STT 461 Homework 3 Solutions

Paul Speaker

2023-03-12

1. Define a probability density function as

$$p(x) = cx^3$$

on the interval $0 < X < 3$.

a. Determine what c is.

$$\int_0^3 x^3 dx = 81/4$$

So $c = 4/81$

b. Use the inverse cdf method to simulate 10,000 outcomes of the distribution. Estimate the mean and standard deviation of the distribution.

$$cdf = \frac{1}{81}x^4$$

$$q(x) = \sqrt[4]{81x}$$

```
samp <- (81*runif(10000))^0.25
mean(samp)
```

```
## [1] 2.407197
```

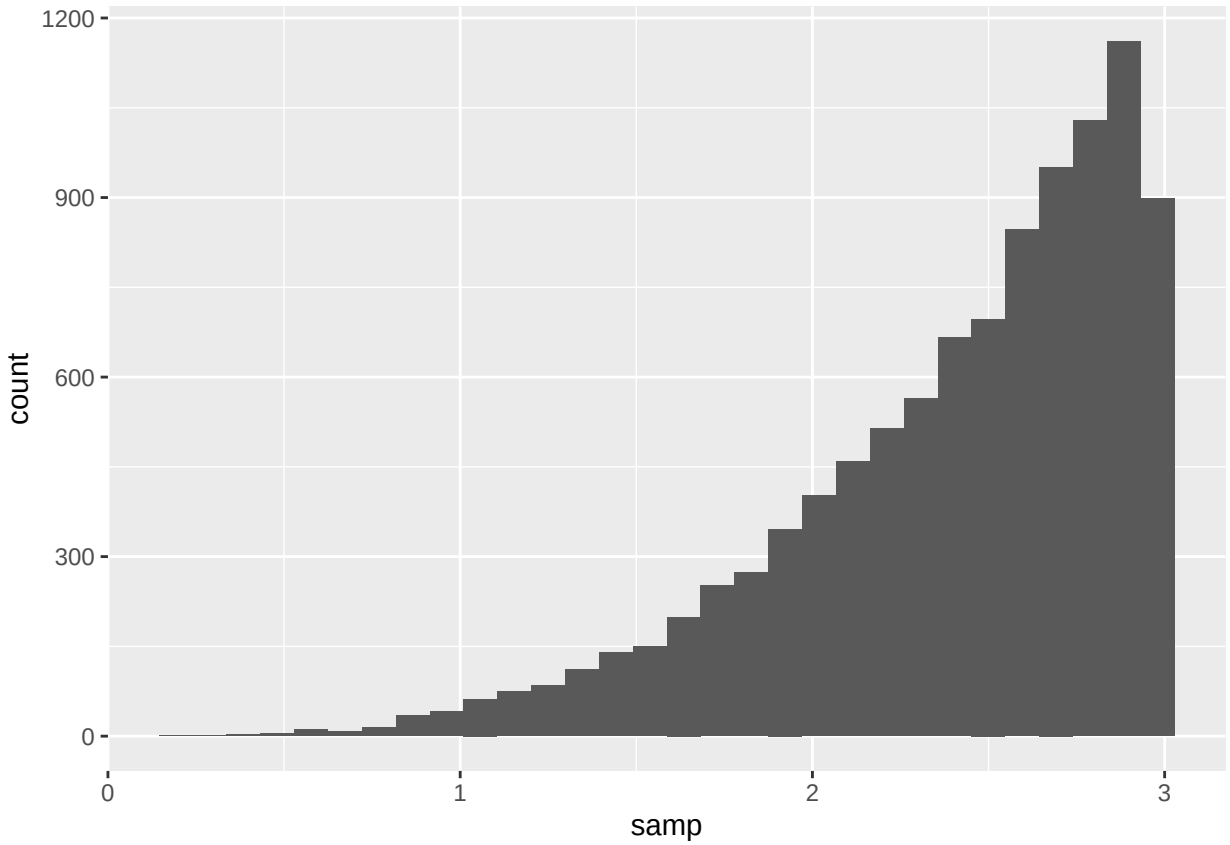
```
sd(samp)
```

```
## [1] 0.4861339
```

c. Plot a histogram of your simulation results. Does it match what you would expect?

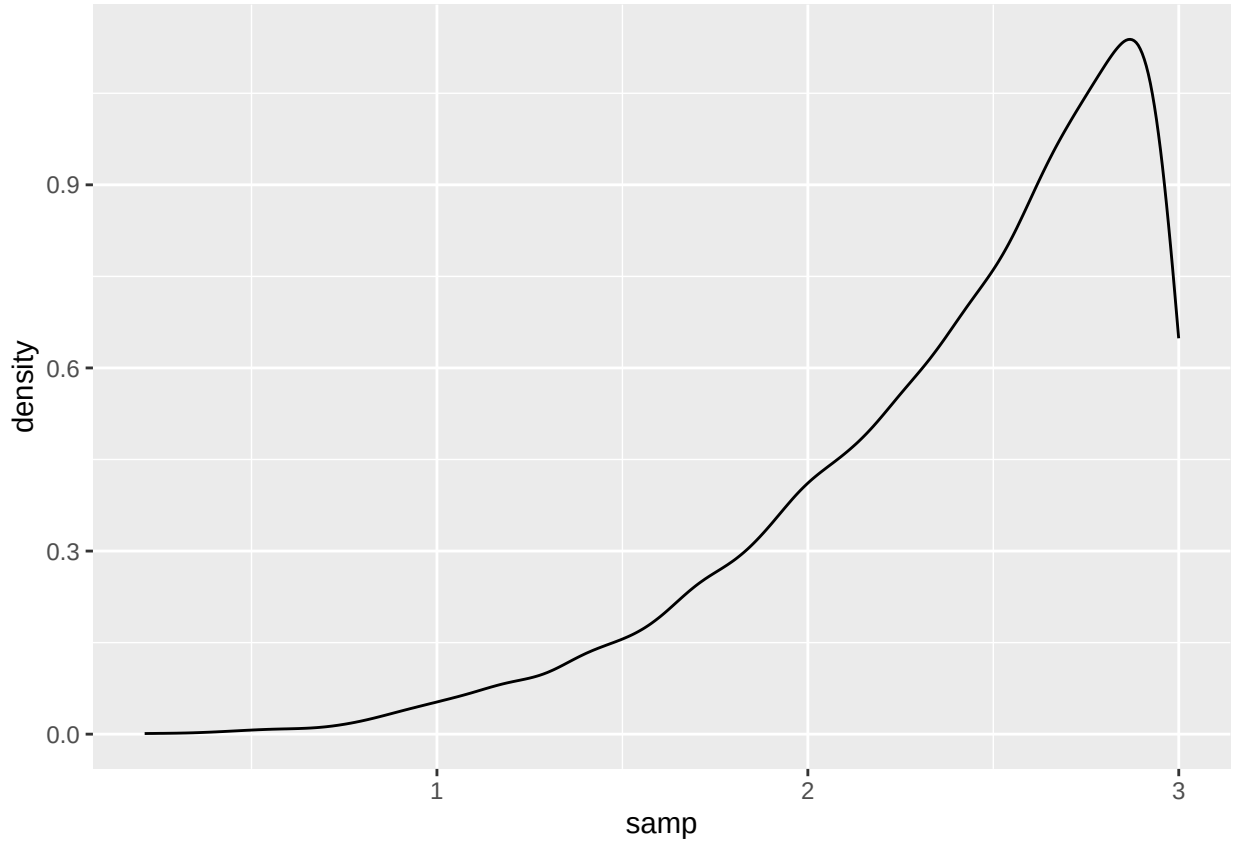
```
library(ggplot2)
sampdf <- data.frame("samp"=samp)
ggplot(data = sampdf, aes(x = samp)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



d. Re-do (c) with geom_density instead of geom_histogram. What is it doing?

```
ggplot(data = sampdf, aes(x = samp)) + geom_density()
```



This is a smoothed histogram. This is also an approximation for kernel density.

e. Numerically approximate the expected value, variance, and standard deviation of the distribution. Calculate the expected value from the definition with an integral and compare how close the 2 are. (This duplicates a previous part, but I'll do it with more samples)

```
samp <- (81*runif(1000000))^0.25
mean(samp)
```

```
## [1] 2.399445
```

```
var(samp)
```

```
## [1] 0.2399237
```

```
sd(samp)
```

```
## [1] 0.48982
```

From the integral

$$\int_0^3 xp(x)dx = \int_0^3 \frac{4}{81}x^4 = \frac{4}{81}\frac{243}{5} = 2.4$$

f. Use a simulation with acceptance/rejection to approximate the expectation of X, given that x is greater than 2.

There are many ways to throw out the smaller values. This is a shortcut without a loop:

```
samp_lim <- ifelse(samp>2,samp, NA)
mean(na.omit(samp_lim))
```

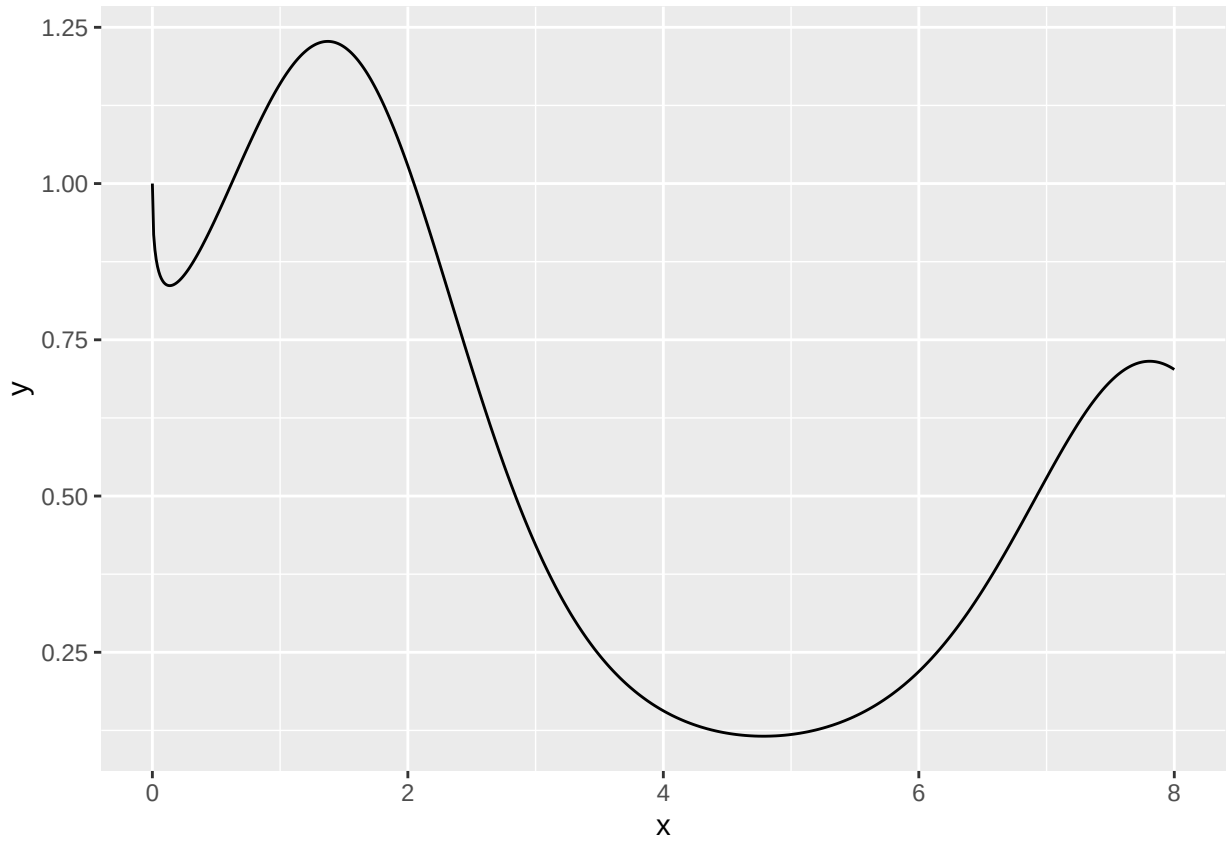
```
## [1] 2.596488
```

2. For the following integral :

$$\int_0^8 \frac{e^{\sin x}}{1+\sqrt{x}} dx$$

a. Generate a plot of the region.

```
x <- seq(0,8, by = 0.01)
y <- exp(sin(x))/(1+sqrt(x))
plot_df <- data.frame("x"=x,"y"=y)
ggplot(data = plot_df, aes(x=x,y=y)) + geom_line()
```



b. Estimate using standard Monte Carlo integration (n = 100,000)

```
x <- runif(100000,0,8)
y <- exp(sin(x))/(1+sqrt(x))
integral1 <- 8*mean(y)
integral1
```

```
## [1] 4.423017
```

c. Estimate using acceptance/rejection Monte Carlo (again, n = 100,000)

```
x <- runif(100000,0,8)
y <- runif(100000,0,1.25)
prop <-  sum(y<exp(sin(x))/(1+sqrt(x)))/100000
integral2 <- prop*8*1.25
integral2
```

```
## [1] 4.3911
```

3. For the following integral:

$$\int_{-\infty}^{\infty} e^{-(x-6)^4}$$

a. Compute the following integral both with the standard Monte Carlo approach (with appropriately truncated bounds) and with Importance Sampling, using an appropriate normal distribution as an importance function. Use n = 10,000 both times.

```
# Standard Monte Carlo
x <- runif(10000,0, 12)
y <- exp(-1*(x-6)^4)
integral1 <- 12*mean(y)
integral1
```

```
## [1] 1.800777
```

```
# Importance Sampling.  Using Gaussian centered at 6 for importance function.
vals <- rnorm(10000,6,1)
integral2 <- mean(exp(-1*(vals-6)^4)/dnorm(vals,6,1))
integral2
```

```
## [1] 1.823348
```

b. Re-do both methods 100 times. Compute the variance of the answers from each method and compare.

```
# Standard Monte Carlo
integral1 <- rep(0,100)
for(i in 1:100){
  x <- runif(10000,0, 12)
  y <- exp(-1*(x-6)^4)
  integral1[i] <- 12*mean(y)
}
mean(integral1)
```

```
## [1] 1.811642
```

```
var(integral1)
```

```
## [1] 0.001216687
```

```
# Importance Sampling.  Using Gaussian centered at 6 for importance function.
integral1 <- rep(0,100)
for(i in 1:100){
  vals <- rnorm(10000,6,1)
  integral2[i] <- mean(exp(-1*(vals-6)^4)/dnorm(vals,6,1))
}
mean(integral2)
```

```
## [1] 1.812771
```

```
var(integral2)
```

```
## [1] 9.815365e-05
```

We can see that the result with importance sampling has about 1/10 the variance, so it is a more accurate method.

4. Suppose we have an exponential probability distribution with parameter lambda = 1/5, so that its expected value is 5. We will perform some simulation experiments to determine the behavior of the sample mean.

a. Generate 2,000 simulations which each constitute 100 outcomes of this distribution. Calculate the mean values for the 2,000 experiments.
b. Next, calculate the mean value and variance for the 2,000 sample means.

```
#for (a) and (b)
sims <- 2000
sampsize <- 100
mn <- replicate(sims,0)
for(i in 1:sims){
mn[i] <- mean(rexp(sampsize, 1/5))
}
mn_100 <- mean(mn)
var_100 <- var(mn)
mn_100
```

```
## [1] 5.003865
```

```
var_100
```

```
## [1] 0.2533212
```

c. Repeat (a) and (b) for N = 1,000, 10,000, 100,000, and 1,000,000 outcomes (2,000 simulations of each). Note: 1,000,000 could take several minutes. N Mean of Sample Means Variance of Sample Means 100
1,000
10,000
100,000
1,000,000

```
sims <- 2000
sampsize <- 1000
mn <- replicate(sims,0)
for(i in 1:sims){
  mn[i] <- mean(rexp(sampsize, 1/5))
}
mn_1000 <- mean(mn)
var_1000 <- var(mn)
sampsize <- 10000
mn <- replicate(sims,0)
for(i in 1:sims){
  mn[i] <- mean(rexp(sampsize, 1/5))
}
mn_10000 <- mean(mn)
var_10000 <- var(mn)

sampsize <- 100000
mn <- replicate(sims,0)
for(i in 1:sims){
  mn[i] <- mean(rexp(sampsize, 1/5))
}

mn_100000 <- mean(mn)
var_100000 <- var(mn)
sampsize <- 1000000
mn <- replicate(sims,0)
for(i in 1:sims){
  mn[i] <- mean(rexp(sampsize, 1/5))
}
```

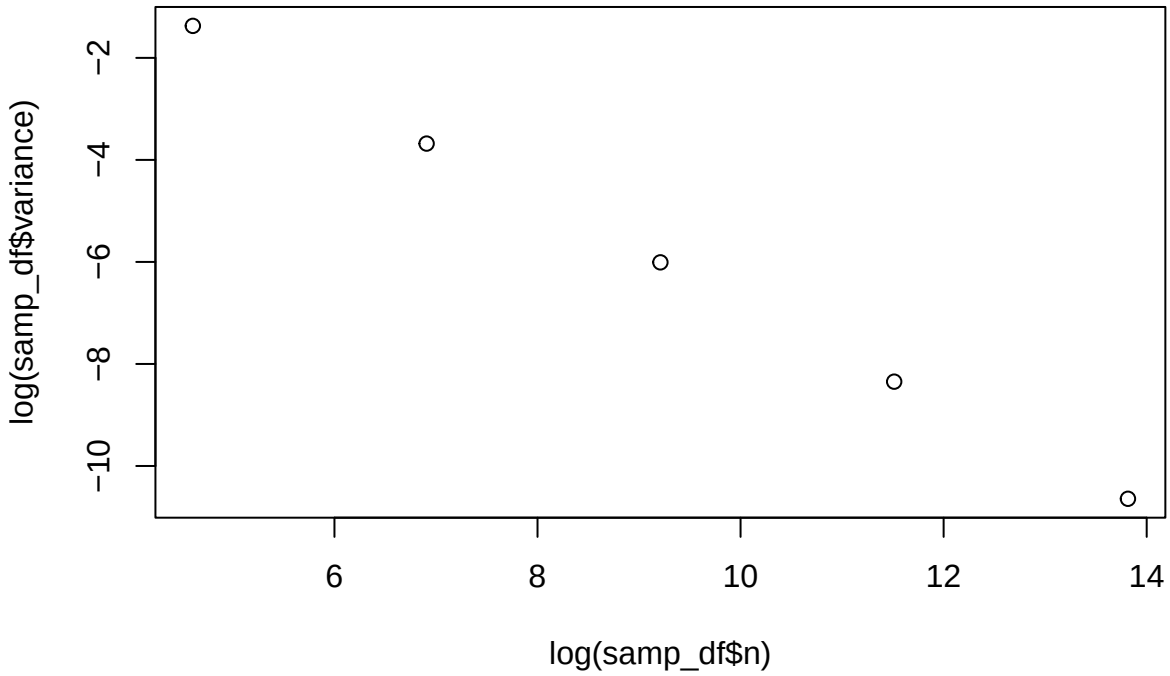
```
mn_1000000 <- mean(mn)
var_1000000 <- var(mn)

meanmean <- c(mn_100, mn_1000, mn_10000, mn_100000, mn_1000000)
varmean <- c(var_100, var_1000, var_10000, var_100000, var_1000000)
meanmean
```

```
## [1] 5.003865 5.001766 4.998906 4.999732 4.999997
```

(d) Plot the log(Variance) vs. log(N). Do you get something which is close to a straight line? What is the slope? Does the relationship between N and the variance close to what you would expect with the Central Limit Theorem?

```
samp_df <- data.frame("n" = 10^(2:6), "variance" = varmean)
plot(log(samp_df$n),log(samp_df$variance))
```




```
summary(lm(data = samp_df, log(variance)~log(n)))
```

```
##
## Call:
## lm(formula = log(variance) ~ log(n), data = samp_df)
##
## Residuals:
##         1         2         3         4         5
## -0.004117  0.009588  0.002006 -0.016310  0.008833
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  3.272257   0.016558   197.6 2.86e-07 ***
## log(n)      -1.007832   0.001695  -594.6 1.05e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01234 on 3 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.536e+05 on 1 and 3 DF,  p-value: 1.049e-08
```

Note: lm is not necessary, just to show relationship Since the slope is -1, the variance experimentally is inversely proportional to n, as we would expect with the Central Limit Theorem.