#1a

```
t <- 7500
#alpha = n = 10 and beta = lambda = 4
n <- 10
lambda <- 4
#vector for exponential random variables
ex.s <- NULL
#vector for gamma random variables
gam.s <- NULL

#Generate t independent gamma random variables
for(j in 1:t) {
	#Generate n independent exponential random variables with parameter
lamda
	for(i in 1:n) {
		y <- log(runif(1))/-lambda
		ex.s <- c(ex.s,y)
	}
	#Sum the n values to generate one gamma and store the value in the vector
g.s
	gam.s <- c(gam.s,sum(ex.s))
	#Reset the vector used to store exponential values
	ex.s <- NULL
}
```

mean(gam.s)

```
#[1] 2.506528
```

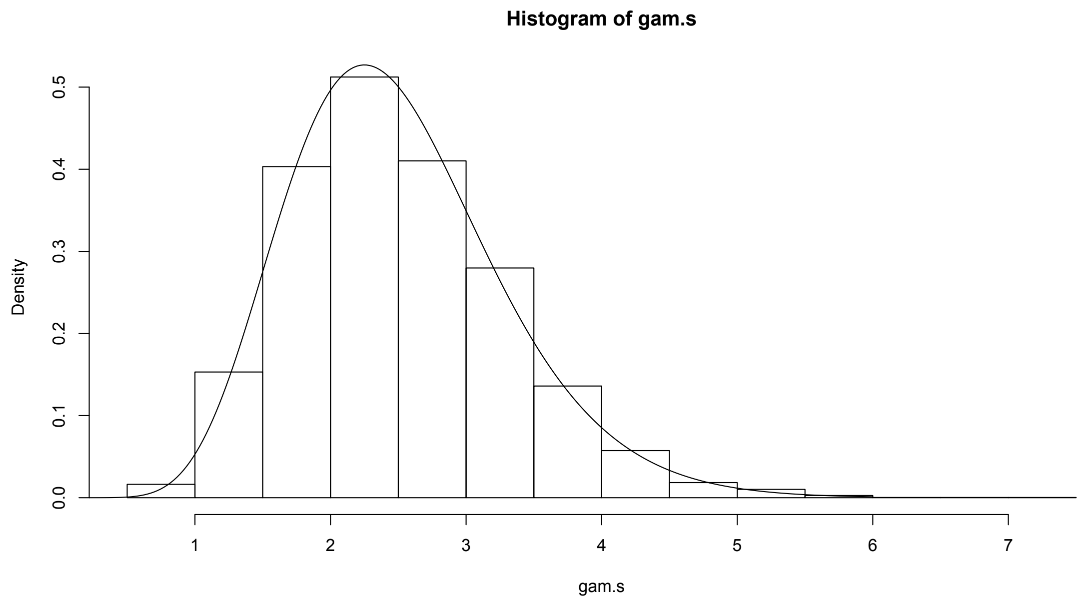
var(gam.s)

```
#[1] 0.645398
```

#Theoretical mean is alpha/beta and variance is alpha/beta^2

```
n/lambda
#[1] 2.5
n/lambda^2
#[1] 0.625

hist(gam.s,probability=T)
lines(seq(0,6,length=250),dgamma(seq(0,6,length=250),10,4))
```



#1b

```
t <- 7500
#vector for Cauchy random variables
cau.s <- NULL

#Generate t independent Cauchy random variables and store in cau.s
for(j in 1:t) {
	y <- tan(pi*runif(1)-pi*.5)
	cau.s <- c(cau.s,y)
}
```

1.c. Rejection Sampling

Find A such that $A \times q(y) \geq f(y)$ for all y.

$$q(y) = \frac{1}{\pi \times (1 + y^2)}$$

$$f(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}$$

$$A \geq \frac{\pi \times (1 + y^2)}{\sqrt{2\pi}} e^{-y^2/2}$$

$$\frac{d}{dy}\left(\frac{\pi \times (1 + y^2)}{\sqrt{2\pi}} e^{-y^2/_2}\right) = \sqrt{\frac{\pi}{2}}(-e^{-y^2/_2})y(y^2 - 1)$$

Roots: -1, 0, 1

$$A \geq \frac{\pi \times (1 + (1)^2)}{\sqrt{2\pi}} e^{-(1)^2/_2} \geq 1.52$$

```
t <- 7500
#Find A such that A*(pi)^-1*(1+y^2)^-1 [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
A <- sqrt(2*pi)*exp(-.5)
#vector for normal independent random variables
norm.s <- NULL
p.s <- NULL

#Generate t normal independent random variables using rejection sampling with
Cauchy
while(length(norm.s) < t) {
	#Generate 1 Cauchy independent random variable
	y.cau <- tan((pi*runif(1))-(pi*.5))
	#Calculate the probability that the previously generated Cauchy is from the
normal distribution
	denom <- A*(1/(pi*(1 + y.cau^2)))
	num <- (1/sqrt(2*pi))*exp(-(y.cau^2)/2)
	prob.cau <- num/denom
	#Generate a uniform independent random variable to use as a probability
	#If the probability that the Cauchy could be from a normal distribution is
greater than the probability represented by the uniform random variable, select the
Cauchy and store in norm.s
	if(runif(1) < prob.cau) {
		norm.s <- c(norm.s,y.cau)
	}
}
```

mean(norm.s)

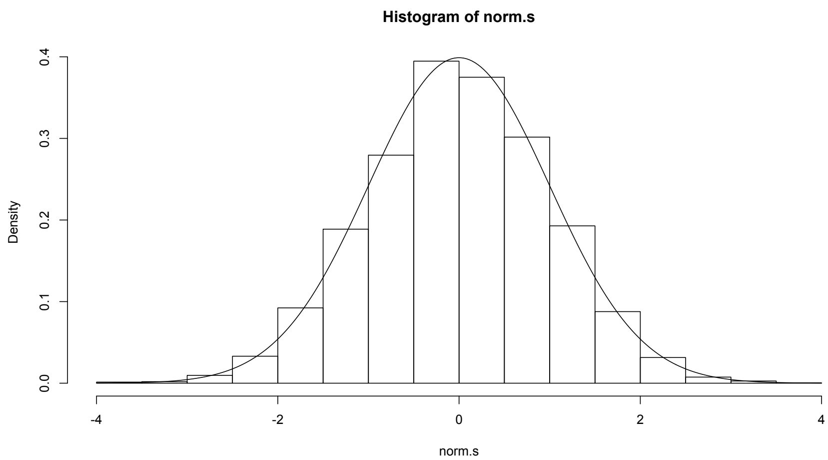
#[1] 0.0008523671

var(norm.s)

#[1] 0.9979126

#Theoretical mean is mu=0 and variance is sigma^2=1

```
hist(norm.s,probability=T)
lines(seq(-4,4,length=250),dnorm(seq(-4,4,length=250)))
```



#2a

```
t <- 7500
#probability of success
p <- 0.83
#vector for geometric variables
geo.s <- NULL
#counter to count number of trials necessary to obtain the first success
#start counter at 1 to include the success
c <- 1

#Generate t independent geometric random variables
for(j in 1:t) {
	#Generate independent Bernoulli trials until runif(1) is less than p,
increment counter each time
	repeat {
		if(runif(1) > p) {
			c <- c + 1
		}
		else {
			break
		}
	}
	#Add the count to geometric vector
	geo.s <- c(geo.s,c)
	#Reset the counter
	c <- 1
}
```

mean(geo.s)

```
#[1] 1.1984
```

var(geo.s)

```
#[1] 0.24387
```

#Theoretical mean is (1-p)/p and variance is (1-p)/p^2

```
#R counts failures until the first success so the simulated mean is 1 off from the theoretical
(1-p)/p
#[1] 0.2048193
(1-p)/p^2
#[1] 0.2467702
```

#Compute the theoretical probability that the integer k is observed

```
#theoretical probability of k=0 is the same as the simulated probability k=1 b/c R counts failures only
k <- c(0,1,2,3)
prob <- ((1-p)^k)*p
prob
#[1] 0.83000000 0.14110000 0.02398700 0.00407779
```

#Compute the simulated probability that the integer k is observed

```
table(geo.s)/t
#geo.s
#      1          2          3
#0.8364     0.1352     0.0232
 #simulated probability of k=0 is 0
```

#2b

```
t <- 7500
#probability of success
p <- 0.11
#number of Bernoulli trials
n <- 200
#vector for Bernoulli random variables
ber.s <- NULL
#vector for binomial random variables
bin.s <- NULL

#Generate t independent binomial random variables
for(j in 1:t) {
	#Generate n independent Bernoulli trials and store 1 if runif(1) is less than p and 0 otherwise
	for(i in 1:n) {
```

```
        if(runif(1) <= p) {
            ber.s <- c(ber.s,1)
        }
        else {
            ber.s <- c(ber.s,0)
        }
    }
    #Sum the Bernoulli vector (count the number of 1s) and store as one
binomial random variable in bin.s
    bin.s <- c(bin.s,sum(ber.s))
    #Reset the Bernoulli vector
    ber.s <- NULL
}
```

mean(bin.s)

```
#[1] 21.954
```

var(bin.s)

```
#[1] 20.27112
```

#Theoretical mean is n*p and variance is n*p*q

```
n*p
#[1] 22
n*p*(1-p)
#[1] 19.58
```

#Compute the theoretical probability that the integer k is observed

```
k <- c(0,1,2,3)
prob <- choose(n,k)*p^k*(1-p)^(n-k)
prob
#[1] 7.550945e-11    1.866526e-09          2.295407e-08          1.872433e-07
```

#Compute the simulated probability that the integer k is observed

```
table(bin.s)/t
#simulated probability k=0,1,2,3 is 0 for all k; binomial simulation resulted in values
between 8 and 37
```

#2c

```
t <- 7500
lambda <- 2.1
#vector for Poisson random variables
pos.s <- NULL
#sum of exponential random variables
sum.ex <- 0
```

```
#count of Poisson random variables
c <- 0

#Generate t independent Poisson random variables
for(j in 1:t) {
	#Generate independent exponential random variables with parameter lamda
until their sum exceeds 1
	repeat {
		y <- log(runif(1))/-lambda
		sum.ex <- sum.ex + y
		if(sum.ex > 1) {
			break
		}
		else {
			c <- c + 1
		}
	}
	#Sum the n values and store the sum in the vector used to store gamma
values
	pos.s <- c(pos.s,c)
	#Reset the count and sum variables
	c <- 0
	sum.ex <- 0
}
```

mean(pos.s)

```
#[1] 2.090533
```

var(pos.s)

```
#[1] 2.088622
```

#Theoretical mean and variance is lambda, 2.1

#Compute the theoretical probability that the integer k is observed

```
k <- c(0,1,2,3)
prob <- (exp(-lambda)*lambda^k)/factorial(k)
prob
#[1] 0.1224564	0.2571585	0.2700164	0.1890115
```

#Compute the simulated probability that the integer k is observed

```
table(pos.s)/t
#pos.s
#		0			1			2			3
#0.1277333333	0.2578666667	0.2676000000	0.1832000000
```