STA 531 HW4

Daniel Truver

2/25/2018

(1) Exercise 3.18 from Casella Book

Let $(Y_1, \ldots, Y_n)$ be a sample from g, and define weights

$$w_i = \frac{f(Y_i)/g(Y_i)}{\sum_{j=1}^{n} f(Y_j)/g(Y_j)}.$$

Implement the SIR algorithm
`For` $m = 1, \ldots, M$
`take` $X_m = Y_i$ `with probability` w_i

We want to show that the SIR algorithm produces a sample from f such that the empirical average

$$\frac{1}{M} \sum_{m=1}^{M} h(X_m)$$

is asymptotically equivalent to the importance sampling estimator based on $(Y_1, \ldots, Y_N)$.

(a) Asymptopia

We will show that the expected value of the SIR estimator is the same as what we would obtain from importance sampling.

$$\begin{aligned} E\left(\frac{1}{M} \sum_{m=1}^{M} h(X_m)\right) &= \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{n} w_i h(Y_i) \\ &= \sum_{i=1}^{n} \frac{h(Y_i) f(Y_i)/g(Y_i)}{\sum_{j=1}^{n} f(Y_j)/g(Y_j)} \\ &= \frac{\sum_{i=1}^{n} h(Y_i) f(Y_i)/g(Y_i)}{\sum_{j=1}^{n} f(Y_j)/g(Y_j)} \end{aligned}$$

We now observe that this is the estimator (3.10) from the Casella book, which we know converges to $E_f h(Y)$ by the strong law of large numbers. So, the expected value of the SIR estimator is the same as the importance sampling estimator.

(b)

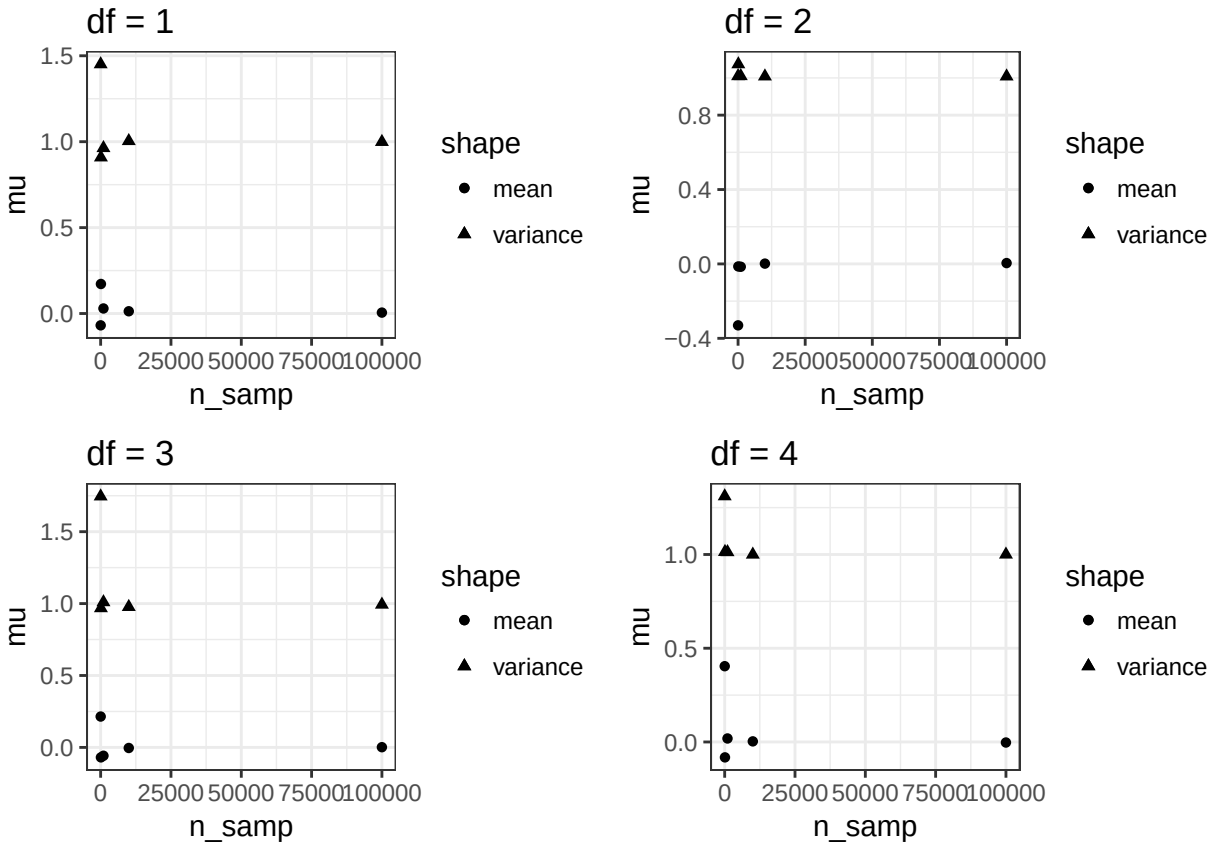
If the algorithm samples without replacement, we are going to get back the distribution g. To see this, take $M = n$, the maximum number of steps in the algorithm here, then, for some permutation s, we will have $(X_1, \ldots, X_M) = (Y_{s(1)}, \ldots, Y_{s(n)})$. The algorithm must stop here since we have used up all available values of Y. These X's are simply draws from g.

---

(2) Importance Sampling with the Normal and T

(a) Estimating Normal from T

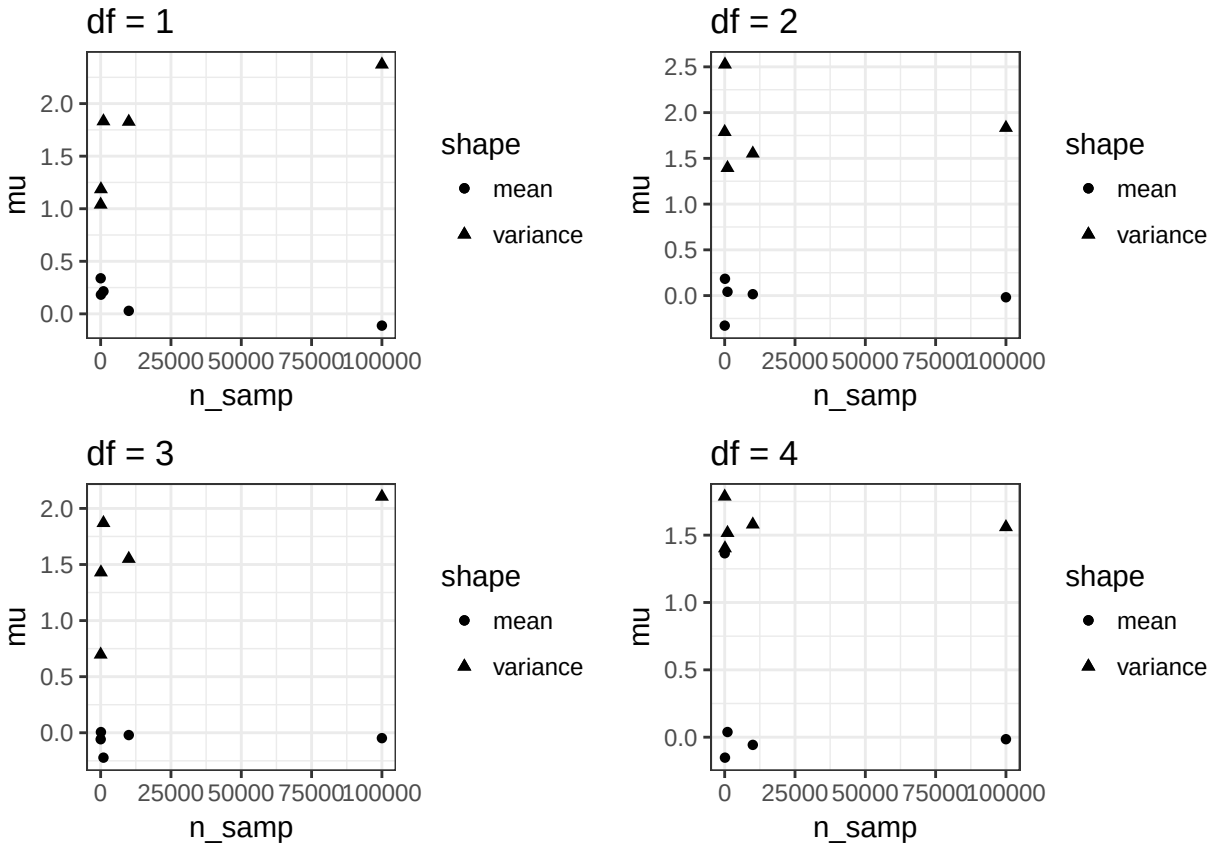
```
nu = c(1,2,3,4)
res = list()
set.seed(2018)
for (v in nu){
  X = rt(n = 100000, df = v)
  fX = dnorm(X)
  gX = dt(df = v, x = X)
  res[[v]] = data.frame(X, fX, gX)
}
glist = list()
i = 1
for (df in res){
  EX = rep(NA, 5)
  EX2 = rep(NA, 5)
  for (k in 1:5){
    ind = 1:10^k
    EX[k] = mean(df$fX[ind]/df$gX[ind] * df$X[ind])
    EX2[k] = mean(df$fX[ind]/df$gX[ind] * (df$X[ind])^2)
  }
  glist[[i]] = list("mu" = EX, "sigma2" = EX2)
  i = i+1
}
k = 1:5
df1 = data.frame(mu = glist[[1]]$mu, var = glist[[1]]$sigma2, n_samp = 10^k)
df2 = data.frame(mu = glist[[2]]$mu, var = glist[[2]]$sigma2, n_samp = 10^k)
df3 = data.frame(mu = glist[[3]]$mu, var = glist[[3]]$sigma2, n_samp = 10^k)
df4 = data.frame(mu = glist[[4]]$mu, var = glist[[4]]$sigma2, n_samp = 10^k)
```



(b) Estimating T from Normal

```
nu = c(1,2,3,4)
res = list()
set.seed(2018)
for (v in nu){
  X = rnorm(n = 100000)
  fX = dt(X, df = v)
  gX = dnorm(X)
  res[[v]] = data.frame(X, fX, gX)
}
glist = list()
i = 1
for (df in res){
  EX = rep(NA, 5)
  EX2 = rep(NA, 5)
  for (k in 1:5){
    ind = 1:10^k
    EX[k] = mean(df$fX[ind]/df$gX[ind] * df$X[ind])
    EX2[k] = mean(df$fX[ind]/df$gX[ind] * (df$X[ind] - EX[k])^2)
  }
  glist[[i]] = list("mu" = EX, "sigma2" = EX2)
  i = i+1
}
k = 1:5
df1 = data.frame(mu = glist[[1]]$mu, var = glist[[1]]$sigma2, n_samp = 10^k)
```

```
df2 = data.frame(mu = glist[[2]]$mu, var = glist[[2]]$sigma2, n_samp = 10^k)
df3 = data.frame(mu = glist[[3]]$mu, var = glist[[3]]$sigma2, n_samp = 10^k)
df4 = data.frame(mu = glist[[4]]$mu, var = glist[[4]]$sigma2, n_samp = 10^k)
```



We're not doing as well here, estimating the t_ν from the Normal. This problem is likely due to the ratio f/g being unbounded in the tails.

(3) Weird Variance

$$\begin{aligned} Var(X/Y) &= Var(XZ) \\ &= E(X^2)E(Z^2) - E(X)^2E(Z)^2 \\ &= (1/3)^2 - (1/2)^4 \\ &= \frac{7}{144} \end{aligned}$$

When we use the delta method, we consider Y as a having an inverse uniform distribution with

$$f_y(y) = \frac{1}{y^2}, \quad y \geq 1$$

but this leads to problems. Namely, the expectations are infinite.

$$\begin{aligned} Var(X/Y) &\approx \frac{Var(X)}{E(Y^2)} + \frac{E(X^2)}{E(Y^4)} Var(Y) \\ &= \frac{(1/12)}{\int_1^\infty y^2 f_y(y)\ dy} + \frac{(1/3)}{\int_1^\infty y^4 f_y(y)\ dy} \left(\int_1^\infty y^2 f_y(y)\ dy - \left(\int_1^\infty y f_y(y)\ dy \right)^2 \right) \\ &= 0 \end{aligned}$$

The difference between this and the true variance calculated above is due to this being an approximation and having to work with a quotient that is not well-behaved in terms of expectations. The theoretical variance is already small to begin with.

I have a feeling this is wrong, so I've also included a simulation.

```
nsim = 1000
varX = rep(NA, nsim)
EY2 = rep(NA, nsim)
EX2 = rep(NA, nsim)
EY4 = rep(NA, nsim)
varY = rep(NA, nsim)
for (i in 1:nsim){
  X = runif(1000)
  Z = runif(1000)
  Y = 1/Z
  varX[i] = var(X)
  EY2[i] = mean(Y^2)
  EX2[i] = mean(X^2)
  EY4[i] = mean(Y^4)
  varY[i] = var(Y)
}
vX = mean(varX)
y2 = mean(EY2)
x2 = mean(EX2)
y4 = mean(EY4)
vY = mean(varY)
delta = vX/y2 + (x2/y4)*vY
cat("Approximate delta is:", delta)
```

```
## Approximate delta is: 4.063822e-12
```