Problem Set 5

Marc Eskew

5/13/2022

Problem 1

1.1

We will develop a control variate to estimate $\alpha = E[Xe^{\frac{X}{2}}]$ by developing a RV, Z, based on the function first three terms of the Taylor expansion of e^x, $g(x) = 1 + x + \frac{x^2}{2}$ which will be correlated with α and the RV $X \sim Weibull(2)$. In order to ensure that $E[Z] = 0$ we need to calculate the expectation of our control variate. When $X \sim Weibull(2)$, then $E[g] = \frac{1}{2}(3 + \sqrt{\pi})$. Therefore, we have a control variate estimate formulated as $W_i = Xe^{\frac{X}{2}} - \beta(1 + X + \frac{X^2}{2} - \frac{1}{2}(3 + \sqrt{\pi}))$

1.2

Using the hazard method, we understand that the Weibull distribution can be characterized by an RV, $X \sim Exp(1)$, or a standard exponential: $X = \left(\frac{W}{\lambda}\right)^k$. This allows us to set $g(x) = x^2$ with $E[g] = 1$ to get the Weibull RV and generate a control variate estimate sampling: $W_i = Xe^{\frac{X}{2}} - \beta(X^2 - 1)$

```
##### Part 1 ####


N <- 10
beta <- 1
val_total <- vector()
val_total_cv <- vector()
for(i in 1:N) {

  x <- rweibull(1,shape =2, scale = 1)

  val <- x * exp(x/2)

  val_total <- append(val_total,val)

  val_cv <- 1+x+(x^2)/2 - .5*(3+sqrt(pi))
  val_total_cv <- append(val_total_cv,val_cv)
}

df1 <- data.frame(val =val_total, cv = val_total_cv, est_cv = val_total-beta*val_total_cv)

sum_val <- sum(df1$est_cv)
std_z <- sqrt(var(df1$est_cv))
err_cal <- 1.96*std_z/(N^.5)
err <- (sum_val/N)*.05

while(err_cal > err) {
```

```
  N <- N + 1
  x <- rweibull(1,shape =2, scale = 1)

  val <- x * exp(x/2)


  val_cv <- 1+x+(x^2)/2 - .5*(3+sqrt(pi))

  df1 <- bind_rows(df1,data.frame(val =val, cv = val_cv, est_cv = val-beta*val_cv))
  sum_val <- sum(df1$est_cv)
  std_z <- sqrt(var(df1$est_cv))
  err_cal <- 1.96*std_z/(N^.5)
  err <- (sum_val/N)*.05
}

sum_val/N
```

```
## [1] 1.582273
```

Using the CV formula from the first part resulted in a estimate of **1.5822725 +- 5% with 95% confidence** after 47 samples

```
##### Part 2 ####


N <- 10
beta <- 1
val_total <- vector()
val_total_cv <- vector()
for(i in 1:N) {

  x <- rweibull(1,shape =2, scale = 1)

  val <- x * exp(x/2)

  val_total <- append(val_total,val)

  val_cv <- x^2 - 1
  val_total_cv <- append(val_total_cv,val_cv)
}

df1 <- data.frame(val =val_total, cv = val_total_cv, est_cv = val_total-beta*val_total_cv)

sum_val <- sum(df1$est_cv)
std_z <- sqrt(var(df1$est_cv))
err_cal <- 1.96*std_z/(N^.5)
err <- (sum_val/N)*.05

while(err_cal > err) {

  N <- N + 1
  x <- rweibull(1,shape =2, scale = 1)

  val <- x * exp(x/2)
```

```
  val_cv <- 1+x+(x^2)/2 - .5*(3+sqrt(pi))

  df1 <- bind_rows(df1,data.frame(val =val, cv = val_cv, est_cv = val-beta*val_cv))
  sum_val <- sum(df1$est_cv)
  std_z <- sqrt(var(df1$est_cv))
  err_cal <- 1.96*std_z/(N^.5)
  err <- (sum_val/N)*.05

}

sum_val/N
```

```
## [1] 1.663004
```

Using the CV formula from the first part resulted in a estimate of **1.6630038 +- 5% with 95% confidence** after 69 samples. The method with the most efficient sampling would be the preferred alternative.

Problem 2

2.1

When the RVs are positively correlated with $\rho = 1$ and the variance of one of the X_i is twice the other X_i and the variance is double, the reversal of the sign on β will allow all RVs to maintain $Var(W(\beta_1, \beta_2)) = 0$. So, as an example, if $Y = 2X_1$ and $Y = X_2$ the beta values would have to be inverse to get 0 variance.

2.2

The optimal values of B_i can be calculate by comparing the covariance to Y.

$$\beta_1^* = \frac{Cov(Y, X_1)}{VarX_1}$$
$$\beta_2^* = \frac{Cov(Y, X_2)}{2VarX_2}$$

Alternatively, we could call a linear regression function to determine the optimal values of β. The regression function will determine the appropriate values of α and β and can be used as a method of deterining the optimal value.

Problem 3

3.1

```
rv_x <- function(i){
  rexp(1,rate = 1/i)
}
reps <- 1000
val_total <- vector()
for(i in 1:reps) {

  x1 <- rv_x(1)
  x2 <- rv_x(2)
  x3 <- rv_x(3)
  x4 <- rv_x(4)
```

```r
  val <- max(x1+x2,x3+x4,x1+x4,x3+x2)
  val_total <- append(val_total,val)
}

std_val <- sqrt(var(val_total))
err_a <- (1.96*std_val)/sqrt(reps)

err_a
```

```
## [1] 0.2987476
```

```r
mean(val_total)
```

```
## [1] 7.985835
```

```r
mean(val_total) + err_a
```

```
## [1] 8.284582
```

```r
mean(val_total) - err_a
```

```
## [1] 7.687087
```

α is between **7.687087 and 8.2845821** with 95% confidence after 1000 replications.

3.2

Utilizing control variates will reduce the error on the confidence interval by reducing the σ observed in our samples.

```r
val_total_cv <- vector()
val_total <- vector()

for(i in 1:reps) {

  x1 <- rv_x(1)
  x2 <- rv_x(2)
  x3 <- rv_x(3)
  x4 <- rv_x(4)

  val <- max(x1+x2,x3+x4,x1+x4,x3+x2)
  val_total <- append(val_total,val)

  val_cv <- x1+x4+x2+x3 - 10
  val_total_cv <- append(val_total_cv,val_cv)

}

beta <- 1


est_cv <- val_total - beta*val_total_cv


std_val_cv <- sqrt(var(est_cv))
err_cv <- (1.96*std_val_cv)/sqrt(reps)
```

```
err_cv
```

```
## [1] 0.09246223
```

```
mean(est_cv)
```

```
## [1] 7.867654
```

```
mean(est_cv) + err_cv
```

```
## [1] 7.960117
```

```
mean(est_cv) - err_cv
```

```
## [1] 7.775192
```

With control variates, α is between **7.9601166 and 7.9601166** with 95% confidence after 1000 replications. Our halfwidths have been reduced from **0.2987476 to 0.0924622**.

Problem 4

4.1

We can model this by looking at the joint probabilities of a Poisson process seeing n arrivals and the probability of sum of claims for each set of n arrivals exceeds b. In order to do this we would need to model Y as:

$$\begin{aligned}
\alpha &= P(Y > b) \\
X &\sim Pois(100) \\
Z &\sim Exp(.01) \\
Y &= x \sum_{i=1}^{x} z_i
\end{aligned}$$

To simplify, the Z can be represented as an Gamma (Erlang) distribution with is the distribution for the sum of k independent exponential variables.

$$\begin{aligned}
\alpha &= P(Y > b) \\
X &\sim Pois(100) \\
Z &\sim Gamma(x, .01) \\
Y &= XZ
\end{aligned}$$

We can write the expression for a as:

$$\alpha = \sum_{i=1}^{\infty} \frac{\lambda_p^i}{i!} e^{-\lambda_p} (1 - \sum_{n=0}^{i} \frac{1}{n!} e^{-\lambda_g b} (\lambda_g b)^n)$$

4.2

Using the known density functions of the Poisson distribution, we can identify p_i associated with strata i and assign set number of samples to draw from that strata.

```
str1 <- ppois(90,100)
strmid <- dpois(91:109,100)
str2 <- 1-ppois(109,100)

sum(str1,strmid,str2)
```

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

```r
total <- data.frame(strata= 90:110,val = c(str1,strmid,str2)) %>%
  mutate(reps = round(10000*val))

total[1,3] <- total[1,3] -3

head(total)
```

```
##   strata        val reps
## 1     90 0.17138512 1711
## 2     91 0.02751532  275
## 3     92 0.02990796  299
## 4     93 0.03215910  322
## 5     94 0.03421181  342
## 6     95 0.03601243  360
```

Then we can use stratified sampling to attain draws to estimate our quantity of interest.

```r
df <- data.frame()

for(i in 1:nrow(total)) {

  for(j in 1:total[i,3]){
    if(i == 1){
      num <- 100
      while(num>90){
        num <- rpois(1,100)
      }
      val <- rgamma(1,shape = num,rate = 1/100)
    } else  if(i == nrow(total)){
      num <- 90
      while(num<100){
        num <- rpois(1,100)
      }
      val <- rgamma(1,shape = num,rate = 1/100)
    } else {
      num <- total[i,1]
      val <- rgamma(1,shape=num,rate = 1/100)
    }

    df <- bind_rows(df,data.frame(bin = i, arrivals = num, val = val))

  }

}

sample_n(df,10)
```

```
##    bin arrivals       val
## 1    7       96  9827.688
## 2    1       86  8895.588
## 3    1       76  8268.389
## 4    5       94 10213.709
## 5    1       75  8630.081
## 6   11      100 10165.411
```

```
## 7    1       84  9055.367
## 8   15      104 11513.628
## 9   15      104 10969.641
## 10  21      106 10889.709
```

By utilizing boot strapping, we can estimate a 95% confidence interval for $P(Y > b)$

```
prob_vec <- vector()
for(i in 1:2000){
  sampledf <- sample_n(df,nrow(df),replace = TRUE)

  sampledf <- sampledf %>%
    mutate(ab = case_when(
      val  < 11960 ~ TRUE,
      TRUE ~ FALSE
    ))

  prob_vec <- append(prob_vec,prop.table(table(sampledf$ab))[[1]])

}

quantile(prob_vec,c(.025,0.975))
```

```
##   2.5%  97.5%
## 0.0539 0.0631
```

We are 95% confident that the α lies between **0.0539 and 0.0631**.