ST3247 Simulation

Lingjie, May 8, 2021

R programming

Common functions

remainder	: %%
matrix multiplication	: %*%
rounding	: floor, ceiling, round, signif
load R commands	: source(filename, echo=TRUE)
set seed	: set.seed(1234)
vectorise function	: Vectorize(fn)
apply	: sapply(X, fn, *params)
	: apply(X, 1, fn)
	: 1 := row, 2 := col
generate sample	: rxxxx
pdf $P(X = x), f_X$	: dxxxx
cdf $P(X \leq x), F_X$	: pxxxx
cdf quantile $F^{-1}(x)$	: qxxxx

Probability and Math Stat Background

Important knowledge

Law of Total Probability
$P(X \in A)$
$= \sum_{i=1}^{n} P(X \in A, Y \in B_i)$
$= \sum_{i=1}^{n} P(X \in A|Y \in B_i)P(Y \in B_i)$
Indicator function: $I(a < x < b)$
Mode of distribution
Discrete: i s.t. $p_i \geq p_j \ \forall j \neq i$
Continuous: x s.t. $f(x) \geq f(w) \ \forall x \neq w$
Conditional Expectation:
$E(X|Y = y) = \int x f_{x|y}(x|y)dx = \frac{\int x f_{x,y}(x,y)dx}{\int f_Y(y)dx}$
and $E(X) = E(E(X|Y))$
Conditional Variance:
$Var(X) = E(Var(X|Y)) + Var(E(X|Y))$
$Var(X) \geq Var(E(X|Y))$

Computing RV pmf Recursively

Binomial

$$\begin{aligned} P(X = x+1) &= \frac{n!}{(n-x-1)!(x+1)!}p^{x+1}(1-p)^{n-x-1} \\ &= \frac{n!(n-x)}{(n-x)!x!(x+1)}p^x(1-p)^{n-x}\frac{p}{1-p} \\ &= \frac{n-x}{x+1}\frac{p}{1-p}P(X = x) \end{aligned}$$

Poisson

For $x \geq 0$

$$\begin{aligned} P(X = x+1) &= e^{-\lambda}\frac{\lambda^{x+1}}{(x+1)!} \\ &= \frac{\lambda}{x+1}e^{-\lambda}\frac{\lambda^x}{x!} \\ &= \frac{\lambda}{x+1}P(X = x) \end{aligned}$$

Solving Min Max RV

Let $Y \sim min(X_1, X_2)$ X_1, X_2 are independent
$P(Y \leq y) = P(min(X_1, X_2) \leq y) = 1 - P(min(X_1, X_2) > y)$
$= 1 - P(X_1 > y, X_2 > y) = 1 - P(X_1 > y)P(X_2 > y)$
Let $Y \sim max(X_1, X_2)$ X_1, X_2 are independent
$P(Y \leq y) = P(max(X_1, X_2) \leq y)$
$= P(X_1 \leq y, X_2 \leq y) = P(X_1 \leq y)P(X_2 \leq y)$

Uniform Distribution

pdf	: $f(x) = \frac{1}{b-a}I(a \leq x \leq b)$
cdf	: $F(x) = \frac{x-a}{b-a}$
$E(X)$	: $\frac{b+a}{2}$
$Var(X)$	: $\frac{(b-a)^2}{12}$

Poisson Process

$N(t)$:= number of events in the time interval $[0, t]$
$N(t + s)$:= number of events in the time interval $[0, t + s]$
$N(t+s) - N(t)$:= num of events in the time interval $[t, t+s]$
$m(t)$:= the area under the intensity function from 0 to t
$m(t + s) - m(t)$:= the area under the intensity function from t to $t + s$
process is defined as Poisson process with rate $\lambda, \lambda > 0$ if

1. $N(0) = 0$
   $\Rightarrow$ nothing happened before time 0
2. Number of events occurring in disjoint time intervals are independent
   $\Rightarrow$ independent increments assumption
   $\Rightarrow$ e.g. $N(t)$ is independent of $N(t + s) - N(t)$
3. Distribution of number of events only depend on length of interval, not location
   $\Rightarrow$ stationary increment assumption
   $\Rightarrow$ distribution of event across every interval of time is the same
4. $\lim_{h\to 0} \frac{P(N(h)=1}{h} = \lambda$
   $\Rightarrow$ within a small interval of length h, the probability of one event is approximately λh
5. $\lim_{h\to 0} \frac{P(N(h)\geq 2)}{h} = 0$
   $\Rightarrow$ within a small interval of length h, the probability of two or more events is approximately 0

Therefore, $N(t) \sim Pois(\lambda t)$

Non-homogeneous Poisson Process

$N(t)$:= number of events by time t
$N(t)$ is a non-homogenous Poisson process with intensity function $\lambda(t)$ if
(same condition as Poisson Process except)

1. $\lim_{h\to p} P($exactly 1 event between t and $t + h)/h = \lambda(t)$

$N(t + s) - N(t) = Pois(m(t + s) - m(t))$

Mean value function

Definition 1 (Mean Value Function)

$$m(t) = \int_0^t \lambda(s)ds, t \geq 0$$

$\lambda(t)$:= intensity at time t, indicates how likely it is that event will occur around time t.
$N(t + s) - N(t) = Pois\,(m(t + s) - m(t))$

Pseudo-Random Number Generation

Types of PRNGs

Linear Congruential Generators

Pseudo-code for LCG

1. Set Z_0
2. For $i = 1$ to n:
   1. Set $Z_i = (aZ_{i-1} + c) \bmod m$
   2. Set $U_i = Z_i/m$

Obtaining Full Period

1. Only positive integer that divides both m and c is 1
2. If q is prime num that divides m, then q divides $a - 1$
3. If 4 divides m, then 4 divides $a - 1$

PRNG in R

shift register with period $2^{19937} - 1$

Statistical Tests for PRNGs

Frequency Test: Check for Uniformity

check: if U_i appear to be evenly distributed between 0 and 1

Correlation Test: Check for Autocorrelation

check: no correlation at lag j for all j

Discrete Random Variable Generation

Objective: Given pmf, generate random variable
Constraint:
1. $E(N)$, 2. num of $Unif[0,1]$, 3. storage space

Algo	No. iter	No. RVs	Storage	Infinite support?
Seq. Inversion	$E(X)+1$	1	-	Y
Ordered. Inversion	$< E(X)+1$	1	Variable	Y
Truncation	1	1	-	Y
Rejection	$\approx c$	$\approx 2c$	-	Y
Table	1	1	M	N
Alias	2	2	3(K-1)	N

Inversion Method

Sequential Inversion

Algorithm 1 [Sequential Inversion]

1. Generate $U \sim Unif[0,1]$
2. Set $X = 0, S = p_0 \Rightarrow P(X=0) = p_0$
3. While $U > S$, do
   3.1 $X = X + 1$
   3.2 $S = S + p_x$
4. Return X

R Implementation [Poisson]

```
U <- runif(n=1, min=0, max=1) #cdf
X <- 0; S <- exp(-lambda)) #P(X=0)
while(U > S){
  X <- X + 1
  S <- S + exp(-lambda)*(lambda^X)/
      factorial(X)
}
print(X)
```

Expected Iterations

$$\begin{aligned}E(N) =&1P(X=0)+2P(X=1)+3P(X=2)+\cdots\\ =&0P(X=0)+1P(X=0)+2P(X=1)+3P(X=2)\cdots\\ &+P(X=0)+P(X=1)+P(X=2)+P(X=3)+\cdots\\ =&E(X)+1\end{aligned}$$

Ordered Inversion

Improved Inversion method by checking the largest interval first

Algorithm 2 [Ordered Inversion]

1. Set-up Stage:
   1.1 Sort p_i's (decreasing)
   1.2 $Y :=$ indices of the sorted p_i's
   1.3 $q_i :=$ to be pmf of sorted p_i's
2. Generation Stage:
   2.1 Generate Z from q_i (algo 1)
   2.2 Return X = Y[Z+1]

R Implementation [discrete pmf]
$X \in (p_0 = 0.2,\ p_1 = 0.55,\ p_2 = 0.25)$

```
X <- c(0.2, 0.55, 0.25)
###############
# Set-up Stage
###############
Q <- sort(X, decreasing=TRUE)
#Q = (0.55, 0.25, 0.2)
Y <- c(1, 2, 0) # stored index
#Y adjust with support (start = 1 or 0)
##################
# Generation Stage
##################
U <- runif(n=1, min=0, max=1) #cdf
Z <- 0; S <- Q[1] #P(Q=0)
while(U>S){
  Z <- Z+1
  S <- S+Q[Z+1]
}
Y[Z+1] #generated X
```

Expected Iterations

$$E(N1) = E(X)+1 = 2.05$$
$$E(N2) = [0(0.55)+1(0.25)+2(0.2)]+1 = 1.65$$

Truncation of a Related Continuous cdf

Required steps

- Identify $G(x)$ by setting $G(i+1) = F(i)$
- Show $\frac{dG}{dx} > 0$ (mono increasing) and $0 \leq G \leq 1$
- G(0) = 0
  Note: if support starts from 1, check $G(1) = 0$
- find $G^{-1}(U)$

Algorithm 3 [Inversion by Truncation]

1. Generate $U \sim Unif[0,1]$
2. Set $X = \lfloor G^{-1}(U) \rfloor$
3. Return X

R Implementation [discrete uniform]
generate $X \sim uniform(0, K-1)$
$F(i) = \sum_{j=0}^{i} \frac{1}{K} = \frac{i+1}{K}$
$G(i+1) = \frac{i+1}{K} \Rightarrow G(x) = \frac{x}{K}$
$\because G^{-1}(U) = KU \therefore X = \lfloor KU \rfloor$

```
U <- runif(n=1) # continuous uniform
X <- floor(K*U)
```

Note: if $Y \sim uniform(1, K) \Rightarrow Y = X + 1$

R Implementation [geometric]
$X \sim Geo(p), q := 1-p$
$P(X=i) = p(1-p)^{i-1} = (1-q)q^{i-1} = q^{i-1} - q^i, i \geq 1$
$F(i) = \sum_{j=1}^{i} P(X=j) = 1 - q^i$ (oscillating sum)
$G(i+1) = F(i) = 1 - q^i \Rightarrow G(x) = 1 - q^{x-1}$
$\because G^{-1}(U) = 1 + \frac{\log(1-U)}{\log q} \therefore X = \lfloor 1 + \frac{\log(1-U)}{\log q} \rfloor$

```
U <- runif(1)
X <- floor(1+log(1-U)/log(q))
```

R Implementation [Random Permutation]

Algorithm 4 [Generating Random Permutation]
Let $P_1 P_2 \cdots P_n$ be any permutation of the num $\{1, 2, \cdots, n\}$

1. Set k=n
2. Generate $Z \sim Unif(1, 2, \cdots, k)$
3. Interchange P_z and P_k
4. Update $k = k - 1$
5. if $k > 1$, return to step 2
6. Return the final permutation $P_1 P_2 \cdots P_n$

```
P <- 1:n # ordered numbers
k <- n
while(k>1){
U <- runif(n=1)
z <- floor(k*U) + 1 #discrete unif(1, k)
Pz <- P[z]; Pk <- P[k]
P[k] <- Pz; P[z] <- Pk # interchange value
```

```r
k <- k - 1
}
return(P)
```

Table Method

Algorithm 5 [Table Method]
$X \in S = \{0, 1, 2, \cdots, K-1\}$
Set-up Stage:
 Create table A length M where each $i \in S$ appear k_i times
Generate Stage:

1. Generate U from $Unif[0, 1]$
2. Let $Z = \lfloor MU \rfloor + 1$
3. Return X = A[Z]

R Implementation [4-point distribution]
$p_0 = 0.120,\ p_1 = 0.111,\ p_2 = 0.419,\ p_3 = 0.350$

```r
A <- c(rep(0, 120), rep(1, 111),
       rep(2, 419), rep(3, 350))
U <- runif(n-1)
Z <- floor(M * U) + 1
X <- A[Z]
```

Rejection Method

Wish to draw $X \sim p_i$ with only access to $Y \sim q_i$
with $p_i \le cq_i, c \ge 1\ \forall i$
$p_i :=$ target dist, $q_i :=$proposal dist, $c :=$rejection constant
P(accepted) = 1/c = 1 / num of iteration
Algorithm 6 [Rejection Algorithm for Discrete RV]

1. generate $U \sim Unif[0, 1]$
2. generate Y from q_i
3. if $Ucq_Y \le p_Y$
   Set $X = Y$
4. Else
   Go to Step 1
5. Return X

R Implementation [infinite distribution]
$p_i = \frac{6}{\pi^2 i^2}, i \ge 1$, consider $q_i = \frac{1}{i(i+1)}, i \ge 1$
generate q_i by inversion by truncation:
$\because G(x) = 1 - 1/x \therefore G^{-1}(U) = 1/(1-U)$
find $c:\ p_i = \frac{6}{\pi^2 i^2} \le \frac{12}{\pi^2 i(i+1)} = \frac{12}{\pi^2} q_i = cq_i$
$U \le (p_i)/(cq_i) = \frac{6}{\pi^2} \frac{\pi^2}{12} \frac{(i^2+i)}{i^2} = \frac{1}{2}(1 + \frac{1}{i})$

```r
while(TRUE){
    U <- runif(n=1)
    V <- runif(n=1)
    Y <- floor(1/(1-V)) #qi
    if(U <= 0.5*(1+1/Y)){
        X <- Y
        break
    }
}
```

Alias Method

based on theorem: any finite pmf can be re-written as uniform mixture of descrete 2-point distribution.

Mixture distribution

Suppose that $q_j^{(1)}, q_j^{(2)}, \cdots, q_j^{(n)}$ are n different pmfs on the finite set $\{0, 1, \cdots, K-1\}$
Supposed p_i is a pmf on the set $\{1, \cdots, n\}$
Then the following pmf is known as a mixture of n discrete distributions:
$P(X = j) = \sum_{i=1}^{n} p_j q_j^{(i)},\ j \in [0, K-1]$
Algorithm 8 (Generating from Mixtures)

1. Generate V from p_i
2. Generate X from $q_j^{(V)}$
3. Return$X$

R Implementation [Generating from a mixture]
X with pmf $r_0 = 0.1, r_1 = 0.1, r_2 = 0.4, r_3 = 0.4$

```r
U <- runif(n=1)
V <- runif(n=1)
if(U <= 0.2){
    if(V <= 0.5){X <- 0}
    else{X <- 1}
}
else{
    if(V <= 0.5){X <- 2}
    else{X <- 3}
}
```

Lemma 1 (Alias Method Lemma)

For a probability vector $\mathbf{P} = (p_0, \cdots, p_{K-1})$,
1. there exists an $i \in [0, K-1]$ $s.t.$ $\mathbf{P_i} < 1/(K-1)$, and
2. for this i, there exists a $j, j \ne i$ $s.t.$ $\mathbf{P_i} + \mathbf{P_j} \ge 1/(K-1)$

Theorem 2 (Alias Method Theorem)

Any finite probability vector $\mathbf{P}$ can be expressed as
$\mathbf{P} = \frac{1}{K-1} \sum_{m=1}^{K-1} \mathbf{Q^{(m)}}$
for suitably defined $\mathbf{Q^{(1)}}, \cdots, \mathbf{Q^{(K-1)}}$
Algorithm 10 (Alias Method Setup)

1. Initialise $\mathbf{P'} = \mathbf{P}, n = K$
2. For m in 1 : $(K-2)$:
   1. Apply $Lemma 1$ to pick i, j from $\mathbf{P'}$
      Note: $\mathbf{P'}$ is n-point dist, $i, j \in [0, K-1]$
   2. Set $\mathbf{Q_i^{(m)}} = (n-1)\mathbf{P'_i}$
   3. Set $\mathbf{Q_j^{(m)}} = 1 - (n-1)\mathbf{P'_i}$
   4. Update
      $\mathbf{P'} = \left[\mathbf{P'} - \frac{\mathbf{1}}{\mathbf{n-1}} \mathbf{Q^{(m)}}\right] \frac{n-1}{n-2}$
   5. Update $n = n - 1$
3. Set $\mathbf{Q^{(K-1)}} = \mathbf{P'}$

R Implementation [Alias Method Setup]
$p_0 = 7/16, p_1 = 4/16, p_2 = 2/16, p_3 = 3/16$

```r
P <- c(7/16, 4/16, 2/16, 3/16)
n <- K <- length(P)
# set up table
Q.table <- matrix(0, nrow=K-1, ncol=3)
for(m in 1:K-2){
    # Pi < 1/(K-1)
    id <- which(P>0 & P < 1/(n-1))

    # choose min pi satisifying cond
    i <- id[which.min(P[id])] - 1
    # since lemma always true, choose max val
    j <- which.max(P) - 1

    # set Q^m row
    Q <- rep(0, K)
    Q.i <- P[i+1]*(n-1)
    Q[i+1] <- Q.i
    Q[j+1] <- 1 - Q.i

    # store results
    Q.table[m,] <- c(i, j, Q.i)

    # update
    P[i+1] <- 0 # pi = 0
```

```
    P[-(i+1)] <- ((P-(1/n-1)*Q)*(n-1)/(n-1))

    # change index
    n <- n - 1
}
    # Q^{K-1}
    id <- which(P>0 & P< 1/(n-1))
    i <- id[which.min(P[id])]-1
    j <- which.max(P) - 1
    Q.table[K-1,] <- c(i, j, P[i+1])

    # store names
    colnames(Q.table) <- c('i','j',
                           'Q.stage.i')
```

Algorithm 11 (Alias Method Generation)

1. Generate $V \sim Unif(1, K-1)$
2. Generate X from $\mathbf{Q}^{(\mathbf{V})}$
3. Return X

R Implementation [Alias Method Generation]

with a given alias table

i	j	Q
0	3	1/12
1	2	5/7

```
n <- nrow(alias.table) + 1
U <- runif(n=1) # for sample
V <- runif(n=1) # for rejection
row.num <- floor((n-1)*U) + 1 # sample
X <- ifelse(V <= alias.table[row.num, 3],
            alias.table[row.num, 'i'],
            alias.table[row.num, 'j'])
```

Continuous Random Variable Generation

Inverse Transform Algorithm

Theorem1: Distribution of $F^{-1}(U)$
Let $U \sim unif(0,1)$, for any strictly increasing cdf F,
$X = F^{-1}(U)$ has distribution F
Algorithm 1 (Inverse Transform Algo)

1. Generate $U \sim unif(0,1)$
2. Return $X = F^{-1}(U)$

R Implementation[Exponentials]
cdf for *exp* is $F(x) = 1 - e^{-\lambda x}$
$\Rightarrow F^{-1}(U) = -\frac{1}{\lambda}log(1-U) \sim Exp(\lambda) \Leftrightarrow -\frac{1}{\lambda}log(U)$

```
U <- runif(1)
X <- -(1/lambda)*log(U)
```

R Implementation[X^n]
$F(x) = x^n \Rightarrow F^{-1}(U) = U^{1/n}$

```
U <- runif(1)
X <- U**(1/n)
```

R Implementation[Maximal Order Statistics]
$X_{(n)} = \max\{X_1, X_2, \cdots, X_n\}$
Method1: take Max of X

```
U <- runif(n)
X <- inverseF(U)
Xn <- max(X)
```

Method2: use $P(X_{(n)} \leq x) = [F(x)]^n$

```
Xn <- inverseF(U**(1/n))
```

Method3: take Max of U

```
U <- runif(n)
Un <- max(U)
X <- inverseF(Un)
```

Inversion by Numerical Solution

In cases where F^{-1} is unknown, we solve $F(X) - U = 0$
Since goal: Given U, find X s.t. $F(X) = U$

Bisection Method

provide an initial interval [a,b], search for solution
Alogrithm 2 (Bisection method)
Given an initial interval [a,b] that contains the solution to $F(X) = U$ and a width δ
Note: require understanding if the solution indeed lies in interval [a,b]

1. While $b - a > 2\delta$

    Set $X = \frac{1}{2}(a+b)$

    If $F(X) \leq U$ set $a = X$

    Else set b = X

2. Return X

R Implementation[Generate Normal(0,1)]
use *pnorm* for cdf, note: mode of N(0,1) = 0

1. Generate $U \sim unif(0,1)$
2. If $U \leq 0.5$

    Solve $\Phi(X) - U = 0$ using bisection, with $a = -7, b = 0$

3. Else

    Solve $\Phi(X) - U = 0$ using bisection, with $a = 0, b = 7$

```
f <- function(X, U) pnorm(X) - U #F(X)-U
invert_f <- function(U) {
    if(U<=0.5){
        out <- uniroot(f, c(-7, 0), U=U)
    } else {
        out <- uniroot(f, c(0, 7), U=U)
    }
    out$root
}
```

Rejection Method

wish to draw X with $f(x)$ with access to $g(y)$
Support of X contained in support of Y
$f(x) \leq cg(x) \quad \forall\ x, c \geq 1$
Note: can find $\max \log(h(x))$ for easy compute
Theorem 2: (Rejection Algo for Continuous RV)

1. The RV generated by rejection algo has a density f
2. The num of iterations of the algo needed is a geometric RV with mean c

Algorithm 3 [Rejection Algo for Continuous RV]

1. Generate $U \sim unif(0,1)$
2. Generate Y from $g(y)$
3. If $Ucg(Y) \leq f(Y)$

    Set $X = Y$

4. Else

    Go to Step 1

Beta(2,4)

$$
\begin{aligned}
f(x) &= \frac{\Gamma(2+4)}{\Gamma(2)\Gamma(4)} x(1-x)^3 \\
&= 20x(1-x)^3, 0<x<1 \\
g(x) &= I(0 \le x \le 1) \\
c &\ge \max_x \frac{f(x)}{g(x)} = h(x) \\
h'(x) &= 20[(1-x)^3 - 3x(1-x)^2] = 0 \Rightarrow x = 0.25 \\
c &\ge \max_x h(x) = h(0.25) = 135/64 \approx 2.11
\end{aligned}
$$

1. Generate $U, V \sim unif(0,1)$
2. if $U \le (256/27)V(1-V)^3$ set $X = V$
3. Else, return to step 1

Gamma(1.5, 1)

$$
\begin{aligned}
f(x) &= \frac{1}{\Gamma(1.5)} x^{1/2} e^{-x} \\
&= \frac{2}{\sqrt{\pi}} x^{1/2} e^{-x}, x > 0 \\
g(x) &= \frac{2}{3} e^{-2x/3} \sim Exp(E(f) = 2/3) \\
c &\ge h(x) = \frac{3}{\sqrt{\pi}} x^{1/2} x^{-x/3} \\
h'(x) &= \frac{3}{\sqrt{\pi}} \left(\frac{1}{2} x^{-1/2} e^{-x/3} - \frac{1}{3} x^{1/2} e^{-x/3} \right) \\
&= 0 \Rightarrow x = 1.5 \\
c &\ge h(1.5) = \frac{3^{3/2}}{(2\pi e)^{1/2}} \approx 1.257
\end{aligned}
$$

1. Generate $U, V \sim unif(0,1)$
2. Set $Y = -(3/2)log(V)$
3. if $U < (2eY/3)^{1/2} e^{-Y/3}$ set $X = Y$
4. Else return to step 1

N(0,1)

1. Generate W from

$$f(w) = \frac{2}{\sqrt{2\pi}} e^{-w^2/2}, 0 < w < \infty$$

using rejection algorithm

2. Generate $U \sim unif(0,1)$
3. If $U \le 0.5$

   Return $X = -W$

4. Else

   Return $X = W$

Rejection algorithm to generate $f(w)$

$$
\begin{aligned}
g(y) &= e^{-y}, 0 < y < \infty \\
h(x) &= \sqrt{2/\pi} e^{x - x^2/2} \\
c &= \max_x h(x) = h(1) = \sqrt{2e/\pi} \approx 1.32
\end{aligned}
$$

1. Generate $U_1, U_2, U_3 \sim unif(0,1)$
2. Set $Y = -log(U_1)$
3. If $U_2 \le exp(-(Y-1)^2/2)$ go to step 5
4. Else return to step 1
5. If $U_3 \le 0.5$, set $X = -Y$, else set $X = Y$
6. Return X

Modified Rejection Method

Incomplete Knowledge of f

Given only kernel of $f \Rightarrow f_1$ with a normalising constant c_N
kernel does not contain any multiplicative constants

$$
\begin{aligned}
\int f_1(x)dx &= \frac{1}{C_N} \\
f(x) &= c_N f_1(x) \\
\text{find c, g s.t.} f_1(x) &\le cg(x)
\end{aligned}
$$

$$
\begin{aligned}
&\text{Gamma}: f_1(x) = x^{a-1} e^{-x/b}, x > 0 \\
&\text{Normal}: f_1(x) = e^{-0.5(x-5)^2}, x \in \mathbf{R} \\
&\text{Beta}: f_1(x) = x^{15}(1-x)^{32}, x \in [0,1]
\end{aligned}
$$

No restriction on $c > 1$, but $\min c$

Modified Rejection Alogrithm

Alogrithm 4 [Modified Rejection Algorithm]

1. Generate $U \sim unif(0,1)$
2. Generate $Y \sim g(y)$
3. If $Ucg(Y) \le f_1(Y)$

   Set $X = Y$

4. Else

   Go to Sep 1

5. Return X

Truncated Gamma

$$
\begin{aligned}
Y &\sim \Gamma(2,1) \\
X &= Y | Y \ge 5 \\
P(X \le x) &= P(Y \le x | Y \ge 5) \\
&= \frac{P(5 \le Y \le x)}{P(Y \ge 5)} \\
&= \frac{\int_5^x y e^{-y} dy}{P(Y \ge 5)} \\
&\propto \int_5^x y e^{-y} dy \\
\Rightarrow f(x) &\propto f_1(x) = x e^{-x} \\
\text{let } g(x) &= \frac{\frac{1}{2} e^{-x/2}}{e^{-5/2}}, x \ge 5 \\
&\Leftrightarrow P(G \le x | G \ge 5), G \sim Exp(1/2) \\
h(x) &= \frac{f_1(x)}{g(x)} = \frac{2xe^{-x}}{e^{5/2 - x/2}} \\
&= 2xe^{-5/2} e^{-x/2}, x \ge 5 \\
\because h'(x) &\le 0 \Rightarrow \max_h h(x) = h(5) = 10e^{-5} \\
\Rightarrow f_1(x) &\le 10e^{-5} g(x)
\end{aligned}
$$

From memoryless property

$$
\begin{aligned}
P(W > 5 + t | W > 5) &= P(W > t) \\
P(W \le 5 + t | W > 5) &= P(W \le t) \\
P(W \le x | W > 5) &= P(W \le x - 5), x > 5 \\
&= P(W + 5 \le x), x > 5
\end{aligned}
$$

$\Rightarrow$ generate $Exp(0.5) + 5$ for $W | W > 5$

1. Generate $U_1, U_2 \sim unif(0,1)$
2. Set $Y = 5 - 2log(U_1)$
3. If $U_2 \le (Y/5) e^{-Y/2} e^{5/2}$ then set $X = Y$
4. Else, return to step 1

Composition or Mixture Method

express target density f as finite mixture of densities

$$f(x) = \sum_{i=1}^{n} p_i f_i(x)$$

Algorithm 5 [Mixture Method for Continuous R.V.s]

1. Generate $Z \sim p_i$
2. Generate $X \sim f_Z$

A mixture of 2 Densities
Example:

$$f(x) = \frac{1}{2\sqrt{2\pi}} \left[e^{x} + e^{-x}\right] e^{-0.5(x^2+1)}$$
$$= \frac{1}{2} f_1(x) + \frac{1}{2} f_2(x)$$

where $f_1(x) = N(1,1), f_2(x) = N(-1,1)$

1. $U \sim unif(0,1)$
2. If $U \leq 0.5$
   Return $X \sim N(1,1)$
3. Else
   Return $X \sim N(-1,1)$

A Mixture of Point Mass and Uniform
Example:

$$f(x) = \frac{1}{3} I(x=0) + \frac{2}{3} I(2 < x < 3)$$

$I(X=0)$ denotes a degenerate RV always taking value 0, other is $unif(2,3)$

1. Generate $U \sim unif(0,1)$
2. If $U \leq 1/3$
   Return X = 0
3. Else
   Return $X \sim unif(2,3)$

Composition for Polynomial Densities

$$\text{consider } f(X) = \sum_{i=0}^{K-1} c_i x^i, x \in [0,1]$$
$$\text{observe } \Leftrightarrow \sum_{i=0}^{K-1} \frac{c_i}{i+1}(i+1)x^i$$

Condition

- c_i are non-negative constants
- $K > 0$
- $\sum_{i=0}^{K-1} c_i/(i+1) = 1$

Algorithm 6 [Composition Method for Polynomial Densities]

1. Generate $Z \sim p_i = \frac{c_1}{(i+1)}, Z \in [0, K-1]$
2. Generate $U \sim Unif(0,1)$
3. Return $X = U^{1/(Z+1)}$

Generating from a Polynomial Density
Example:

$$f(x) = x^3 + \frac{9}{2}x^5$$
$$= \frac{1}{4}4x^3 + \frac{3}{4}6x^5$$

1. Generate $V, U \sim unif(0,1)$
2. If $V \leq 1/4$, set $X = U^{1/4}$
3. Else, set $X = U^{1/6}$

Box-Muller Transformation

Let $X, Y \sim N(0,1)$ independently, R, Θ denote polar coordinates of (X,Y)

$$R^2 = X^2 + Y^2$$
$$tan\Theta = \frac{Y}{X}$$
$$f(x,y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2}$$
$$\text{let } d = x^2 + y^2 \geq 0, \;\; \theta = \tan^{-1}(\frac{y}{x}) \in [0, 2\pi]$$
$$f(d,\theta) = \left(\frac{1}{2}e^{-d/2}\right)\left(\frac{1}{2}\pi\right)$$

Algorithm 7 [Box-Muller Transformation]

1. Generate $U_1, U_2 \sim Unif(0,1)$
2. Set $R^2 = -2log(U_1), \Theta = 2\pi U_2$
3. Set
   $$X = Rcos(\Theta) = \sqrt{-2log(U_1)}cos(2\pi U_2)$$
   $$Y = Rsin(\Theta) = \sqrt{-2log(U_1)}sin(2\pi U_2)$$
4. To obtain $N(\mu, \sigma^2)$, use linear transformation of X, Y

Poisson Process

Theorem 3 [Distribution of Inter-Arrival Times]

- The inter-arrival times $X_1, X_2, \cdots$ are i.i.d. $Exp(\lambda)$
- The time of the nth arrival, $S_n = \sum_{i=1}^{n} X_i$ is distributed as $\Gamma(n, \lambda)$

Theorem 4 [Conditional Distribution of $\{S_n\}$]
Given $N(t) = n$, the arrival times $S_1, S_2, \cdots, S_n$ have the same distribution as the order statistics corresponding to n independent RV uniformly distribution on the interval $(0,t)$

Homogeneous Poisson Process

Algorithm 8 [Method1: Homogeneous Poisson Process]

1. Set $t = 0, I = 0$
2. Repeat:
   Generate $U \sim unif(0,1)$
   Set $t = t - \frac{1}{\lambda}log(U)$
   If $t > T$ exit the algo (no events by time T)
   Else
   Set $I = I + 1$
   Set $S_i = t$

S_n will contain the n event times before T, in increasing order
Remark: Algo 8 offers alternative way to generate Pois. Since $N(1) \sim Pois(\lambda)$, we set $T = 1$ and count number of events.
Algorithm 9 [Method2: Homogeneous Poisson Process]

1. Generate $N(T) \sim Pois(\lambda T)$
2. Generate $U_1, \cdots, U_{N(T)} \sim unif(0,1)$
3. Set the collection of event times to be $\{TU_1, TU_2, \cdots, TU_{N(T)}\}$

Nonhomogeneous Poisson Process

Algorithm 10 [Thinning Method for Nonhomogeneous Poisson Process]

1. Set $t = 0, I = 0$
2. While $t \leq T$

   Generate $U, V \sim unif(0,1)$

   Set $t = t - \frac{1}{\lambda} log(U)$. If $t > T$ exit algo

   If $V \leq \lambda(t)/\lambda$

   Set $I = I + 1$

   Set $S_i = t$

Note: $\lambda \geq \lambda(t) \; \forall \; t$

Estimation in Monte Carlo Simulations

1. Identify X
2. Generate $X_1, X_2, \cdots, X_n$
3. Estimate $E(X)$ using $\bar{X}$

Properties of Sample Mean and Sample Variance

Strong Law of Large Numbers (SLLN)

Theorem 1 (SSLN)
Given iid $X_1, X_2, \cdots, X_n$, with $E(X) < \infty$

$$\frac{1}{n}\sum_{i=1}^{n} X_i \to E(X) \text{ with prob 1}$$

Central Limit Theorem (CLT)

Given iid $X_1, X_2, \cdots, X_n$, with $-\infty < E(X_1) = \mu < \infty$ and $Var(X_1) = \sigma^2 < \infty$

$$\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \to N(0,1) \text{ converge in prob}$$

Stoppping Rules

Confidence Interval

Algorithm 3 & 4 (Confidence Interval Stopping Rule)

1. $L :=$ max acceptable interval width
2. Generate $X_1, X_2, \cdots, X_n$ s.t.

$$2z_{(\alpha/2)}\frac{s}{\sqrt{n}} < L$$
$$2z_{(\alpha/2)}\frac{\sqrt{\bar{X}(1-\bar{X})}}{\sqrt{n}} < L \text{ for Ber(p)}$$

3. The CI for $E(X)$

$$\bar{X} \pm z_{(\alpha/2)}\frac{s}{\sqrt{n}}$$
$$\bar{X} \pm z_{(\alpha/2)}\frac{\sqrt{\bar{X}(1-\bar{X})}}{\sqrt{n}} \text{ for Ber(p)}$$

Standard Error

Algorithm 5 & 6 (Standard Error Stopping Rule)

1. $d :=$ max acceptable standard error deviation
2. Generate $X_1, X_2, \cdots, X_n$ s.t.

$$\frac{s}{\sqrt{n}} < d$$
$$\frac{\sqrt{\bar{X}(1-\bar{X})}}{\sqrt{n}} < d \text{ for Ber(p)}$$

3. return $\bar{X}$ as estimate of $E(X)$

Recursive update of $\bar{X}, s^2$

$$\bar{X}_{j+1} = \bar{X}_j + \frac{X_{j+1} - \bar{X}_j}{j+1}$$
$$s^2_{j+1} = \left(1 - \frac{1}{j}\right) s^2_j + (j+1)(\bar{X}_{j+1} - \bar{X}_j)^2$$

```
meanUpdate <- function(X.new, X.bar, j){
    X.bar + (X.new - X.bar)/(j+1)
    }
varUpdate <- function(s2, X.bar,
                              X.bar.new, j){
    (1-1/j)*s2+(j+1)*
                    (X.bar.new - X.bar)**2
    }
```

Monte Carlo Integration

Direct Monte Carlo Integration

$$\int_{-\infty}^{\infty} h(x)f(x)dx \Leftrightarrow E(h(X)) \approx \frac{1}{n}\sum_{i=1}^{n} h(x_i)$$

Important: 1. introduce a pdf to integral, 2. Simulate the introduced pdf
Note: similiar result applies for summation

Example: integrating over interval a, b

$$\theta = \int_a^b g(x)dx = \int_a^b g(x)(b-a)\frac{1}{b-a}I(a \leq x \leq b)dx$$

1. generate $U_1, U_2, \cdots, U_n \sim Unif(0,1)$
2. set $V_i = (b-a)U_i + a \; \forall i$
3. Estimate θ with $\frac{1}{n}\sum_{i=1}^{n} g(V_i)(b-a)$

Example: integrating over interval $-\infty, \infty$

$$\theta = \int_{-\infty}^{\infty} xe^{(-x^2)}dx = \int_{-\infty}^{\infty} \sqrt{2\pi}xe^{(-0.5x^2)}\frac{1}{\sqrt{2\pi}}e^{(-0.5x^2)}dx$$

1. generate $X_1, X_2, \cdots, X_n \sim N(0,1)$
2. estimate θ with $\frac{1}{n}\sum_{i=1}^{n} \sqrt{2\pi}X_i e^{-0.5X_i^2}$

Importance Sampling

$$\theta = E[h(X)] = \int h(x)f(x)dx = \int \frac{h(x)f(x)}{g(x)}g(x)dx$$
$$= \int h(x)\frac{f(x)}{f_t(x)}f_t(x)dx$$

Algorithm 7 (Importance Sampling)

1. Generate $X_1, X_2, \cdots, X_n \sim g$
2. Estimate θ using

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n}\frac{h(X_i)f(X_i)}{g(X_i)}$$

Tilted Density

Suppose f is pdf, $M(t) = \int exp(tx)f(x)dx$ be mgf, the tilted density of $f :=$

$$f_x(x) = \frac{exp(tx)f(x)}{M(t)} I(-\infty < t < \infty)$$

useful for estimating small probability. However, this is a family of distribution and choosing t is tricky

Example of tilted densities

pdf	tilted densities
$Exp(\lambda)$	$Exp(\lambda - t), t < \lambda$
$Ber(p)$	$Ber(\frac{pe^t}{pe^t+1-p}), M(t) = pe^t + (1-p)$

Binomial Tail

want to estimate $P(X \geq 16), X \sim Bin(20, 0.4)$
pmf: $P(X = x) = C_x^{20} p^x (1-p)^{20-x}, p = 0.4$
tilted pmf: $P(X_t = x) = \frac{exp(tx)}{(pe^t+1-p)^{20}} C_x^{20} p^x (1-p)^{20-x}$
$P(X \geq 16) = \sum_{x=0}^{20} I(X \geq 16) \left[C_x^{20} p^x (1-p)^{20-x}\right]$
$= \sum_{x=0}^{20} I(x \geq 16) \frac{(pe^t+1-p)^{20}}{exp(tx)} \left[\frac{exp(tx)}{(pe^t+1-p)^{20}} C_x^{20} p^x (1-p)^{20-x}\right]$

1. Direct Monte Carlo
   [1] gen $X_1, X_2, \cdots, X_n$ from $Bin(20, 0.4)$
   [2] Compute $Y_i = I(X_i \geq 16), \ i \in [1, n]$
   [3] Estimate using $\bar{Y}$
2. Importance Sampling
   [1] Generate $X_1, X_2, \cdots, X_n$ from tilted density, with $t = 1.8$
   [2] Compute
   $W_i = I(X_i \geq 16) \frac{(pe^t+1-p)^{20}}{exp(tX_i)}. \ i \in [1, n]$
   [3] Estimate using $\bar{W}$

Discrete Event Simulation

Condition: system where one or more phenomena of interest change value or change state at discrete points in time.
Key points:

1. Event cause the state of system to change
2. Change involves sequence of actions to be taken on the entities and variables being recorded.
3. Simulated time is constant while these actions take place.

Components

1. System
   A collection of objects that interact through time according to certain rules.
2. System State (SS)
   A collection of varaiables of interest in the system under study.
3. Entity
   An object in a system that requires explicit representation
4. Event
   A change in the system state
5. Event List (EL)
   A list of tureu or upcoming events

Generating Events from Nonhomogeneous Poisson Process

Algorithm 1 (Nonhomogeneous Poisson Process)

1. let $t = s$
2. generate $U \sim Unif[0, 1]$
3. Update $t = t - \frac{1}{\lambda} log(U)$
4. Gen $V \sim Unif[0, 1]$
5. If $V \leq \lambda(t)/\lambda$
   set $T_s = t$ and exit the algorithm
6. Else Return to step 2

R Implementation[Generate $Pois(\lambda_t)$]

```
genArrival <- function(s, lambda=6){
    t0 <- s; u <- runif(1)
    t0 <- t0 - (1/lambda)*log(u)
    v <- runif(1)
    while(v <= get.lambda(t0)/lambda){
        u <- runif(1)
        t0 <- t0 + (-1/lambda)*log(u)
        v <- runif(1)
    }
    t0
}
```

Single Server Queue

Interested to find T_p :=time past T that the last customer departs

Problem Set-up

Arrival time: $T_t \sim Pois(\lambda(t))$
ranges refer to timing of the day (e.g. 2nd hour etc)

$$\lambda(t) = \begin{cases} 2, & x \in [0,3) \cup [6,8) \\ 3, & x \in [3,4) \cup [5,6) \cup [8,9) \\ 6, & x \in [4,5) \end{cases}$$

Serving time: $Y \sim N\left(0.25, \frac{2}{60}^2\right)$
No additional arrivals allowed after 5pm (9hrs).

DES variable definition

1. System: service station
2. Entity: server, customer, $t :=$ current time
3. Event: arrival, departure of customers
4. SS: (n)
   $n :=$ num of customers in the system, exclude those who have left, including those at station and queue
5. EL: (t_A, t_D)
   $t_A, t_D :=$ time of next arrival, departure
6. Param: (t)
   $t :=$ current time

Initialising the model

1. $SS = (0)$
2. EL $= (T_0, \infty)$
3. Param $= (0)$

Updating System

1. $t_A \leq t_D, t_A \leq T$: arrival before T
   [1] Param $= (t = t_A)$
   [2] $SS = (n = n + 1)$
   [3] $t_A = T_t$
   [4] If $SS = 1$: update $t_D = t + Y$
2. $t_D < t_A, t_D \leq T$: departure before T
   [1] Param $= (t = t_D)$
   [2] $SS = (n = n - 1)$
   [3a] If $SS = 0$, update $t_D = \infty$
   [3b] else: set $t_D = t + Y$

3. $\min(t_A, t_D) > T, n > 0$: customer in system, after T

   [1] Param $= (t = t_D)$

   [2] $SS = (n = n - 1)$

   [3] If $SS > 0$, update $t_D = t + Y$

4. $\min(t_A, t_D) > T, n = 0$: no customer, after T

   [1] return $T_p = \max(t - T, 0)$

R code

```
# init
n <- 0; ta <- genPois(s=0); td <- Inf
now.time <- 0
end.time <- 10 #days to simulate
run <- TRUE
# helper functions
customerArrival <- function(){
    # simulate customer arrival
    now.time <<- ta
    n <<- n + 1
    ta <<- genPois(now.time)
    if(n==1){
        td <<- now.time + genServingTime()
    }
}
customerDeparture <- function(){
    # simulate customer departure
    now.time <<- td
    n <<- n - 1
    if(n==0){
        td <<- Inf
    } else {
        td <<- now.time + genServingTime()
    }
}
while(run){
    if(min(ta, td) <= end.time){
        # before T
        if(ta <= td){
            # arrival
            customerArrival()
        } else {
            # departure
            customerDeparture()
        }
    } else if (n > 0){
        # remaining customer, after T
        customerDeparture()
    } else {
        # no customer, after T
        Tp <- max(now.time - end.time, 0)
        run <- FALSE # end
    }
}
```

Min of Exp is Exp

$X_i \sim Exp(\lambda_i)$ independently, $X = \min\{X_1, X_2, \cdots, X_n\}$

$$\begin{aligned}
P(X \leq x) &= 1 - P(X > x) \\
&= 1 - P(X_1 > x, X_2 > x, \cdots, X_n > x) \\
&= 1 - \prod_{i=1}^{n} P(X_i > x) \\
&= 1 - \prod_{i=1}^{n} e^{-\lambda_i x} \\
&= 1 - e^{x \sum_{i=1}^{n} \lambda_i} \\
&= Exp(\sum_{i=1}^{n} \lambda_i)
\end{aligned}$$

Insurance Claims

Interested to find probability of non-negative capital after $T = 5$ months

Problem Set-up

Claim amount	$:= Y = 100W \sim \Gamma(3, 4/100)$
Premium amt per pholder	$:= c$
Event time	$:= X \sim Exp(\nu + n\mu + n\lambda)$
Type of event	$:= J$

$$\begin{aligned}
P(J = 1) = p_1 &= \frac{\nu}{\nu + n\mu + n\lambda} \\
P(J = 2) = p_2 &= \frac{n\mu}{\nu + n\mu + n\lambda} \\
P(J = 3) = p_3 &= \frac{n\lambda}{\nu + n\mu + n\lambda}
\end{aligned}$$

where 1. New policyholder join time $\sim Exp(\nu)$, 2. Pholders staying time $\sim Exp(\mu)$, 3. Claim event time $\sim Exp(\lambda)$

DES variable definition

1. System: insurance company

2. Entity: 1. claim, 2. customer, 3. capital

3. Event: 1. New policyholder, 2. Lost policyholder, 3. claim

4. SS: (n, a)
   $n :=$ num of policyholders
   $a :=$ current capital

5. EL: (t_E, J)
   $t_E :=$ time of next event
   $J :=$ type of event

6. Param $= (t, I)$
   $t :=$ current time
   $I :=$ indicator for non-negative capital

Initialising the model

1. $SS = (n_0, a_0), a_0 \geq 0$

2. $EL = (X, J)$

3. Param $= (0, 1)$

Updating System

1. $t_E > T$: after T

   [1] set $I = 1$ and exit

2. $t_E \leq T$: before T

   [1] $a = a + nc(t_E - t)$

   [2] $t = t_E$

   [3a] if J $= 1$: $n = n + 1$

   [3b] if J $= 2$: $n = n - 1$

   [3c] if J $= 3$:

   [3ci] if $Y > a$: set $I = 0$ and exit

   [3cii] else: set $a = a - Y$

   [4] update next event time $t_E = t + X$

   [5] update J

R code

```r
n <- 10; a <- 500 # preset n0 and a0
te <- genTime(); j <- genType()
now.time <- 0; end.time <- 10 # end time
I.capital <- 1 #sufficient capital
while(te <= end.time){
    a <<- a + n*premium*(te - now.time)
    now.time <<- te
    if(j==1){
        # new customer
        n <<- n + 1
    } else if (j==2){
        # customer quit
        n <<- n - 1
    } else {
        # claim event
        claim <- genClaim()
        if(claim > a){
            # if claim more than capital
            I.capital <<- 0
            return(I.capital)
        } else {
            # if claim less than capital
            a <<- a - claim
        }
    }
    j <<- genType()
    te <<- now.time + genTime()
}
```

Repair Problem

Interested to approximate $E(T)$, $T :=$ time system crashes

Problem Set-up

Failure time $:= X \sim F$
Service time $:= Y = G \sim Unif(a,b)$
n $:=$ num of required working machine
s $:=$ num of spare machines

DES variable definition

1. System: factory
2. Entity: 1.spare machine, 2.in-use machine, 3. repairman
3. Event: 1.failure, 2.completion of repair
4. SS = (SS)
   $SS :=$ num of machines down at time t,
   $0 \leq SS \leq n+s$
5. EL: $(t_1 \leq t_2 \leq \cdots \leq t_n, t^*)$
   $t_1, \cdots, t_n :=$ ordered times where n machines currently in use will fail
   $t^* :=$ time machine complete repair
6. Param $= (t)$
   $t :=$ current time

Initialising the model

1. $SS = (0)$
2. EL $= (X_1, \cdots, X_n, \infty)$
3. Param $= (0)$

$t_1, \cdots, t_n$ are ordered $X_1, \cdots, X_n \sim F$

Updating System

1. $t_1 < t^*$: failure
   [1] Param $= (t = t_1)$
   [2] $SS = (SS = SS + 1)$
   [3a] if $SS = s + 1$:
   exit
   [3b] else if $SS < s + 1$:
   update $(t_2, t_3, \cdots, t_n, X + t)$ as ordered vector
   [3bi] if $SS = 1$:
   update $t^* = t + Y$
2. $t_1 > t^*$: repair completion
   [1] Param $= (t = t^*)$
   [2] $SS = (SS = SS - 1)$
   [3a] if $SS > 0$:
   update $t^* = t + Y$
   [3b] else if $SS = 0$:
   update $t^* = \infty$

R code

```r
# init
n <- 5; s <- 2 # required and spare machine
failed <- 0 # num of failed machine
fail.time <- sort(sapply(1:n,
                  function(x) genFailure()))
repair.time <- Inf; now.time <- 0
# helper function
updateFailure <- function(){
    # update machine failure
    now.time <<- fail.time[1]
    failed <<- failed + 1
    if(failed == s+1){
        # failed more than required
        return(now.time)
    } else if (failed < s + 1) {
        # add in spare machine
        fail.time[1] <<- now.time +
                        genFailure()
        fail.time <<- sort(fail.time)
        if(failed == 1){
            # if only 1 failure,
            # gen repair time
            repair.time <<- now.time +
                            genRepair()
        }
    }
}
updateRepair <- function(){
    # update machine repaired
    now.time <<- repair.time
    failed <<- failed - 1
    if(failed > 0){
        # if there is machine to repair
        repair.time <<- now.time +
                        genRepair()
    } else if (failed == 0){
        # no more machine to repair
        repair.time <<- Inf
    }
}
while(failed <= s){
    if(fail.time[1] < repair.time){
        updateFailure()
    } else {
        updateRepair()
    }
}
```

Inventory Model

Interested in estimating expected profit up to fixed time T, profit $= R - H - C$

Problem Set-up

customer arrival time	$:= T \sim Pois(\lambda)$
demand	$:= D \sim G$
s	:= threshold
S	:= inventory limit
ordering policy	$:= (s, S)$, when $x < s$ and no outstanding order, ordered s.t. $x = S$
price	$:= r$
cost	$:= c(y)$
order time	$:= L$
holding cost	$:= h$

DES variable definition

1. System: store
2. Entity: 1. customer, 2. order
3. Event: 1. customer visiting store, 2. order arriving
4. SS: (x, y)
   $x :=$ inventory amount
   $y :=$ order amount
5. EL: (t_0, t_1)
   $t_0 :=$ time next customer arrive
   $t_1 :=$ time order fulfilled
6. Param $:= (t, H, R, C)$
   $t :=$ current time
   $H :=$ holding cost
   $R :=$ revenue
   $C :=$ cost

Initialising the model

1. $SS = (S, 0)$
2. EL $= (X, \infty)$
3. Param $= (0, 0, 0, 0)$

Updating System

1. $t_0 < t_1$: new customer arrival
   [1] Update Param
   $H = H + (t_0 - t)xh$
   $t = t_0$
   $w = \min(D, x)$
   $R = R + wr$
   [2] update SS: $x = x - w$
   [3] update next arrival time: $t_0 = t + T$
   [4] if $(x < s)\&(y = 0)$:
   update $y = S - x$
   update $t_1 = t + L$
2. $t_0 \geq t_1$: order arrival
   [1] update Param
   $H = H + (t_0 - t)xh$
   $t = t_1$
   $C = C + c(y)$
   [2] update $x = x + y$
   set $y = 0$
   set $t_1 = \infty$

R code

```r
x <- S; y <- 0
t0 <- genArrival(); t1 <- Inf
now.time <- 0; end.time <- 10 #num of days
H <- 0; R <- 0; Cost <- 0
customerArrival <- function(){
    # customer arrive in store
    H <<- H + (t0 - now.time)*x*h
    now.time <<- t0
    demand <- genDemand()
    w <- min(demand, x)
    R <<- R + w * r
    x <<- x - w
    t0 <<- now.time + genArrival()
    if((x < s)&(y==0)){
        # if inventory less than threshold
        # and no pending orders
        y <<- S - x
        t1 <<- now.time + L
    }
}
orderArrival <- function(){
    # order arrive in store
    H <<- H + (t0 - now.time) * x * h
    now.time <<- t1
    Cost <<- Cost + cost(y)
    y <<- 0
    t1 <<- Inf
}
while(min(t0, t1) <= end.time){
    if(t0 < t1){
        customerArrival()
    } else {
        orderArrival()
    }
}
```

Queueing System with Two Parallel Servers

Interested in time spent in system by each customer, number of services performed by each server

Problem Set-up

service time by server i	$:= Y_i \sim G_i$
arrival time of customer	$:= T_t \sim Pois(\lambda(t))$

customers are labelled by their arrival, customer 1 arrive earlier than customer 2 etc.
If both servers are occupied, customer join queue, else customer join server 1 if it's free else server 2

DES variable definition

1. System: two parallel servers
2. Entity: 1. customer, 2. server
3. Event: 1. customer arrival, 2. customer departure
4. SS: (n, i_1, i_2)
   $n :=$ total num of customers in system
   $i_1 := i^{th}$ customer is with server 1
   $i_2 := i^{th}$ customer is with server 2
5. EL: (t_A, t_1, t_2)
   $t_A :=$ next arrival time
   $t_1 :=$ completion time for server 1
   $t_2 :=$ completion time for server 2
6. Param: (t, N_A, C_1, C_2)
   $t :=$ time
   $N_A :=$ number of arrival by time t
   $C_1 :=$ number of customers served by server 1
   $C_2 :=$ number of customers served by server 2
7. A(n), D(n): matrix of nx1
   n^{th} entry represents the arrival time and departure time of customer n

Initialising the model

1. $SS = (0, 0, 0)$
2. EL $= (T_0, \infty, \infty)$
3. Param $= (0, 0, 0, 0)$
4. An = Dn = NULL

Updating System

1. $\min(t_A, t_1, t_2) = t_A$: new arrival

   [1] Param $= (t = t_A, N_A = N_A + 1, C_1, C_2)$

   [2] update $A(N_A) = t$

   [3] update $t_A = T_t$

   [4a] if $n = 0$:
   update $SS = (n = 1, i_1 = N_A, i_2 = 0)$
   update $t_1 = t + Y_1$

   [4b] if $n = 1$:
   update $SS = (n = 2, i_1 = N_A, i_2)$
   update $t_1 = t + Y_1$ (or update server 2)

   [4c] if $n > 1$:
   update $SS = (n = n + 1, i_1, i_2)$

2. $t_1 < t_A, t_1 \leq t_2$: departure from server 1

   [1] Param $= (t = t_1, N_A, C_1 = C_1 + 1, C_2)$

   [2] update $D(i_1) = t$

   [3a] if $n = 1$:
   update $SS = (n = 0, i_1 = 0, i_2 = 0)$
   update $t_1 = \infty$

   [3b] if $n = 2$:
   update $SS = (n = 1, i_1 = 0, i_2)$
   update $t_1 = \infty$

   [3c] if $n > 2$:
   let $m = \max(i_1, i_2)$
   update $SS = (n - 1, m + 1, i_2)$
   update $t_1 = t + Y_1$

3. $t_2 < t_A, t_1 > t_2$: departure from server 2

   [1] Param $= (t = t_2, N_A, C_1, C_2 = C_2 + 1)$

   [2] update $D(i_2) = t$

   [3a] if $n = 1$:
   update $SS = (n = 0, i_1 = 0, i_2 = 0)$
   update $t_2 = \infty$

   [3b] if $n = 2$:
   update $SS = (n = 1, i_1, i_2 = 0)$
   update $t_2 = \infty$

   [3c] if $n > 2$:
   let $m = \max(i_1, i_2)$
   update $SS = (n - 1, i_1, m + 1)$
   update $t_2 = t + Y_2$

R code

```r
# init
n <- 0; i1 <- 0; i2 <- 0
ta <- genArrival(s=0); t1 <- Inf; t2 <- Inf
now.time <- 0; N <- 0; C1 <- 0; C2 <- 0
An <- NULL; Dn <- NULL
# helper function
updateArrival <- function(){
    now.time <<- ta; N <<- N + 1
    An[N] <<- now.time
    ta <<- genArrival(s=now.time)
    if(n==0){
        n <<- n + 1
        i1 <<- N
        i2 <<- 0
        t1 <<- now.time + genServTime.1()
    } else if (n==1){
        n <<- 2
        if(max(t1, t2)==t1){
            # server 1 is free
            i1 <<- N
            t1 <<- now.time +
                        genServTime.1()
        } else{
            # server 2 is free
            i2 <<- N
            t2 <<- now.time +
                        genServTime.2()
        }
    } else {
        n <<- n + 1
    }
}
updateDeparture.1 <- function(){
    now.time <<- t1; C1 <<- C1 + 1
    Dn[i1] <<- now.time
    if(n==1){
        n <<- 0
        i1 <<- 0; i2 <<- 0
        t1 <<- Inf
    } else if (n==2){
        n <<- 1
        i1 <<- 0
        t1 <<- Inf
    } else {
        m <- max(i1, i2)
        n <<- n - 1
        i1 <<- m + 1
        t1 <<- now.time + genArrival()
    }
}
updateDeparture.2 <- function(){
    now.time <<- t2; C2 <<- C2 + 1
    Dn[i2] <<- now.time
    if(n==1){
        n <<- 0
        i1 <<- 0; i2 <<- 0
        t2 <<- Inf
    } else if (n==2){
        n <<- 1
        i2 <<- 0
        t2 <<- Inf
    } else {
        m <- max(i1, i2)
        n <<- n - 1
        i2 <<- m + 1
        t2 <<- now.time + genArrival()
    }
}
run <- TRUE
while(run){
    if(min(ta, t1, t2)<=end.time){
        # server running
        if(min(ta, t1, t2) == ta){
            updateArrival()
        } else if (t1 <= t2){
            updateDeparture.1()
        } else {
            updateDeparture.2()
        }
    } else if (n>0){
        # remaining jobs
        if(t1 <= t2){
            updateDeparture.1()
        } else {
            updateDeparture.2()
        }
    } else {
        # no jobs, after end time
        run <- FALSE
    }
}
```

Variance Reduction

Antithetic Variable	Control Variable
1. Need same distribution for X and X'	1. X and Y have different distribution
2. Need negative correlation	2. Just need some correlation
3. The mean of X is being estimated	3. The mean of Y must be known
4. No estimation needed	4. Need to estimate c^*

Antithetic Variables

Definition1 (Antithetic Pairs)

X_1, X_2 are defined to be an antithetic pair if they are identically distributed but negatively correlated

Estimation without Antithetic Random Variable

- Identify RV X and write a program to simulate it
- Generate i.i.d sample $X_1, \cdots, X_n \sim X$
- $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$, $s^2 = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})^2$
- $(1-\alpha)100\%$ CI: $\bar{X} \pm z(\alpha/2)s/\sqrt{n}$

Estimation with Antithetic RV

- Identify RV X and write a program to simulate it
- Generate i.i.d sample $X_1, \cdots, X_{n/2}$ and their antithetic pairs $X'_1, \cdots, X'_{n/2}$, where n is even integer
- Set $Y_i = \frac{1}{2}(X_i + X'_i)$
- $\bar{Y} = \frac{1}{n/2}\sum_{i=1}^{n/2} Y_i$, $s^2 = \frac{1}{n/2-1}\sum_{i=1}^{n/2}(Y_i - \bar{Y})^2$
- $(1-\alpha)100\%$ CI: $\bar{Y} \pm z(\alpha/2)s/\sqrt{n/2}$

$var(\bar{Y}) \le var(\bar{X})$

General Method: $U, 1-U$

Theorem 2 (Correlation for Monotone Functions)

Suppose h is a function of d variables, and is a monotone function of each of its coordinates. In other words, for any i from 1 to d, it holds that if $x_i \le y_i$, then either $h(x_1, x_2, \cdots, x_i, \cdots, x_d) \le h(x_1, x_2, \cdots, y_i, \cdots, x_d)$ or $h(x_1, x_2, \cdots, x_i, \cdots, x_d) \ge h(x_1, x_2, \cdots, y_i, \cdots, x_d)$.

If the above condition holds, then $Cov[h(U_1, \cdots, U_d), h(1-U_1, \cdots, 1-U_d)] \le 0$

$$
\begin{aligned}
X_i &= h(U_1, \cdots, U_d) \\
X'_i &= h(1-U_1, \cdots, 1-U_d) \\
X &= h(F^{-1}(U_1), F^{-1}(U_2), \cdots, F^{-1}(U_d)) \\
X'_i &= h(F^{-1}(1-U_1), F^{-1}(1-U_2), \cdots, F^{-1}(U_d))
\end{aligned}
$$

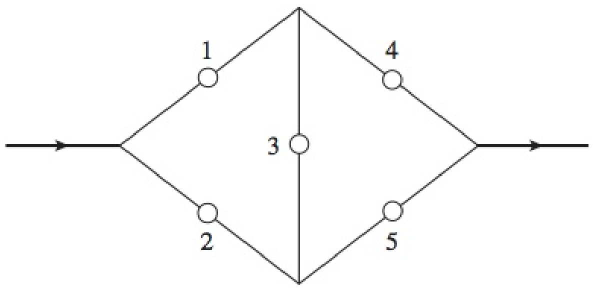
Note: check $\frac{d}{dx} > 0$ for strictly increasing condition

Example1: Monte Carlo Integration with Antithetic Pairs

$\mu = E(e^U) = \int_0^1 e^x dx$

1. Generate $U \sim unif(0,1)$
2. Set $X_1 = e^U$, $X'_1 = e^{1-U}$
3. Estimate $E(e^U)$ using $\bar{X} = (X_1 + X'_1)/2$

Example2: Simulating Reliability



$s_i = 1$ if functioning and $s_i = 0$ if failed.
$\Phi = 1 \Leftrightarrow$ overall system is functioning

$$
\begin{aligned}
&\Phi(s_1, s_2, s_3, s_4, s_5) = \max\{s_1 s_3 s_5, s_2 s_3 s_4, s_2 s_5, s_1 s_4\} \\
&P(S_i = 1) = p_i = 1 - P(S_i = 0) \\
&r(p_1, p_2, p_3, p_4, p_5) = P(\Phi = 1) = E(\Phi)
\end{aligned}
$$

Using Antithetic Pairs

1. Generate $U_1, U_2, \cdots, U_5 \sim Unif(0,1)$
2. Set $S_i = 1$ if $U_i \le p_i$ else 0
3. Compute $W_i = \Phi(S_1, \cdots, S_5)$
4. Set $S'_i = 1$ if $1 - U_i \le p_i$ else 0
5. Compute $W'_i = \Phi(S'_1, \cdots, S'_5)$

Specific Method for Normal

if $X \sim N(\mu, \sigma^2)$, then antithetic RV is $X' = 2\mu - X$

$$
\begin{aligned}
X_i &= h(W_1, \cdots, W_d) \\
X'_i &= h(2\mu_1 - W_1, \cdots, 2\mu_d - W_d)
\end{aligned}
$$

Control Variables

estimate $\mu = E(X)$, require μ_Y to be known

$$
\begin{aligned}
\hat{\mu} &= X + c^*(Y - \mu_Y) \\
c^* &= -\frac{Cov(X,Y)}{Var(Y)}
\end{aligned}
$$

$$
Var(X + c^*(Y - \mu_Y)) = Var(X) - \frac{[Cov(X,Y)]^2}{Var(Y)}
$$

Note: (var reduction / var(X) = corr(X,Y))

$$
E(U) = \frac{1}{2} \quad E(U^2) = \frac{1}{3}
$$

$$
\rho(X,Y)^2 = \frac{[Cov(X,Y)]^2}{Var(Y)} \cdot \frac{1}{Var(X)}
$$

Using Control Variables in Simulation

1. Identify a RV X and write a program to simulate it
2. Generate $X_1, X_2, \cdots, X_n$
3. Identify $Y_1, \cdots, Y_n$ that was part of the simulation with a known μ_Y
4. Estimate $\hat{c}^*$ from data
5. Set $W_i = X_i + c^*(Y_i - \mu_Y)$
6. Compute $\bar{W} = \frac{1}{n}\sum_{i=1}^n W_i$, and $s^2 = \frac{1}{n-1}\sum_{i=1}^n (W_i - \bar{W})^2$
7. Then $(1-\alpha)100\%$ CI: $\bar{W} \pm z(\alpha/2)s/\sqrt{n}$

Back to Example1: Integrating e^x

Using U as control variate, $E(U) = 0.5$
$Var(e^U + c^*(U - 1/2)) << Var(e^U)$
use $X_i = e^U + c^*(U - 1/2)$

Back to Example2: Reliability Function

Using $Y = \sum_{i=1}^5 S_i$ as control variate,
$E(Y) = E(\sum_{i=1}^5 S_i) = \sum_{i=1}^n p_i$
use $X_i = W_i + c^*(Y_i - \sum_{i=1}^5 p_i)$

Var of estimator

Control variables : $Var(W)$
Antithetic pair : $Var(Y)$

Importance Sampling

Importance sampling can be used more generally to reduce variance of estimator. However, it is difficult to choose an alternative density to use as theoretical analysis is often difficult.

Example3: Quality Control Studies

Let $X_1, X_2, \cdots$ be a sequence of i.i.d $N(\mu, 1)$ RV with $\mu < 0$
Wish to know when will partial sum exceed limits

$$S_n = \sum_{i=1}^{n} X_i$$
$$N = \min\{n : S_n < -A \text{ or } S_n > B\}$$

A, B are fixed known positive numbers, N is stopping time
Interested in estimating $P(S_n > B)$
Instead of $N(\mu, 1)$, consider $N(-\mu, 1)$ and estimate with $\bar{W}$

$$W_i = I\left(\sum_{i=1}^{N} X_i > B\right) \frac{\prod_{i=1}^{N} f_\mu(X_i)}{\prod_{i=1}^{N} f_{-\mu}(X_i)}$$
$$\because \frac{f_\mu(x)}{f_{-\mu(x)}} = e^{2\mu x}$$
$$\Rightarrow W_i = I\left(\sum_{i=1}^{N} X_i > B\right) exp(2\mu \sum_{i=1}^{N} X_i)$$

Statistical Validation Techniques

General step for hypothesis testing

1. Take note of assumption
2. Specify H_0, H_1 and significant level $\alpha = 0.05$
3. Compute test statistics
4. Obtain p-value of obs a more extreme test statistics
5. State the conclusion (reject or do not reject H_0)

Goodness of Fit Tests: Discrete data

$$H_0 : P(Y_j = i) = p_i, i \in [0, K-1]$$

Let $N_i :=$ number of $Y_j = i$
For a fixed i

$$X_j = \begin{cases} 1, Y_j = i \\ 0, Y \neq i \end{cases}$$

$j \in [1, n]$. Then under $H_0, P(X_j = 1) = P(Y_j = i) = p_i$

$$N_i = \sum_{j=1}^{n} X_j \sim Bin(n, p_i)$$

Therefore, under H_0, $E(N_i) = np_i$. Using $(N_i - np_i)^2$ as indication for how likely our assumption is true

$$T = \sum_{i=0}^{K-1} \frac{(N_i - np_i)^2}{np_i} \sim \chi^2_{K-1}$$

N_i := num of observation that took value i
p_i := postulated probability of observing the value i
n := num of obs we made
K := num of different possible values Y_i can take
When n is large $(n \geq 50), T \sim \chi^2_{K-1}$
When n is small, use simulation to estimate the p-value

χ^2 test summary

Summary of χ^2 test when all parameters in H_0 are fully specified

1. independent set of $Y_1, \cdots, Y_n$ from real-life process
2. $H_0 : Y_1, \cdots, Y_n$ are from pmf p_i and H_1 is not
3. $T = \sum_{i=0}^{K-1} \frac{(N_i - np_i)^2}{np_i} \sim \chi^2_{K-1}$
4. p-value is area under χ^2_{K-1} to the right of observed test stat

Example 1 (χ^2 Goodness of Fit test)

$H_0 : p_i = 0.2, i \in [0, 4]$
$N_i = \{12, 5, 19, 7, 7\}$, $E(N_i) = 10$
$T = (4 + 25 + 81 + 9 + 9)/10 = 12.8$
$P_{H_0}(T \geq 12.8) = P(\chi^2_4 \geq 12.8) = 0.0123$

```
pchisq(12.8, df=4, lower.tail=FALSE)
```

χ^2 test after estimation of parameters

If pmf in H_0 is not completely specified and we need to estimate m parameters

$$\chi^2_{K-1-m}$$

Example 2 (Comparing to $Pois(\lambda)$ Distribution)

i	0	1	2	3	4	5 or more
N_i	6	2	1	9	7	5
$\hat{p}_i$	0.05	0.1596	0.2312	0.2237	0.1622	0.1682

1. find $\hat{\lambda} = \frac{\sum_{i=1}^{30} Y_i}{30} = 87/30$, $Y_i = N_i \cdot i$
2. $T_{obs} = \sum_{i=0}^{5} \frac{(N_i - 30\hat{p}_i)^2}{30\hat{p}_i} = 19.887$
3. $P(\chi^2_{K-1-m=6-1-1=4} \geq 19.887) = 0.0005$

Estimating p-value by Simulation

Algorithm 1 (Estimating p-value in Goodness-of-fit Test)

1. use observed data $Y_1, Y_2, \cdots, Y_n$ to compute the observed test statistics
$$T_{obs} = \sum_{i=0}^{K-1} \frac{(N_i - np_i)^2}{np_i}$$
2. For $s \in [1, M]$, M is the max iter
   [1] Generate $Y_1^s, Y_2^s, \cdots, Y_n^s$ from pmf p_i in H_0
   [2] Compute $N_0^s, N_1^s, \cdots, N_{K-1}^s$ from $Y_1^s, \cdots, Y_n^s$
   [3] Compute $T_s = \sum_{i=0}^{K-1} \frac{(N_i^s - np_i)^2}{np_i}$
   [4] Set $V_s = I(T_s \geq T_{obs})$
3. estimate p-value with
$$\bar{V} = \frac{1}{M} \sum_{s=0}^{M} V_s$$

Goodness of Fit Tests: Continuous data

Forming Discrete Data from Continuous

1. Bin Y_i into K distinct intervals $(-\infty, y_1], (y_1, y_2], \cdots, (y_{K-1}, \infty)$
2. Set $Y_i^d = i - 1$ if Y_j lies in interval $(y_{i-1}, y_i]$. Set up N_i and test goodness of fit
3. $H_0 : P(Y_j^d = i) = F(Y_{i=1}) - F(y_i), i \in [0, K-1]$

Kolmogorov Smirnov Test

where $y_{(j)}$ are the ascending ordered Y_j, F_e is empirical cdf, $y := F(x)$

$$F_e(x) = \frac{\#i : Y_i \le x}{n}$$
$$D = \max_x |F_e(x) - F(x)|$$
$$= \max\left\{\frac{j}{n} - F(y_{(j)}), F(y_{(j)}) - \frac{j-1}{n}\right\}, j \in [1, n]$$

$$P_F(D \ge d) = P_F\left(\max_{0\le y\le 1} |\frac{\#i : U_i \le y}{n} - y| \ge d\right)$$

calculate max $\{\frac{j}{n} - F(y_{(j)}), F(y_{(j)}) - \frac{j-1}{n}\}$

1. get $F(y_i)$
2. sort $F(y_i)$ ascending
3. calculate max $\{\frac{j}{n} - F(y_{(j)}), F(y_{(j)}) - \frac{j-1}{n}\}$

Algorithm 2: Estimating p-value in KS test by simulation

For $s \in [1, M]$, $M :=$ number of iteration to run

1. Generate $U_1^{(s)}, U_2^{(s)}, \cdots, U_n^{(s)}$ i.i.d from unif(0,1)
2. Compute
$$W_k = \max_{0\le y\le 1} |\frac{\#i : U_i \le y}{n} - y|$$
3. Set $V_s = I(W_k \ge D_{obs})$

Estimate p-value with

$$\bar{V} = \frac{1}{M}\sum_{s=1}^{M} V_s$$

Remember to sort the generated $U_i^{(s)}$

Two Sample Rank Sum Test

$X_1, \cdots, X_n, Y_1, \cdots, Y_m$, data from simulation and real world
$H_0 : n + m$ values are iid

1. order $n + m$ values
2. R_i denote the rank of the X_i among the $n + m$ valuse
3. $R = \sum_{i=1}^n R_i$ as the test stat
4. X_i, Y_i from diff dist for extreme large or small R
5. when n, m are large

$$R \sim N(\frac{n(n+m+1)}{2}, \frac{nm(n+m+1)}{12}$$
$$\frac{R - \frac{n(n+m+1)}{2}}{\sqrt{\frac{nm(n+m+1)}{12}}} \sim N(0,1)$$
$$r^* = \frac{r - \frac{n(n+m+1)}{2}}{\sqrt{\frac{(nm(n+m+1))}{12}}}$$
$$\text{p-value} = \begin{cases} 2P(Z < r^*) & r \le n(n+m+1)/2 \\ 2P(Z > r^*) & r > n(n+m+1)/2 \end{cases}$$

Note: if tie, take average rank. Remember to c.c.

Example 5 (Two Sample Rank Sum test)

$X = \{132, 104, 162, 171, 129\}$,
$Y = \{107, 94, 136, 99, 114, 122, 108, 130, 106, 88\}$
$rank(X) = R_i = [12, 4, 14, 15, 10], R = \sum R_i = 55$
p-value $= 2P_{H_0}(R \ge 55) \approx 2P(Z \ge \frac{54.5-40}{\sqrt{50(16)/12}})$, cc

Validating Poisson Processes

Nonhomogeneous Poisson Process

$N_i :=$ num of arrivals on day i

$$N_i \sim Pois(m(T))$$
$$m(T) = \int_0^T \lambda(s)ds$$

Then using $E(N_i)$ and goodness of fit test

Homogenous Poisson Process

Key assumption to test, where T is the total time interval

$$H_0 : \text{arrival time } \sim unif(0, T)$$
$$H_1 : \text{not uniform}$$

Note: it's arrival time and not inter-arrival time (which follow exp dist)

Quick tips

Easiest way to generate RV

Ber	: $u \sim unif(0, 1)$, set $X = 1$ if $u \ge p$ else 0
Binom	: generate n Ber with p
Pois Process	: homogeneous, thining method
Exp	: numeric inversion $X = -\frac{1}{\lambda}log(u)$
Gamma	: generate α Exp with λ
Normal	: Box-Muller transformation with u_1, u_2

Finding simulation algo cdf

Use $P(Y \le y \cap U \ge p)$ instead of conditional probability