Solutions to Extra Problems

Hanning Su
Student ID: 855767

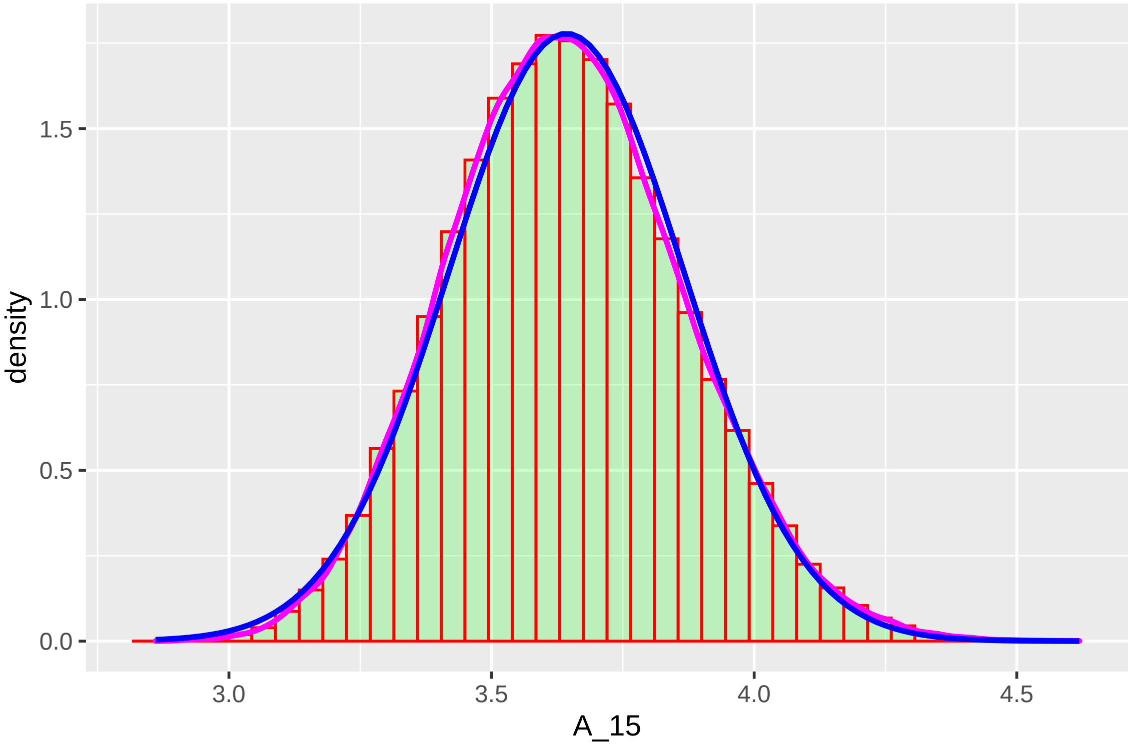
July 6, 2021

Question 1

```
library(ggplot2)

df <- data.frame(n = c(1:100000),
                 i_1 = c(runif(100000, 0.06, 0.12)),
                 i_2 = c(runif(100000, 0.06, 0.12)),
                 i_3 = c(runif(100000, 0.06, 0.12)),
                 i_4 = c(runif(100000, 0.06, 0.12)),
                 i_5 = c(runif(100000, 0.06, 0.12)),
                 i_6 = c(runif(100000, 0.06, 0.12)),
                 i_7 = c(runif(100000, 0.06, 0.12)),
                 i_8 = c(runif(100000, 0.06, 0.12)),
                 i_9 = c(runif(100000, 0.06, 0.12)),
                 i_10 = c(runif(100000, 0.06, 0.12)),
                 i_11 = c(runif(100000, 0.06, 0.12)),
                 i_12 = c(runif(100000, 0.06, 0.12)),
                 i_13 = c(runif(100000, 0.06, 0.12)),
                 i_14 = c(runif(100000, 0.06, 0.12)),
                 i_15 = c(runif(100000, 0.06, 0.12)),
                 A_15 = c(rep(NA, 100000)),
                 LOG_A_15 = c(rep(NA, 100000)))

for (i in 1:100000) {
  df[[17]][i] = (1 + df[[2]][i]) * (1 + df[[3]][i]) * (1 + df[[4]][i]) *
                (1 + df[[5]][i]) * (1 + df[[6]][i]) * (1 + df[[7]][i]) *
                (1 + df[[8]][i]) * (1 + df[[9]][i]) * (1 + df[[10]][i]) *
                (1 + df[[11]][i]) * (1 + df[[12]][i]) * (1 + df[[13]][i]) *
                (1 + df[[14]][i]) * (1 + df[[15]][i]) * (1 + df[[16]][i])

  df[[18]][i] = log((1 + df[[2]][i]) * (1 + df[[3]][i]) * (1 + df[[4]][i]) *
                (1 + df[[5]][i]) * (1 + df[[6]][i]) * (1 + df[[7]][i]) *
                (1 + df[[8]][i]) * (1 + df[[9]][i]) * (1 + df[[10]][i]) *
                (1 + df[[11]][i]) * (1 + df[[12]][i]) * (1 + df[[13]][i]) *
                (1 + df[[14]][i]) * (1 + df[[15]][i]) * (1 + df[[16]][i]))
}

A_15 <- df[[17]]
mean(A_15 > 4) #simulated result

## [1] 0.06036
```

In addition to the results derived from assuming normality for A_{15} and from approximation using CLT as given by Dr. Jin's solution, I add a result derived from simulating a sample of A_{15} of size 100000. The result as shown above is the proportion of simulated A_{15}'s that is greater than 4.

```r
#result from assuming normality for A_15
pnorm(4, mean = 1.09^15, sd = sqrt(1.1884^15 - 1.09^30), lower.tail=FALSE)

## [1] 0.05553055

ggplot(data=df, aes(A_15)) +
  geom_histogram(aes(y =..density..),
                 col="red",
                 fill="green",
                 alpha=.2,
                 bins = 40) +
  geom_density(col=6, size=1) +
  stat_function(col=4, size=1, fun = dnorm, args = list(mean = 1.09^15, sd = sqrt(1.1884^15 - 1.09^30)))
```

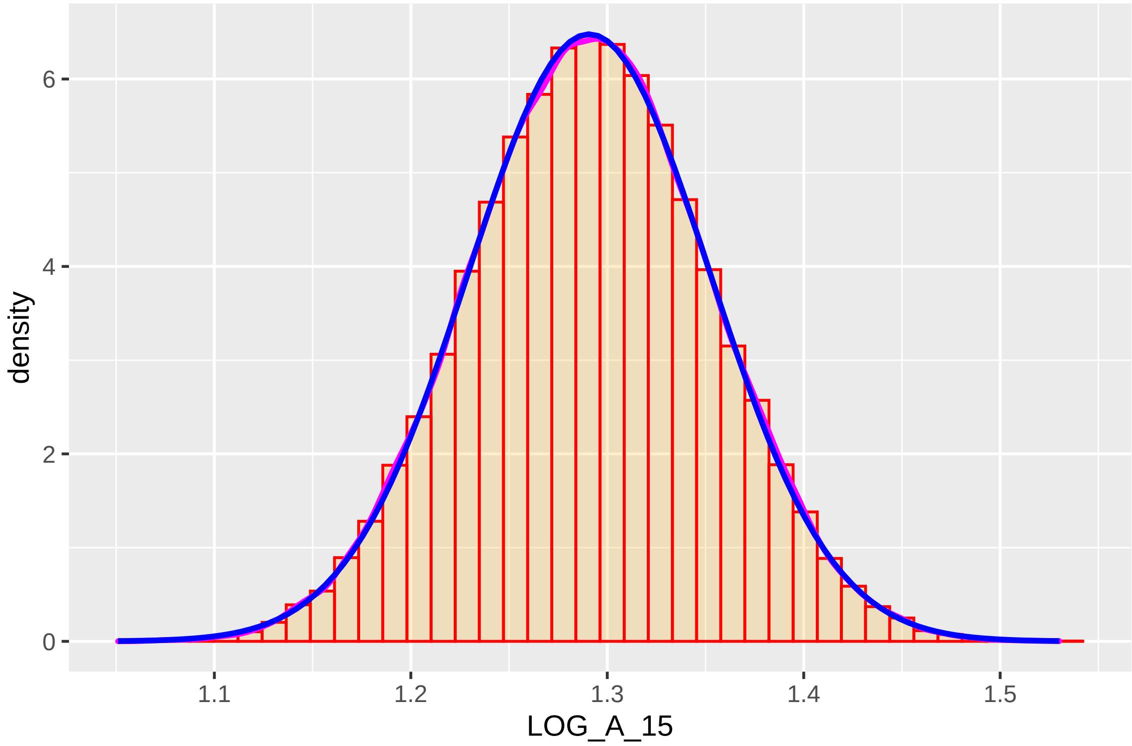


The frequency histogram and the fitted density line (pink) of simulated sample of A_{15} are shown above. The blue line is the theoretical density line of a normal distribution with mean 1.09^{15} and variance $1.1884^{15} - 1.09^{30}$. We can see that the pink line is slightly more right skewed compared with the thretical line. Therefore, assuming normality for A_{15} is a relatively rough choice.

```r
#result from approximating the distribution of log(A_15) using CLT
pnorm(log(4), mean = 15 * 0.08605, sd = sqrt(15 * 0.0002528), lower.tail=FALSE)
```

```
## [1] 0.06038278
```

```r
ggplot(data=df, aes(LOG_A_15)) +
  geom_histogram(aes(y =..density..),
                 col="red",
                 fill="orange",
                 alpha=.2,
                 bins = 40) +
  geom_density(col=6, size = 1) +
  stat_function(col=4, fun = dnorm, args = list(mean = 15 * 0.08605, sd = sqrt(15 * 0.0002528)), size =
```



The frequency histogram and the fitted density line (pink) of simulated sample of $\log(A_{15})$ are shown above. The blue line is the theoretical density line of a normal distribution with mean 15(0.08605) and variance 15(0.0002528). We can see that the pink line fits more closely the thretical line than the one in previous graph does. Therefore, approximating the distribution of $\log(A_15)$ as normal by CLT should be a better choice.

Question 2

```r
integrand1 <- function(x) {(1 + x) / 0.06}
integrand2 <- function(x) {(1 + x)^2 / 0.06}
nu1 <- integrate(integrand1, 0.06, 0.12)$value
nu2 <- integrate(integrand2, 0.06, 0.12)$value

moment_calculator2 <- function (n, r) {
  if (r == 1)
  {
    if (n == 1) {
      return( nu1 )
    } else {
      return (nu1 * (1 + moment_calculator2(n - 1, r)))
    }
  }
  else if (r == 2)
  {
    if (n == 1) {
      return( nu2 )
    } else {
      return (nu2 * (1 + 2 * moment_calculator2(n - 1, 1) + moment_calculator2(n - 1, r)))
    }
  }
  else
  {
    print("higher moment functionality not yet available.")
  }
}

first_moment <- moment_calculator2(15, 1)
first_moment #mean
```

```
## [1] 32.0034
```

```r
second_moment <- moment_calculator2(15, 2)
sd <- sqrt(second_moment - first_moment^2)
sd #standard deviation
```

```
## [1] 1.366168
```

Question 4

(i)

```
# Assume gamma = (1 + i_t) follows log normal distribution mu = 0.08 sigma = 0.04
# gammainv = (1 + i_t)^-1 follows log normal distribution mu = -0.08 sigma = 0.04
s = 0.1
mu = 0.08
sigma = 0.04
integrand1 <- function(x) {x / (x*sigma*sqrt(2*pi)) * exp(-(log(x) - mu)^2/(2*sigma^2))}
integrand2 <- function(x) {x^2 / (x*sigma*sqrt(2*pi)) * exp(-(log(x) - mu)^2/(2*sigma^2))}

nu_1 <- integrate(integrand1, 0, Inf)$value
nu_2 <- integrate(integrand2, 0, Inf)$value

moment_calculator3 <- function (n, r) {
  if (r == 1)
  {
    if (n == 1) {
      return( nu_1 )
    } else {
      return ((moment_calculator3(n - 1, r) + (1 + s)^(n - 1)) * nu_1)
    }
  }
  else if (r == 2)
  {
    if (n == 1) {
      return( nu_2 )
    } else {
      return (((1 + s)^(2 * n - 2) + 2 * (1 + s)^(n - 1) * moment_calculator3(n - 1, 1)
              + moment_calculator3(n - 1, r)) * nu_2)
    }
  }
  else
  {
    print("higher moment functionality not yet available.")
  }
}

df1 <- data.frame(n = c(1:20),
                  First_Moment = c(rep(NA, 20)),
                  Second_Moment = c(rep(NA, 20)),
                  Variance = c(rep(NA, 20)))

for (r in 2:4) {
  for (n in 1:20){
    if (r != 4) {
      df1[[r]][n] = moment_calculator3(n, r - 1)
    } else {
     df1[[r]][n] = round(df1[[r - 1]][n] - (df1[[r - 2]][n])^2, digits = 4)
    }
  }
}
```

I implement here the recursion formula as derived in the solution of question 4. I assume here that $s = 0.1$ and that $1 + i_t$ follows a log normal distribution with $\mu = 0.08$ and $= 0.04$.

Table 1:

	n	First_Moment	Second_Moment	Variance
1	1	1.084	1.177	0.002
2	2	2.368	5.618	0.011
3	3	3.879	15.084	0.037
4	4	5.649	32.001	0.095
5	5	7.711	59.669	0.207
6	6	10.106	102.541	0.407
7	7	12.877	166.568	0.745
8	8	16.074	259.652	1.291
9	9	19.750	392.217	2.144
10	10	23.969	577.943	3.444
11	11	28.798	834.696	5.383
12	12	34.314	1,185.705	8.223
13	13	40.605	1,661.062	12.320
14	14	47.765	2,299.609	18.157
15	15	55.901	3,151.324	26.380
16	16	65.134	4,280.326	37.850
17	17	75.597	5,768.657	53.711
18	18	87.439	7,721.036	75.475
19	19	100.825	10,270.820	105.133
20	20	115.940	13,587.490	145.295

(ii)

```
base_case1 <- 1
base_case2 <- 1
s <- 0.1

mu = -0.08
sigma = 0.04
integrand1 <- function(x) {x / (x*sigma*sqrt(2*pi)) * exp(-(log(x) - mu)^2/(2*sigma^2))}
integrand2 <- function(x) {x^2 / (x*sigma*sqrt(2*pi)) * exp(-(log(x) - mu)^2/(2*sigma^2))}

mu_1 <- integrate(integrand1, 0, Inf)$value
mu_2 <- integrate(integrand2, 0, Inf)$value

moment_calculator4 <- function (n, r) {
  if (r == 1)
  {
    if (n == 1) {
      return( base_case1 )
    } else {
      return (1 + (1 + s) * mu_1 * moment_calculator4(n - 1, r))
    }
  }
  else if (r == 2)
  {
    if (n == 1) {
      return( base_case2 )
    } else {
      return (1 + 2 * mu_1 * (1 + s) * moment_calculator4(n - 1, 1) +
                mu_2 * (1 + s)^2 * moment_calculator4(n - 1, r))
    }
  }
  else
  {
    print("higher moment functionality not yet available.")
  }
}

df2 <- data.frame(n = c(1:20),
                  First_Moment = c(rep(NA, 20)),
                  Second_Moment = c(rep(NA, 20)),
                  Variance = c(rep(NA, 20)))


for (r in 2:4) {
  for (n in 1:20){
    if (r != 4) {
      df2[[r]][n] = moment_calculator4(n, r - 1)
    } else {
      df2[[r]][n] = round(df2[[r - 1]][n] - (df2[[r - 2]][n])^2, digits = 4)
    }
  }
}
```

I implement here the recursion formula as derived in the solution of question 4. I assume here that $s = 0.1$ and that $1 + i_t$ follows a log normal distribution with $\mu = 0.08$ and $= 0.04$. I used the result from lecture that if $X \sim LN(\mu, \sigma), we have X^{-1} \sim LN(-\mu, \sigma)$

Table 2:

	n	First_Moment	Second_Moment	Variance
1	1	1	1	0
2	2	2.016	4.067	0.002
3	3	3.049	9.305	0.008
4	4	4.099	16.822	0.024
5	5	5.165	26.731	0.053
6	6	6.249	39.148	0.099
7	7	7.350	54.196	0.167
8	8	8.470	71.999	0.262
9	9	9.607	92.691	0.389
10	10	10.763	116.406	0.555
11	11	11.938	143.287	0.766
12	12	13.132	173.480	1.028
13	13	14.345	207.138	1.349
14	14	15.578	244.420	1.735
15	15	16.831	285.490	2.196
16	16	18.105	330.520	2.740
17	17	19.399	379.687	3.377
18	18	20.714	433.176	4.115
19	19	22.050	491.177	4.966
20	20	23.408	553.889	5.941