Lisa Over
Homework #3
February 3, 2015

1. Beta Probabilities: X ~ Beta(2,5)

Monte Carlo Integration

Code

```
alpha <- 2
beta <- 5
bet.s <- NULL
x.s <- NULL
N <- 25000
a <- 0.5
b <- 1
#Generate N independent random uniform variables
for(i in 1:N) {
        x <- runif(1)
        x.s <- c(x.s,x)
}
#Use Monte Carlo integration x=y(b-a)+a transformation with a=0.5 and b=1 to obtain vector
with beta values uniform on 0.5 to 1 – find mean(g(x.s*( b-a) + a)*(b-a)) – note g(x)*(b-a)
beta.s <- (gamma(alpha + beta)/(gamma(alpha)*gamma(beta)))*(x.s*(b-a) + a)^(alpha-1)*
(1-(x.s*(b-a) + a))^(beta-1)*(b-a)
#mean of beta.s is the probability that values are greater than 0.5
mean(beta.s)
```

P(X>1/2)

#[1] 0.1074668

Rejection Sampling and Uniform Proposal Density

Code

```
alpha <- 2
beta <- 5
#A >= f(1/5) because d/dy(f(x)=0 when x=1/5 so A >= f(1/5)=1536/625=2.4576
A <- (gamma(alpha + beta)/(gamma(alpha)*gamma(beta)))*(1/5)^(alpha-1)*(1-(1/5))^(beta-1)
#vector for normal independent random variables
beta.s <- NULL
p.s <- NULL
t <- 25000
#Generate t beta independent random variables using rejection sampling with uniform
while(length(beta.s) < t) {
        #Generate 1 uniform independent random variable
        y.unif <- runif(1)
```

Lisa Over
Homework #3
February 3, 2015

```
	#Calculate the probability that the previously generated uniform is from the beta distribution
	denom <- A*1
	num <- (gamma(alpha + beta)/(gamma(alpha)*gamma(beta)))*(y.unif)^(alpha-1)*(1-(y.unif))^(beta-1)
	prob <- num/denom
	#Generate a uniform independent random variable to use as a probability
	#If the probability that the y.unif uniform could be from a beta distribution is greater than the probability represented by this new uniform random variable, select the y.unif uniform and store in beta.s
	if(runif(1) < prob) {
		beta.s <- c(beta.s,y.unif)
	}
}
#Detemine the number of values that are greater than 0.5
length(beta.s[beta.s > .5])
#Calculate the proportion of values greater than 0.5
length(beta.s[beta.s > .5])/length(beta.s)
```

P(X>1/2)

[1] 0.10888

PBETA FUNCTION IN R

```
#pbeta computes less than first parameter (.5) so answer is 1-pbeta(.5,2,5)

1-pbeta(.5,2,5)
```

P(X>1/2)

#[1] 0.109375

2. STANDARD NORMAL PROBABILITIES: X ~ N(0,1)

MONTE CARLO INTEGRATION

Code

```
norm.s <- NULL
x.s <- NULL
N <- 25000
a <- 0
b <- 1.96
#Generate N independent random uniform variables
for(i in 1:N) {
	x <- runif(1)
```

Lisa Over
Homework #3
February 3, 2015

```
        x.s <- c(x.s,x)
}
#Use Monte Carlo integration x=y(b-a)+a transformation with a=0 and b=1.96 to obtain vector with normal values uniform on 0 to 1.96 – find 2*mean(g(x.s*(b-a) + a)*(b-a)) – note g(x)*(b-a)
norm.s <- (1/sqrt(2*pi))*exp((-(x.s*(b-a) + a)^2)/2)*(b-a)
#2 times the mean of norm.s is the probability that values are between -1.96 and 1.96
mean(norm.s)*2
```

P(-1.96<X<1.96)

#[1] 0.9486925

REJECTION SAMPLING WITH EXPONENTIAL REFERENCE DENSITY

Code

```
t <- 15000
#Find A such that A*(exp)^-x [i.e., A*q(y)] is greater than or equal to 1/sqrt(2*pi)*e^(-y^2/2) [i.e., f(y)] for all y so maximize f(y)/q(y) gives A = sqrt(exp(1)/(2*pi))
A <- sqrt(exp(1)/(2*pi))
#vector for normal independent random variables
norm.s <- NULL
lambda <- 1
#Generate t normal independent random variables using alternative rejection sampling with exponential dist
while(length(norm.s) < t) {
        #Generate 1 exponential independent random variable using uniform random var and integrated exponential density e^(-lambda*x)
        y.exp <- log(runif(1))/-lambda
        #Calculate the probability that the previously generated exp is from the normal distribution
        denom <- A*exp(-y.exp)
        num <- (1/sqrt(2*pi))*exp(-(y.exp^2)/2)
        prob <- num/denom
        #Generate a uniform independent random variable to use as a probability
        #If the probability that the exponential could be from a normal distribution is greater than the probability represented by the uniform random variable, select the exponential and store in norm.s
        if(runif(1) < prob) {
                #Generate a uniform random variable to use to make certain values negative
                if(runif(1) < .5) {
                        y.exp <- y.exp*-1
                }
                norm.s <- c(norm.s,y.exp)
        }
}
```

Lisa Over
Homework #3
February 3, 2015

length(norm.s[norm.s > -1.96 & norm.s < 1.96])/length(norm.s)

P(-1.96<X<1.96)

#[1] 0.9518667

PNORM FUNCTION IN R

1 - 2*pnorm(-1.96,0,1)

P(-1.96<X<1.96)

#[1] 0.9500042

3. BIVARIATE NORMAL PROBABILITIES (POLAR METHOD)

STANDARD NORMAL REALIZATIONS

Code

```
polar <- function() {
	t <- 25000

	x.s <- NULL
	y.s <- NULL
	v1 <- 0
	v2 <- 0

	for(i in 1:t) {
		#initialize ssqr to be greater than 1 so loop will start
		ssqr <- 2
		#This loop replaces the use of sin and cos to save computational time
		#generate two random numbers on the unit square (u1 and u2), transform them
(v1 and v2), and to generate a number on the unit circle (ssqr)
		while(sqrt(ssqr) > 1) {
			u1 <- runif(1)
			u2 <- runif(1)
			v1 <- 2*u1-1
			v2 <- 2*u2-1
			ssqr <- v1^2 + v2^2
		}
		#Transform joint density of x and y and rewrite in polar coordinates
		#vectors x.s and y.s are both normal realizations from the joint density (3D
"bell" shape)
		x <- sqrt(-2*log(ssqr))*v2/sqrt(ssqr)
		x.s <- c(x.s,x)
		y <- sqrt(-2*log(ssqr))*v1/sqrt(ssqr)
```

Lisa Over
Homework #3
February 3, 2015

```
                y.s <- c(y.s,y)

                if(i%%10000 == 0) {
                        print(i)
                }
        }
        #return vectors as a list with labels "x" and "y"
        vectors <- list("x" = x.s, "y" = y.s)
        return(vectors)
}

system.time(vectors <- polar())

hist(vectors$x)
hist(vectors$y)

#Use the normal realizations to find the value k that satisfies P(sqrt(X^2 + Y^2)) < k = 1/2
hyp <- sqrt(vectors$x^2 + vectors$y^2)
median(hyp)
```

Value of k such that P(sqrt(x^2 + y^2) < k)=1/2

#[1] 1.177607