

Monte Carlo

2022-11-29

Implement of the algorithms

Method 1

```
a_tnorm_mid <- function(fa, l, u, n) {
  break_points = seq(l, u, length=n+1) ###cut the range into n intervals
  res = rep(NA, n)
  for ( i in 1:n) {
    mid = (break_points[i]+break_points[i+1])/2
    res[i] = fa(mid)*dnorm(mid)
  }
  return (mean(res)*(u-l)/(pnorm(u)-pnorm(l)))
}
```

Method 2

```
a_tnorm_naive <- function(fa, l, u, n) {
  x <- rnorm(n, 0, 1)
  x[x>u]=0
  x[x<l]=0
  return (mean(fa(x))/(pnorm(u)-pnorm(l)))
}
```

Method 3

The l function is $\tilde{f}(x) \propto \frac{\log(\phi(u)) - \log(\phi(l))}{u-l} x + \frac{\log(\phi(l))u - \log(\phi(u))l}{u-l}$. Then $\tilde{f}(x) \propto \frac{\log(\phi(u)) - \log(\phi(l))}{u-l} x + \frac{\log(\phi(l))u - \log(\phi(u))l}{u-l}$. Let $a = \frac{\phi(u)}{\phi(l)}$ and $b = u-l$. Then, we consider to simulate from $f(x) = C a^{x/b}$, where $C = \frac{\log(a)}{b(a^{u/b} - a^{l/b})}$. Then $F(x) = \frac{C b}{\log(a)} (a^{x/b} - a^{l/b})$.

```
target <- function(x, l, u) {
  dnorm(x) / (pnorm(u) - pnorm(l))
}

a_tnorm_is <- function(fa, u, l, n) {
  a = dnorm(u) / dnorm(l)
  b = u-l
  C = log(a) / (b*a^(u/b) - b*a^(l/b))
  x = runif(n, 0, 1)
  y = b*log(a^(1/b) + x*log(a)/(C*b)) / log(a)
  mean(fa(y)*target(y, l, u) / (C*a^(y/b)))
}
```

Method 4

The tangent line is $y = -(l+u)/2 * x + b$. Thus, we simulate from exponential distribution on $[l, u]$.

```

proposal <- function(x, l, u) {
  shape = (l+u)/2
  exp(-shape*x) / (exp(-l*shape)-exp(-u*shape))*shape
}

a_tnorm_rej <- function(fa, l, u, n) {
  x= runif(n, 0, 1)
  shape = (l+u)/2
  y = -log(exp(-l*shape)- x*(exp(-l*shape)-exp(-u*shape)))/shape
  M = target((l+u)/2, l, u)/proposal((l+u)/2, l, u)
  y = y[runif(n, 0, 1)<target(y, l, u)/(proposal(y, l, u)*M)]
  mean(fa(y))
}

```

Set l=0,u=1

Question a)

```

fa <- function(x) x^2
l=0
u=1
true_value = a_tnorm_mid(fa, l, u, 1000000)
print(true_value)

```

```
## [1] 0.2911251
```

Question b)

```

n2 = 500000
n3 = 100000
n4 = 180000
time2 = rep(NA, 50)
time3 = rep(NA, 50)
time4 = rep(NA, 50)
for ( i in 1:50) {
  time2[i] = system.time(a_tnorm_naive(fa, l, u, n2))[3]
  time3[i] = system.time(a_tnorm_is(fa, l, u, n3))[3]
  time4[i] = system.time(a_tnorm_rej(fa, l, u, n4))[3]
}
print(mean(time2))

```

```
## [1] 0.036
```

```
print(mean(time3))
```

```
## [1] 0.0358
```

```
print(mean(time4))
```

```
## [1] 0.0338
```

```
rep_times = 5000
res_m2 = rep(NA, rep_times)
res_m3 = rep(NA, rep_times)
res_m4 = rep(NA, rep_times)
for ( i in 1:rep_times) {
  res_m2[i] = a_tnorm_naive(fa, l, u, n2)
  res_m3[i] = a_tnorm_is(fa, l, u, n3)
  res_m4[i] = a_tnorm_rej(fa, l, u, n4)
}
```

```
print(mean(res_m2-true_value)^2)
```

```
## [1] 6.368675e-10
```

```
print(mean(res_m3-true_value)^2)
```

```
## [1] 5.166015e-11
```

```
print(mean(res_m4-true_value)^2)
```

```
## [1] 1.143663e-10
```

We observe that Method (3) achieves the best performance in terms of the MSE.

Set l=0,u=2

Question a)

```
l=0
u=2
true_value = a_tnorm_mid(fa, l, u, 1000000)
print(true_value)
```

```
## [1] 0.7737413
```

Question b)

```

rep_times = 5000
res_m2 = rep(NA, rep_times)
res_m3 = rep(NA, rep_times)
res_m4 = rep(NA, rep_times)
for ( i in 1:rep_times) {
  res_m2[i] = a_tnorm_naive(fa, l, u, n2)
  res_m3[i] = a_tnorm_is(fa, l, u, n3)
  res_m4[i] = a_tnorm_rej(fa, l, u, n4)
}

```

```
print(mean(res_m2-true_value)^2)
```

```
## [1] 4.968769e-10
```

```
print(mean(res_m3-true_value)^2)
```

```
## [1] 2.862519e-11
```

```
print(mean(res_m4-true_value)^2)
```

```
## [1] 5.192027e-10
```

Similarly, we observe that Method (3) achieves the best performance in terms of the MSE. ## Set $l=-3, u=1$

Question a)

```

l=-2
u=1
true_value = a_tnorm_mid(fa, l, u, 1000000)
print(true_value)

```

```
## [1] 0.5724958
```

Question b)

```

rep_times = 5000
res_m2 = rep(NA, rep_times)
res_m3 = rep(NA, rep_times)
res_m4 = rep(NA, rep_times)
for ( i in 1:rep_times) {
  res_m2[i] = a_tnorm_naive(fa, l, u, n2)
  res_m3[i] = a_tnorm_is(fa, l, u, n3)
  res_m4[i] = a_tnorm_rej(fa, l, u, n4)
}

```

```
print(mean(res_m2-true_value)^2)
```

```
## [1] 1.053325e-09
```

```
print(mean(res_m3-true_value)^2)
```

```
## [1] 1.368324e-12
```

```
print(mean(res_m4-true_value)^2)
```

```
## [1] 9.884473e-10
```

Similarly, we observe that Method (3) achieves the best performance in terms of the MSE.

To conclude, with similar time, Method (3) achieves the best performance.