the code more times. Below is a list of the outcomes

1708; 1675; 1691; 1671; 1661;
1624; 1650; 1689; 1663; 1650;

It seems that $E[T]$ is somewhere around 1660 minutes, but we would like to ensure that with high probability we make an error smaller than 5 minutes...

We want to ensure that our MonteCarlo estimate of $E[T]$ is accurate within 5 minutes with probability larger than 0.99.
Shared Supercomputer

After running the code once with $n = 100$ we get

$$\bar{T} = 1695.98 \quad \text{and} \quad \hat{V}(T) = 64987.75 .$$

In order to guarantee the desired accuracy we need the sample size to satisfy the formula below, with $\epsilon = 5$ and $\alpha = 0.01$.

$$n \geq \hat{V}(X) \left( \frac{z_{\alpha/2}}{\epsilon} \right)^2 = 64987.75 \left( \frac{2.575}{5} \right)^2 = 17237$$

Clearly we should use a larger sample size. Let’s try $n = 20000$.

We get

$$\bar{T} = 1666.65 \quad \text{and} \quad \hat{V}(T) = 57359.32 ,$$

and we are quite confident that the value of $\mathbb{E}[T]$ we estimate has an error that is smaller than 5 minutes.