**STAT 6313**

**Mini Project1**

**Spring 2017**

**Name of the Members**:

Ankita Patil (Net id: asp160730)

Krupali Patel (Net id: kxp160630)

**Contribution of each member:**

We analyzed the problem together and solved the problems.

|  |  |
| --- | --- |
| Question | Contributor |
| Q. 1 (a) | Ankita |
| Q. 1 (b) | Ankita |
| Q. 1(c) | Krupali |
| Q. 1(d) | Krupali |
| Q. 1(e) | Ankita & Krupali |
| Q. 2(a) | Krupali |
| Q. 2(b) | Krupali |
| Q. 2(c) | Ankita |
| Q. 2(d) | Ankita |
| Q. 2(e) | Ankita & Krupali |
| Q. 2(f) | Ankita & Krupali |

**Section 1**

**Q. 1 Suppose a random variable X has the following probability density function: f(x)**

**equals 4\*x^3 when x is between 0 and 1, and equals 0 otherwise.**

**Q. 1(a) Compute E(X), Var(X) and P(X > 0.5) analytically, i.e., using their formulas.**

E(X) =  0∫1x f(x) dx

       =  0∫1x. 4\*x3 dx

       =   0∫1 4\*x4 dx

       =  [4\*x5/5]01

= 4/5 [1-0]

E(X)= 4/5

E(X) =0.8

VAR(X) = E(X2) –[E(X)]2

E(X2) = 0∫1x2 f(x) dx

       =  0∫1x2. 4\*x3 dx

       =   0∫1 4\*x5 dx

       =  [4\*x6/6]01

= 4/6 [1-0]

E(X2)= 2/3

VAR(X) = E(X2) –[E(X)]2

= 2/3 - (4/5)2

VAR(X) = 0.02667

P(X > 0.5) = 1 – P(X <= 0.5)

= 1 – -∞∫0.5 f(x) dx

= 1 – -∞∫0.5 4\*x3  dx

= 1 – [4\*x4/4]-∞0.5

=1 – (0.5)4

P(X > 0.5) = 0.9375

**Q. 1 (b) Explain how you would simulate a draw from the distribution of X.**

To simulate a X

1. Generate U,
2. Set U = F(X)
3. Solve for X

Let’s find CDF

F(X) = P [X <= x]

= 0∫x 4y3 dy 0 < y <x

= [4\*y4/4]0x

= x4

Set U = F (X)

U = X4

Compute X = F-1(U)

X = U1/4

**Q. 1(c) Approximate E(X), Var(X) and P(X > 0.5) using Monte Carlo simulation with 1,000**

**draws 5 times. Summarize the results in a table.**

Summary:

1000 draws 5 times

|  |  |  |  |
| --- | --- | --- | --- |
| # | E(X) | VAR(X) | P(X > 0.5) |
| 1 | 0.8049495 | 0.02487147 | 0.946 |
| 2 | 0.7971737 | 0.02849375 | 0.93 |
| 3 | 0.8012173 | 0.02719301 | 0.932 |
| 4 | 0.807095 | 0.02509989 | 0.941 |
| 5 | 0.7989894 | 0.02801675 | 0.94 |

**Q. 1 (d) Repeat (c) with 10,000 draws.**

Summary :

10,000 draws 5 times

|  |  |  |  |
| --- | --- | --- | --- |
| # | E(X) | VAR(X) | P(X > 0.5) |
| 1 | 0.7999046 | 0.02691711 | 0.9373 |
| 2 | 0.7962628 | 0.0271476 | 0.9347 |
| 3 | 0.8020731 | 0.02577148 | 0.9412 |
| 4 | 0.8036739 | 0.02622246 | 0.9388 |
| 5 | 0.8002165 | 0.02723241 | 0.9331 |

**Q. 1(e) Compare you results in (a), (c) and (d). Explain, with justification, what you observe.**

Results obtained in part (a), (c) and (d) are approximately equal.

But the results obtained in part (d) are more closer to the results obtained analytically. Therefore, we can say that as the sample size increases we get more accurate results.

Hence, Monte Carlo simulation gives us more accurate results as we increase our sample size.

**Q. 2 IQ test scores have a population mean and standard deviation of 100 and 15,**

**respectively. Assume that the scores follow a normal distribution.**

**Q. 2(a) Compute the 95-th percentile of this distribution the usual way.**

By definition X ~ N[mean, sd]

P[ X <= x0.95 ] = 0.95

P[ z <= (x0.95 – mean)/sd]

P[ z<= z0.95 ]

x0.95 =mean + z0.95 \* sd

x0.95 = 124.6728

**Q. 2(b) Suppose your IQ score equals the percentile you computed in (a). What does this mean?**

If IQ score equals 124.6728, it means that 95 % of the values lie below 124.6728.

In other words, we can say that 5% of the IQ score values lie above your IQ score

**Q. 2(d)Approximate the 95-th percentile of the distribution using Monte Carlo simulation**

**with 1,000 draws 5 times.**

Summary

1000 draws 5 times

|  |  |
| --- | --- |
| # | 95th Percentile |
| 1 | 124.0109 |
| 2 | 124.6597 |
| 3 | 125.2573 |
| 4 | 124.2851 |
| 5 | 125.2573 |

**Q. 2(e) Repeat (d) with 10,000 draws.**

Summary

10,000 draws 5 times

|  |  |
| --- | --- |
| # | 95th Percentile |
| 1 | 124.6842 |
| 2 | 124.355 |
| 3 | 124.927 |
| 4 | 124.4744 |
| 5 | 124.7341 |

**Q. 2(f) Compare your results in (a), (d) and (e). Explain, with justification, what you observe.**

Results obtained in part (a) and (d) are approximately equal but a few values are differing by some factor.

On the contrary, results in (a) and (e) are closer and hence, accuracy is more. Therefore, as the number of draws increases we get the value for 95th percentile closer to the value calculated analytically.

Hence, Monte Carlo simulation gives us more accurate results as we increase our sample size.

**Section 2**

**Q. 1(C) Approximate E(X), Var(X) and P(X > 0.5) using Monte Carlo simulation with 1,000**

**draws 5 times. Summarize the results in a table**.

#=======================================================================

#Following function calculates n random draws when it accepts n (1000 in this case) as an argument

#=======================================================================

simulate.x = function(n)

{

u= runif(n)

q= sqrt(sqrt(u))

return(q)

}

#=======================================================================

#Following function will compute Expectance using the draws obtained from the above function #that is function simulate.x , we use in-built R function mean() to calculate the expectance. #An in-built R function var() to calculate variance. The following function also calculates #probability(x>0.5)

#=======================================================================

compute.x= function(y)

{

expectedValue = mean(y)

variance = var(y)

cat("\tExpected Value ",expectedValue)

cat("\t Variance ",variance)

cat("\tProbabilty(x>0.5) ", length(y[y>0.5])/length(y))

cat("\n")

}

#=======================================================================

#Following function repeats 5 times with the help of in-built replicate() to calculate Expectance, variance and probability(X>0.5) using compute.x for 1000 random draws computed by function simulate.x

#=======================================================================

replicate(5,compute.x(simulate.x(1000)))

**Q. 1(d) Repeat (c) with 10,000 draws**

#=======================================================================

#Following function calculates n random draws when it accepts n (10,000 in this case) as an argument

#=======================================================================

simulate.x = function(n)

{

u= runif(n)

q= sqrt(sqrt(u))

return(q)

}

#====================================================================

#Following function will compute Expectance using the draws obtained from the above function

#that is function simulate.x , we use in-built R function mean() to calculate the expectance. #An in-built R function var() to calculate variance. The following function also calculates #probability(x>0.5)

#=======================================================================

compute.x= function(y)

{

expectedValue = mean(y)

variance = var(y)

cat("\tExpected Value ",expectedValue)

cat("\t Variance ",variance)

cat("\tProbabilty(x>0.5) ", length(y[y>0.5])/length(y))

cat("\n")

}

#=======================================================================

##Following function repeats 5 times with the help of in-built replicate() to calculate Expectance, variance and probability(X>0.5) using compute.x for 10,000 random draws computed by function simulate.x

#=======================================================================

replicate(5,compute.x(simulate.x(10000)))

**Q. 2 (c) Explain how you would simulate a draw from the distribution of the IQ scores.**

# simulating a draw from the distribution of IQ scores

#rnorm() generates 1000 numbers from a normal distribution with mean 100 and sd=15

draw.iqScore=rnorm(1000,100,15)

**Q. 2(d) Approximate the 95-th percentile of the distribution using Monte Carlo simulation**

**with 1,000 draws 5 times**.

#===================================================================

#Approximating 95th percentile using Monte Carlo Simulation with 1000 draws, 5 times

#====================================================================

#MCsimulation1 stores a matrix consisting of 1000 rows and 5 columns.

#rnorm() generates 1000 numbers from a normal distribution with mean 100 and sd=15, it is then #replicated 5 times with the help of replicate()

#quantile() generates 95th percentile for the set of 1000 draws.

MCsimulation1 = replicate(5,rnorm(1000,100,15))

cat("draw1.percentile ", quantile(MCsimulation1[,1], c(0.95)))

cat("\ndraw2.percentile ", quantile(MCsimulation1[,2], c(0.95)))

cat("\ndraw3.percentile ", quantile(MCsimulation1[,3], c(0.95)))

cat("\ndraw4.percentile ", quantile(MCsimulation1[,4], c(0.95)))

cat("\ndraw5.percentile ", quantile(MCsimulation1[,5], c(0.95)))

**Q. 2(e)** **Repeat (d) with 10,000 draws.**

#=======================================================================

#Approximating 95th percentile using Monte Carlo Simulation with 10,000 draws, 5 times

#=======================================================================

#MCsimulation2 stores a matrix consisting of 10,000 rows and 5 columns.

MCsimulation2 = replicate(5,rnorm(10000,100,15))

cat("\ndraw1.percentile ", quantile(MCsimulation2[,1], c(0.95)))

cat("\ndraw2.percentile ", quantile(MCsimulation2[,2], c(0.95)))

cat("\ndraw3.percentile ", quantile(MCsimulation2 [,3], c(0.95)))

cat("\ndraw4.percentile ", quantile(MCsimulation2 [,4], c(0.95)))

cat("\ndraw5.percentile ", quantile(MCsimulation2 [,5], c(0.95)))