Final 1

Erick Martinez - 216087033

2023-04-15

For the first simulation study, I am going to apply and compare several Monte Carlo integration methods for the integral $\int_0^1 \frac{lnx}{1+x} = -\frac{\pi^2}{12} \approx -0.8224670$ with support on $[0, 1]$.

I will also find the computing time for each method to compare accuracy with efficiency.

Regular Monte Carlo

$f(x) = 1$

```r
# Set parameters and function
n = 2000
h = function(x) {log(x) / (1 + x)}

#  Monte Carlo function
mc_simple = function(n) {

  X = runif(n, min=0, max=1)      ## RV Gen
  hh = h(X)

  I_hat = mean(hh)                ## Approximation
  err = var(hh)/n                 ## Error

  return(list(Ihat=I_hat, se=err))
}

# Output results
results_simple = mc_simple(n)

results_simple$Ihat ; results_simple$se
```

```
## [1] -0.843584
```

```
## [1] 0.0004947842
```

```r
# 95% Confidence interval
alpha = 0.05

z = qnorm(p=alpha/2, lower.tail = FALSE) ### Finding z_{alpha/2}
```

```r
### Calculating the Confidence Interval width
CI = c(results_simple$Ihat - z * sqrt(results_simple$se), results_simple$Ihat + z * sqrt(results_simple$

CI
```

```
## [1] -0.8871810 -0.7999871
```

Importance sampling

We are going to compare two different densities, this densities should be close to being proportional to $|h(x)|$:

a) $g(x) = \frac{1}{4x}$

```r
# New parameters and functions
n = 2000
h_sampling1 = function(x) { 4 * x * log(x) / (1 + x) }
g1 = function(x) {1 / (4*x)}

# MC sampling function
mc_sampling1 = function(n) {

  X = exp(4 * runif(n, min=0, max=1))
  hh = h_sampling1(X)

  I_hat = mean(hh)                ## Approximation
  err = var(hh)/n                 ## Error

  return(list(Ihat=I_hat, se=err))
}

# Output results
results_sampling1 = mc_sampling1(n)

results_sampling1$Ihat ; results_sampling1$se
```

```
## [1] 7.297926
```

```
## [1] 0.01114837
```

b) $g(x) = 2x$

```r
# New parameters and functions
n = 2000
h_sampling2 = function(x) {log(x) / ((1 + x) * (2*x))}
g2 = function(x) {2*x}

# MC sampling function
mc_sampling2 = function(n) {

  X = sqrt(runif(n, min=0, max=1))
  hh = h_sampling2(X)
```

```r
  I_hat = mean(hh)                  ## Approximation
  err = var(hh)/n                   ## Error

  return(list(Ihat=I_hat, se=err))
}

# Output results
results_sampling2 = mc_sampling2(n)

results_sampling2$Ihat ; results_sampling2$se
```

```
## [1] -0.7457242
```

```
## [1] 0.005277583
```

c) $g(x) = -2x + 2$

```r
# New parameters and functions
n = 2000
h_sampling3 = function(x) {log(x) / ((1 + x) * (-2*x + 2))}
g3 = function(x) {-2*x + 2}
# MC sampling function
mc_sampling3 = function(n) {

  X = 1 - sqrt(1 - runif(n, min=0, max=1))
  hh = h_sampling3(X)

  I_hat = mean(hh)                  ## Approximation
  err = var(hh)/n                   ## Error

  return(list(Ihat=I_hat, se=err))
}

# Output results
results_sampling3 = mc_sampling3(n)

results_sampling3$Ihat ; results_sampling3$se
```

```
## [1] -0.8165399
```

```
## [1] 0.0001276007
```

d) $g(x) = -3x + 2$

```r
# New parameters and functions
n = 2000
h_sampling4 = function(x) {log(x) / ((1 + x) * (-3*x + 2))}
g4 = function(x) {-3*x + 2}
# MC sampling function
mc_sampling4 = function(n) {
```

```r
  X = (2/3) - sqrt((4/9) - (2 * runif(n, min=0, max=1) / 3))
  hh = h_sampling4(X)

  I_hat = mean(hh)                  ## Approximation
  err = var(hh)/n                   ## Error

  return(list(Ihat=I_hat, se=err))
}

# Output results
results_sampling4 = mc_sampling4(n)
```

```
## Warning in sqrt((4/9) - (2 * runif(n, min = 0, max = 1)/3)): NaNs produced
```

```r
results_sampling4$Ihat ; results_sampling4$se
```

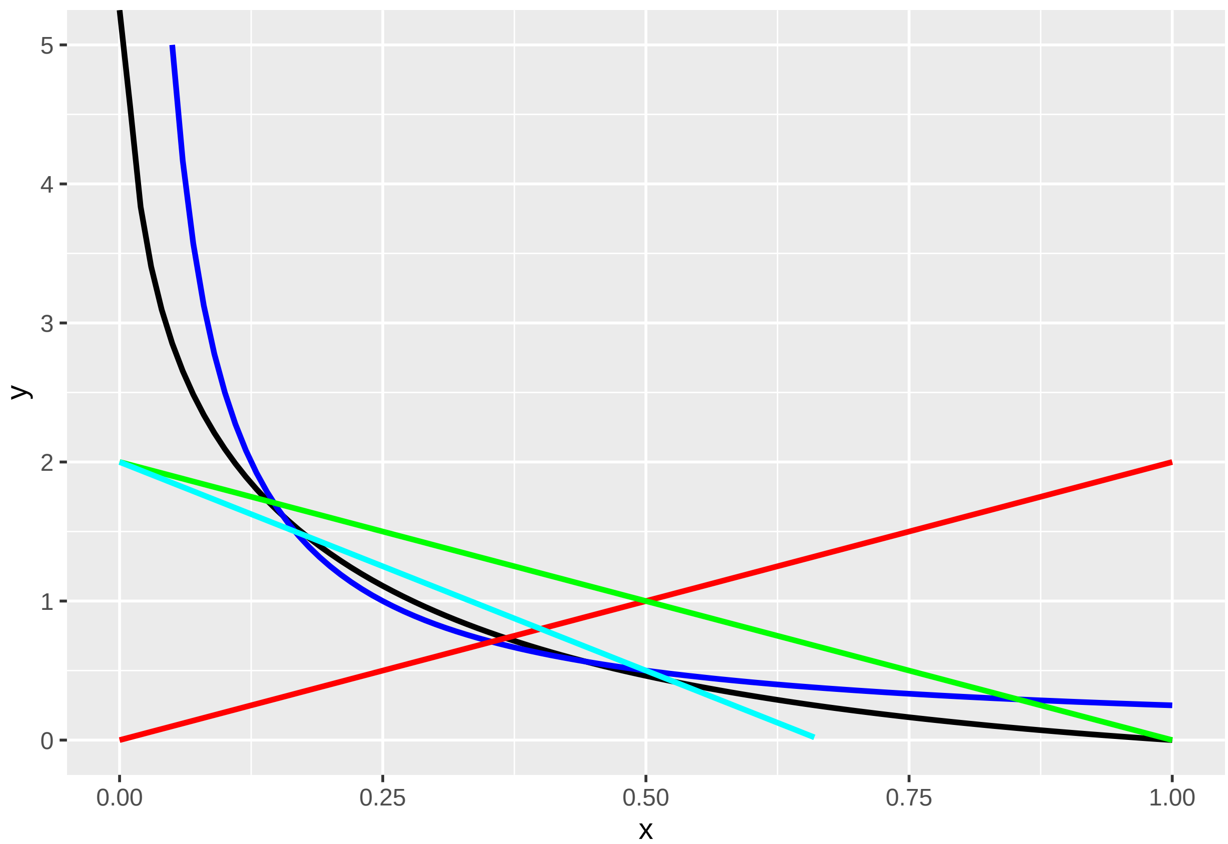
```
## [1] NaN
```

```
## [1] NA
```

Density plot

```r
# Define absolute of h
h_abs = function(x) {abs(log(x) / (1 + x))}

# Plot
ggplot(data.frame(x = c(0, 1)), aes(x=x), col=group) +
  stat_function(fun=h_abs, color="black", linewidth=1) +
  stat_function(fun=g1, color="blue", linewidth=1) +
  stat_function(fun=g2, color="red", linewidth=1) +
  stat_function(fun=g3, color="green", linewidth=1) +
  stat_function(fun=g4, color="cyan", linewidth=1) +
  xlim(0,1) + ylim(0,5)
```

```
## Warning: Removed 34 rows containing missing values (`geom_function()`).
```



Stratified sampling

a) $f(x) = 1$

```
# Create functions and parameters
h = function(x) {log(x) / (1 + x)}

# Create a generalized stratified sampling function for m subsets
mc_stratified = function(n, m) {
  a = 1 / m
  nn = n / m

  ## Initial loop parameters
  T_hat_vec = rep(0,m)
  a_loop = 0

  ## Strata I_hat computation loop
  for (i in 1:m){
    X = runif(nn, min=a_loop, max=(a_loop + a))
    T_hat_vec[i] = mean(h(X))
    a_loop = a_loop + a
  }

  ## Find I hat and its error
```

```r
  I_hat = sum(a * T_hat_vec)

  err = sum((a^2 * var(T_hat_vec) / n))

  ## Output
  return(list(Ihat=I_hat, se=err))
}

# Call the function
## m = 8
print("Stratified sampling with f=1 & m=8:")
```

```
## [1] "Stratified sampling with f=1 & m=8:"
```

```r
results_strat8 = mc_stratified(2000, 8)

results_strat8$Iha ; results_strat8$se
```

```
## [1] -0.8232213

## [1] 7.357698e-06
```

```r
## m = 16
print("Stratified sampling with f=1 & m=16:")
```

```
## [1] "Stratified sampling with f=1 & m=16:"
```

```r
results_strat16 = mc_stratified(2000, 16)

results_strat16$Iha ; results_strat16$se
```

```
## [1] -0.8337822

## [1] 1.995921e-06
```

```r
## m = 32
print("Stratified sampling with f=1 & m=32:")
```

```
## [1] "Stratified sampling with f=1 & m=32:"
```

```r
results_strat32 = mc_stratified(2000, 32)

results_strat32$Iha ; results_strat32$se
```

```
## [1] -0.8196881

## [1] 4.554351e-07
```

b) $g_3(x) = -2x + 2$

```r
# Create a generalized stratified sampling function for m subsets
mc_stratified_g3 = function(n, m) {
  a = 1 / m
  nn = n / m

  ## Initial loop parameters
  T_hat_vec = rep(0,m)
  a_loop = 0

  ## Strata I_hat computation loop
  for (i in 1:m){
    X = 1 - sqrt(1 - runif(nn, min=a_loop, max=(a_loop + a)))
    T_hat_vec[i] = mean(h_sampling3(X))

    a_loop = a_loop + a
  }

  ## Find I hat and its error
  I_hat = sum(a * T_hat_vec)

  err = sum((a^2 * var(T_hat_vec) / n))

  ## Output
  return(list(Ihat=I_hat, se=err))
}

# Call the function
## m = 8
print("Stratified sampling with g3 & m=8:")
```

```
## [1] "Stratified sampling with g3 & m=8:"
```

```r
results_strat_g3_8 = mc_stratified_g3(2000, 8)

results_strat_g3_8$Ihat ; results_strat_g3_8$se
```

```
## [1] -0.8142154
```

```
## [1] 1.859456e-06
```

```r
## m = 16
print("Stratified sampling with g3 & m=16:")
```

```
## [1] "Stratified sampling with g3 & m=16:"
```

```r
results_strat_g3_16 = mc_stratified_g3(2000, 16)

results_strat_g3_16$Ihat ; results_strat_g3_16$se
```

```
## [1] -0.8227604
```

```
## [1] 5.062025e-07
```

```r
## m = 32
print("Stratified sampling with g3 & m=32:")
```

```
## [1] "Stratified sampling with g3 & m=32:"
```

```r
results_strat_g3_32 = mc_stratified_g3(2000, 32)

results_strat_g3_32$Ihat ; results_strat_g3_32$se
```

```
## [1] -0.823763
```

```
## [1] 1.293607e-07
```

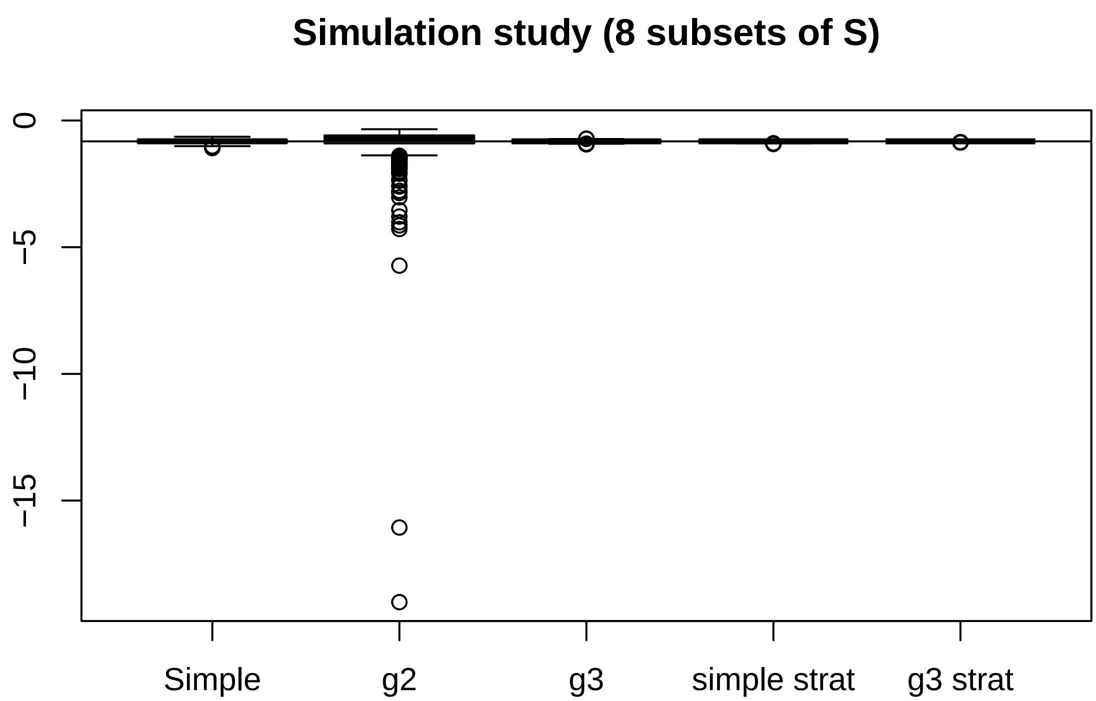
Simulation study

```r
# Create a function to perform an estimator
estimator = function(n, m, B=1000, mainn) {
  ## Simple MC
  res_Ihat = rep(0,B)
  res_se = rep(0, B)

  ## g1
  res_Ihat_g2 = rep(0,B)
  res_se_g2 = rep(0, B)

  ## g3
  res_Ihat_g3 = rep(0,B)
  res_se_g3 = rep(0, B)

  ## Stratified simple
  res_Ihat_strat = rep(0,B)
  res_se_strat = rep(0, B)
  ## Stratified g3
  res_Ihat_g3_strat = rep(0,B)
  res_se_g3_strat = rep(0, B)

  ## Iterate the estimator
  for (i in 1:B){
    ### Store each iteration
    m_simple = mc_simple(n)
    m_g2 = mc_sampling2(n)
    m_g3 = mc_sampling3(n)
    m_strat = mc_stratified(n, m)
    m_g3_strat = mc_stratified_g3(n, m)

    res_Ihat[i] = m_simple$Ihat ; res_se[i] = m_simple$se

    res_Ihat_g2[i] = m_g2$Ihat ; res_se_g2[i] = m_g2$se
```

```
    res_Ihat_g3[i] = m_g3$Ihat ; res_se_g3[i] = m_g3$se

    res_Ihat_strat[i] = m_strat$Ihat ; res_se_strat[i] = m_strat$se

    res_Ihat_g3_strat[i] = m_g3_strat$Ihat ; res_se_g3_strat[i] = m_g3_strat$se

  }

boxplot(res_Ihat, res_Ihat_g2, res_Ihat_g3, res_Ihat_strat, res_Ihat_g3_strat, names=c("Simple", "g2",
abline(h=-1 * pi^2 / 12)
}
```

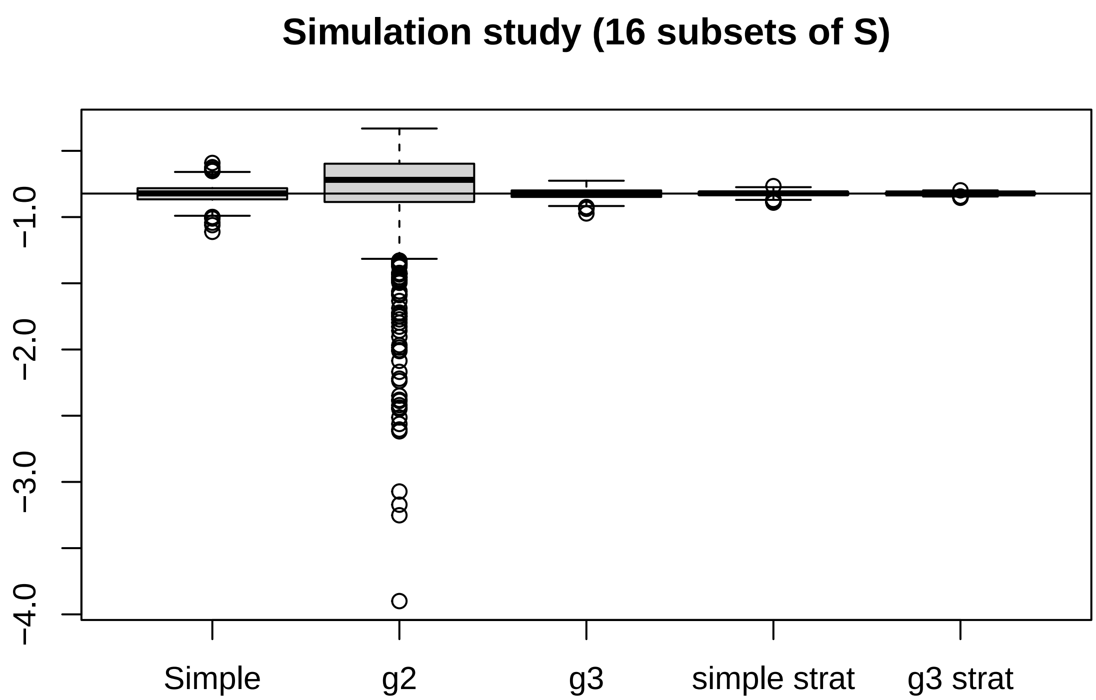
a) 8 sub samples of $S = [0, 1]$

```
estimator(n=200, m = 8, mainn="Simulation study (8 subsets of S)")
```



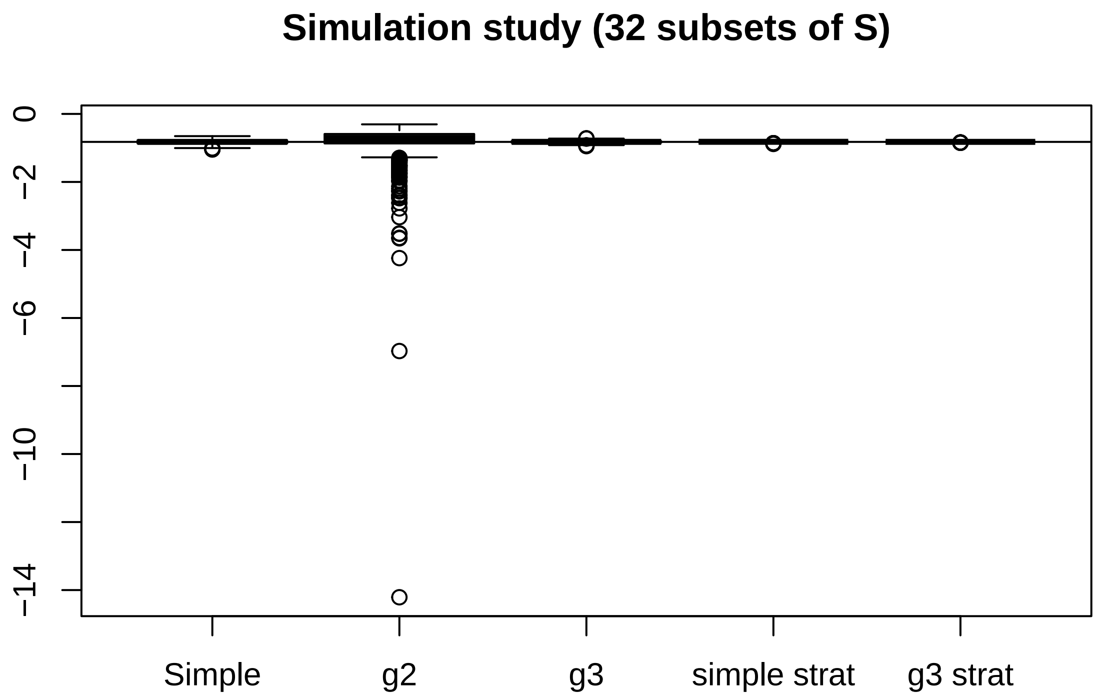
b) 16 sub samples of $S = [0, 1]$

```
estimator(n=200, m = 16, mainn="Simulation study (16 subsets of S)")
```



c) 32 sub samples of $S = [0, 1]$

```
estimator(n=200, m = 32, mainn="Simulation study (32 subsets of S)")
```


Simulation study (32 subsets of S)
0
−2
−4
−6
−10
−14
Simple
g2
g3
simple strat
g3 strat