Homework 01-25-2021

Jeffrey Liang ZL2974

Problem 1

Develop two Monte Carlo methods for the estimation of $\theta = \int_0^1 e^{x^2} dx$ and implement in **R** .

Answer: your answer starts here...

Method 1

Introduce Let $Y \sim U(0,1)$

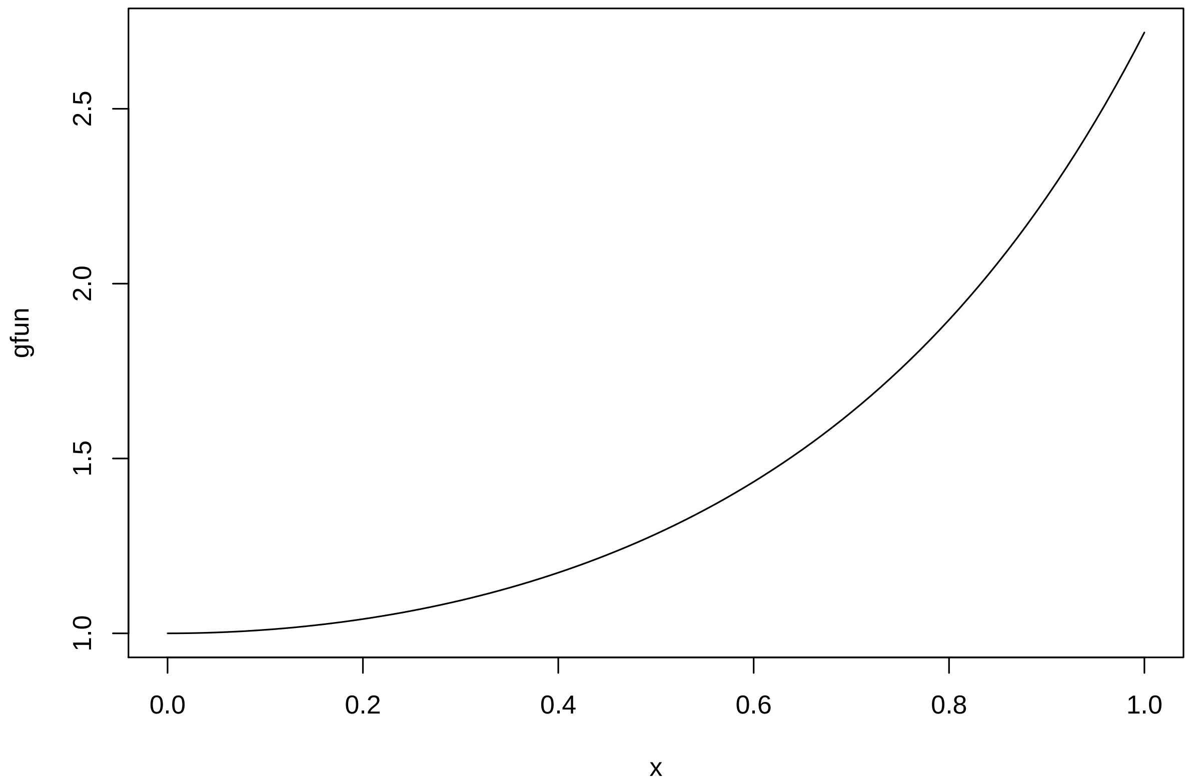
$$\theta = \int_0^1 e^{y^2} * 1dx = E[e^{y^2}]$$

```
set.seed(123123)
y = runif(1000, 0, 1)

gfun = function(y) {
  return((exp(y ^ 2)))
}

gx =
  gfun(y)

theta_estimate =
  list(theta = mean(gx),
       var_theta = var(gx))

integrate(function(x)
  exp(x ^ 2), 0, 1)
```

```
## 1.46 with absolute error < 1.6e-14
```

Method 2

```
plot.function(gfun,0,1)
```



looking at the function, which is similar to a exponetial funciton, thus introducing $f(x) = e^x$ as control variate, we have:

$$\theta = \int_0^1 \beta * e^x + (e^{x^2} - \beta * e^x)dx = \beta * e^x|_0^1 + E[(e^{x^2} - \beta * e^x)]$$

```r
ffun = function(y){
  return(exp(y))
}

fx = ffun(y)

b = lm(gx~fx)$coef[2]

theta_estimate_2 =
  list(theta = mean(b*(exp(1)-1)+(gx-b*fx)),
       theta2 = sum(lm(gx~fx)$coef*c(1,mean(fx))),
       var_theta =var((b*(exp(1)-1)+(gx-b*fx)))
       )

theta_estimate_2
```

```
## $theta
```

```
## [1] 1.46
##
## $theta2
## [1] 1.45
##
## $var_theta
## [1] 0.0124
```

Problem 2

Show that in estimating $\theta = E\{\sqrt{1-U^2}\}$ it is better to use U^2 rather than U as the control variate, where $U \sim U(0,1)$. To do this, use simulation to approximate the necessary covariances. In addition, implement your algorithms in **R**.

Answer: your answer starts here...

Objective:

We are interest in estimating the performance of U^2 and U as a control variate to estimate $\theta = E\{\sqrt{1-U^2}\}$, which $U \sim U(0,1)$,

Data Genearation Mechanisms :

We generate data from $U \sim U(0,1)$ with sample size(n) from 10^1 to 10^5

Estimate and method

$\theta = E\{\sqrt{1-U^2}\}$ will be estimate by:

- $\theta_0 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{1-U^2}$
- $\theta_1 = \int \beta_1 u f_1(u)du + \frac{1}{n}\sum_{i=1}^{n}(\sqrt{1-U^2} - \beta_1 U)$
- $\theta_1 = \int \beta_2 u^2 f_2(u)du + \frac{1}{n}\sum_{i=1}^{n}(\sqrt{1-U^2} - \beta_2 U^2)$

where β is estimate by $\frac{-cov(\sqrt{1-U^2}, g_i(U))}{var(g_i(U))}$

each estimate is simulated 1000 times

Performance

Variance of the estimate and the mean sum of square of θ_i and θ_0 will be estimate as well as their efficiency $\frac{var(theta_0) - cov(theta_0, theta_i)}{var(theta_0)}$

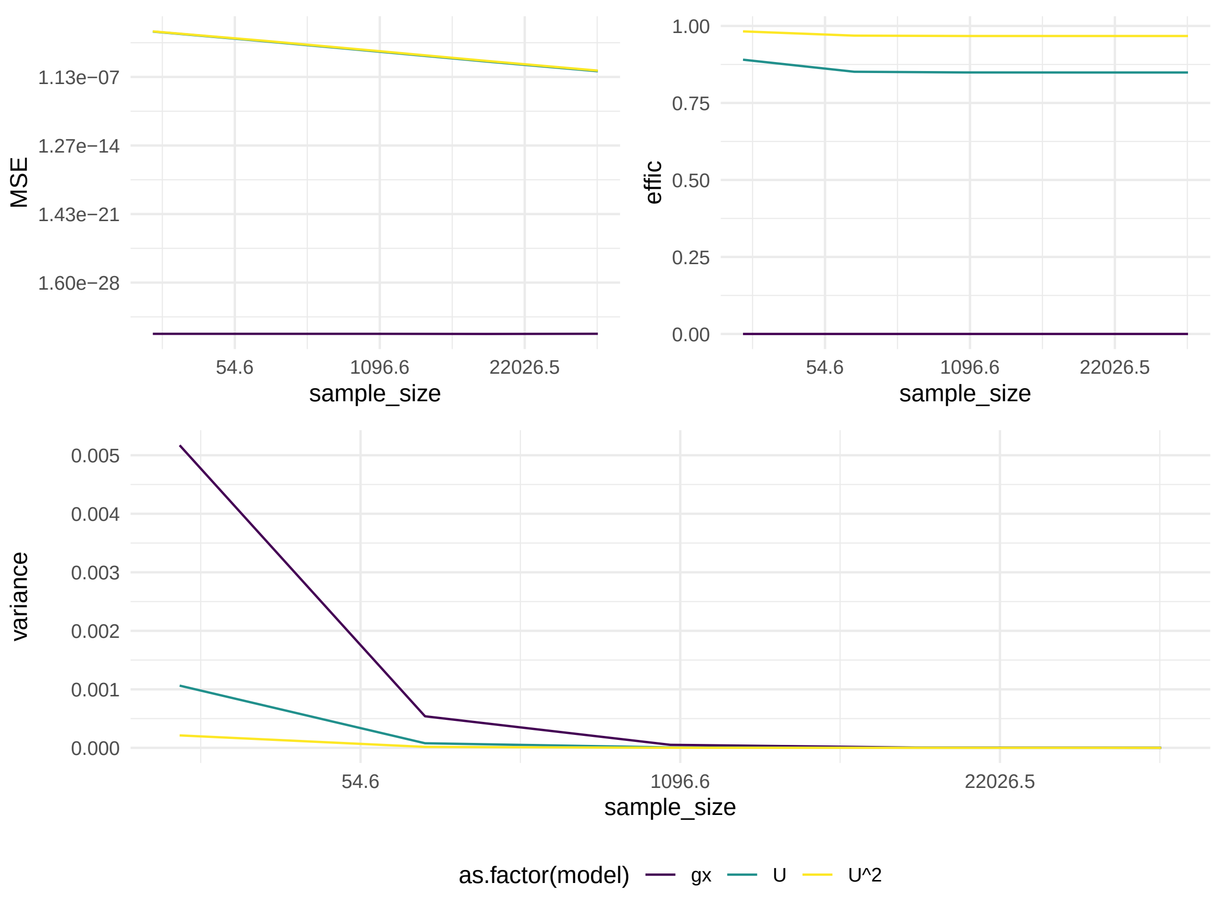
```r
gfun = function(x)
  return(sqrt(1 - x ^ 2))

f1fun = function(x)
  return(x ^ 2)

f2fun = function(x)
  return(x)

simulating =
  function(sample_size = 100,simulation = 1000){
    set.seed(123123)

    result = tibble()

    for (i in 1:simulation) {
      y = runif(sample_size, 0, 1)

      gx = gfun(y)

      f1x = f1fun(y)

      b1 = lm(gx ~ f1x)$coef[2]

      f2x = f2fun(y)

      b2 = lm(gx ~ f2x)$coef[2]

      theta_1 =
        b1 * (1 / 3) + gx - b1 * f1x

      theta_2 =
        b2 * (1 / 2) + gx - b2 * f2x

      result = result %>%
        rbind(
          tibble(
            sample_size = rep(sample_size, 3),
            model = c("gx", "U^2", "U"),
            theta = c(mean(gx), mean(theta_1), mean(theta_2)),
            variance = c(var(gx), var(theta_1), var(theta_2)),
            MSE = list(gx, theta_1, theta_2) %>% lapply(function(x)
              mean(x - mean(gx)) ^ 2) %>% as.numeric(),
            effic = c(0, (var(gx) - var(theta_1)), (var(gx) - var(theta_2))) / var(gx)
          )
        )
    }

    result = result %>%
      group_by(model,sample_size) %>%
      summarise(
            variance = var(theta),
            theta = mean(theta),
```

```r
            MSE = mean(MSE),
            effic = mean(effic)
      ) %>%
      ungroup()

    return(result)

  }

result = tibble()

for (i in seq(1,5,1)){
  n = 10^i

  result = result %>%
    rbind(
      simulating(n,1000)
    )
}

knitr::kable(result,digits = 6)
```

model	sample_size	variance	theta	MSE	effic
gx	1e+01	0.005171	0.786	0.000000	0.000
U	1e+01	0.001063	0.794	0.004481	0.890
U^2	1e+01	0.000213	0.789	0.004687	0.982
gx	1e+02	0.000540	0.786	0.000000	0.000
U	1e+02	0.000079	0.787	0.000471	0.851
U^2	1e+02	0.000017	0.786	0.000525	0.969
gx	1e+03	0.000052	0.786	0.000000	0.000
U	1e+03	0.000007	0.786	0.000044	0.849
U^2	1e+03	0.000001	0.785	0.000050	0.967
gx	1e+04	0.000005	0.785	0.000000	0.000
U	1e+04	0.000001	0.785	0.000004	0.849
U^2	1e+04	0.000000	0.785	0.000005	0.967
gx	1e+05	0.000001	0.785	0.000000	0.000
U	1e+05	0.000000	0.785	0.000000	0.849
U^2	1e+05	0.000000	0.785	0.000000	0.967

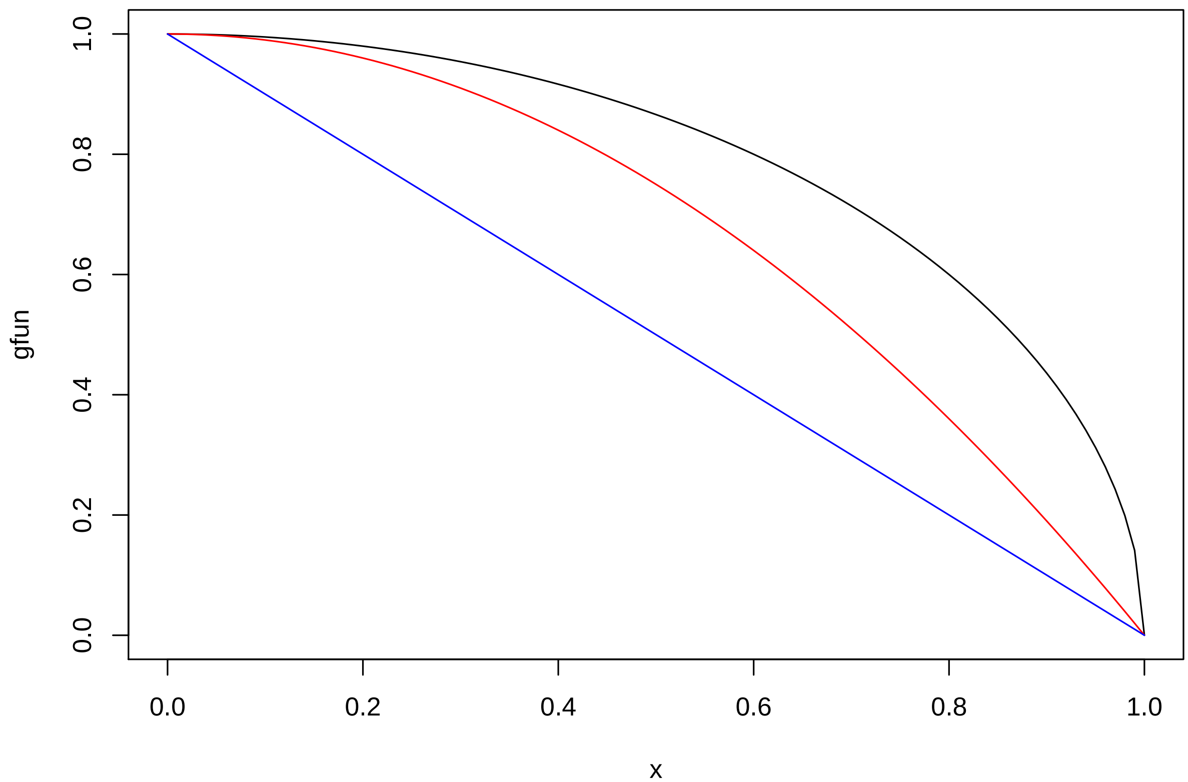
```r
p1 = result %>%
  ggplot(aes(
    x = sample_size,
    y = MSE,
    color = as.factor(model),
    group = as.factor(model)
  )) +
  geom_path()+
  scale_x_continuous(trans = "log")+
  scale_y_continuous(trans = "log")

p2 = result %>%
```

```r
  ggplot(aes(
    x = sample_size,
    y = effic,
    color = as.factor(model),
    group = as.factor(model)
  )) +
  geom_path()+
  scale_x_continuous(trans = "log")

p3 = result %>%
  ggplot(aes(
    x = sample_size,
    y = variance,
    color = as.factor(model),
    group = as.factor(model)
  )) +
  geom_path()+
  scale_x_continuous(trans = "log")

(p1+p2)/p3 + plot_layout(guides = "collect")
```



The mean square error of U^2 is similar to U, but the result shows that U^2 has higher efficiency than U and a lower variance of simulation estimation, the plot shows that, $1 - U^2$ is more resemble to the original function.

```
plot.function(gfun,0,1)
plot.function(function(x) 1-f1fun(x),0,1,add=T,col = "red")
plot.function(function(x) 1-f2fun(x),0,1,add=T,col = "blue")
```



Problem 3

Obtain a Monte Carlo estimate of

$$\int_1^\infty \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

by importance sampling and evaluate its variance. Write a **R** *function to implement your procedure.*

Answer: your answer starts here...

From wolfarm/alpha we know that the integral integrate to 0.40.

Method 1

$$\theta = \int \frac{\frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} * (I > 1) + 0 * (I <= 0)}{Gamma(1, 1/2)} * Gamma(1, 1/2) dx \tag{1}$$

```r
set.seed(123123)

result_1 = tibble()

for (i in 1:1000) {
  y = rgamma(1000, 1, 1 / 2)

  gfun = function(x)
    return(x ^ 2 / sqrt(2 * pi) * exp(-x ^ 2 / 2) * (x > 1))

  gx = gfun(y)

  y = dgamma(y, 1, 1 / 2)

  theta = mean(gx / y)

  result_1 = result_1 %>%
    rbind(tibble(
      theta = theta
    ))
}

result_1 %>%
  summarise(variance = var(theta),
            theta = mean(theta)) %>%
  knitr::kable(digits = 6)
```

variance	theta
0.000243	0.401

Method 2

We first us change of variables, where we take $x = tan(y)$, s.t

$$\begin{aligned}
\theta &= \int_1^\infty \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= \int_{\pi/4}^{\pi/2} \frac{tan(y)^2}{\sqrt{2\pi}} * e^{-\frac{tan(y)^2}{2}} \frac{1}{cos(y)^2} dy \\
&= \int_1^\infty \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= \int_{\frac{\pi/4^{\pi/2} \frac{tan(y)^2}{\sqrt{2\pi}} * e^{-\frac{tan(y)^2}{2}} \frac{1}{cos(y)^2}}{\frac{1}{(\pi/2-\pi/4)}} * \frac{1}{(\pi/2-\pi/4)}} dy
\end{aligned}$$

```r
set.seed(123123)

result_2 = tibble()

for (i in 1:1000){

  y = runif(1000, pi / 4, pi / 2)

  yfun =
    function(x) {
      return((tan(x) ^ 2 * exp(-tan(x) ^ 2 / 2)) / (sqrt(2 * pi) * cos(x) ^ 2))
    }

  gx = yfun(y)

  yx = 1 / (pi / 2 - pi / 4)

  theta = mean(gx / yx)

  theta

  result_2 = result_2 %>%
    rbind(tibble(theta = theta))
}

result_2 %>%
  summarise(variance = var(theta),
            theta = mean(theta)) %>%
  knitr::kable(digits = 6)
```

variance	theta
0.000112	0.401

Method 3

$$\int_1^\infty x^2 * \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

,

if we use $g(x) = x^2 * I(x > 1) + 0 * I(x \leq 0)$ and introduce standard normal as $p(x)$, the function becomes:

$$\int_1^\infty \frac{x^2 * \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = E[x^2 * I(x > 1)]$$

,

```r
set.seed(123123)

result_3 = tibble()


for (i in 1:1000) {
```

```
  y = rnorm(1000)

  gfun = function(x)
    return(x ^ 2 * (x > 1))

  gx = gfun(y)

  theta = mean(gx)

  theta

  result_3 = result_3 %>%
    rbind(tibble(theta = theta))
}

result_3 %>%
  summarise(variance = var(theta),
            theta = mean(theta)) %>%
  knitr::kable(digits = 6)
```

variance	theta
0.00125	0.4