Project Code

Ambrose

December 18th, 2020

Loading in libraries.

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE)
library(qrng)
library(actuar)
```

Initialization of variables used in code.

```
alpha <- c(1.25, 1.75)
t <- c(5,10,15)
x <- c(50, 75, 100)
n <- c(10000, 1000000)
```

Naive Monte Carlo Method code for Pareto Distribution

```
NMC_pareto <- function(n, a, t, x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rpareto(N[i], shape=a, scale = 1)
    if (sum(Xi) > x){
      mu[i] <- 1
    }
    else{
      mu[i] <- 0
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
mc_mean <- vector(length = len)
mc_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- NMC_pareto(n[i], alpha[j], t[k], x[m])
        mc_mean[s] <- round(nmc$mc_mean,digits=6)
        mc_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Antivariate Variate technique code for Pareto Distribution

```
AV_MOD <- function(n,a,t,x){
  mu <- vector(length=n/2)
  N <- rpois(n/2, t)
  for (i in 1:(n/2)){
    U <- runif(N[i])
    Xi <- qpareto(U, , shape=a, scale = 1)
    xi_tilde <- qpareto(1- U, shape=a, scale=1)
    mu[i] <- 0
    if (sum(Xi) > x){
        mu[i] <- 1
    }
    if (sum(xi_tilde) > x){
        mu[i] <- mu[i] + 1
    }
    mu[i] <- mu[i] / 2
  }
  mu_mean <- mean(mu)
  mu_var <- sum((mu - mu_mean)^2) / (n/2 - 1)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / (n/2))
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- AV_MOD(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

The Tail of the Pareto Distribution

```
pareto_tail <- function(x, a){
  (1+x)^(-a)
}
```

Conditional Monte Carlo Method conditioned on order statistics code for Pareto Distribution

```
CMC_order_pareto <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rpareto(N[i], shape=a, scale = 1)
    Xi <- sort(Xi)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    if (N[i] >= 2){
      mu[i] <- pareto_tail(max(x - (sum(Xi) - Xi[N[i]]),  Xi[N[i]-1]),a) / pareto_tail(Xi[N[i]-1],a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_order_pareto(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient version code for Pareto Distribution

```
CMC_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rpareto(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    else if (N[i] >= 2){
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*pareto_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_Efficient(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Naive Monte Carlo Method code for fixed t for Pareto Distribution

```
NMC_pareto <- function(n, a, t, x){
  mu <- vector(length=n)
  for (i in 1:n){
    Xi <- rpareto(t, shape=a, scale = 1)
    if (sum(Xi) > x){
      mu[i] <- 1
    }
    else{
      mu[i] <- 0
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
mc_mean <- vector(length = len)
mc_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- NMC_pareto(n[i], alpha[j], t[k], x[m])
        mc_mean[s] <- round(nmc$mc_mean,digits=6)
        mc_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

CMC Method Efficient fixed code for Pareto Distribution

```
CMC_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  for (i in 1:n){
    Xi <- rpareto(t, shape=a, scale = 1)
    mu[i] <- 0
    if (t == 1){
      mu[i] <- pareto_tail(x,a)
    }
    else if (t >= 2){
      S_Ni <- sum(Xi[1:(t-1)])
      Xi <- sort(Xi[1:(t-1)])
      mu[i] <- t*pareto_tail(max(Xi[t-1], x - S_Ni),a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_Efficient(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

RQMC code for Pareto Distribution

```
RQMC_pareto <- function(n, a, t, x, m=25){
  mu <- vector(length=n)
    U <- sobol(n, randomize = "digital.shift")
    N <- qpois(U, t)
    for (i in 1:n){
      Xi <- c(0)
      if (N[i] >= 1){
        U <- sobol(N[i], randomize = "digital.shift")
        Xi <- qpareto(U, shape=a, scale = 1)
      }
      if (sum(Xi) > x){
        mu[i] <- 1
      }
      else{
        mu[i] <- 0
      }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mean(mu_mean), HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- RQMC_pareto(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

CMC Method conditioned on order statistics combined with RQMC code for Pareto Distribution

```
CMC_order_RQMC_pareto <- function(n,a,t,x){
  mu <- vector(length=n)
  U <- sobol(n, randomize = "digital.shift")
  N <- qpois(U, t)
  for (i in 1:n){
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    if (N[i] >= 2){
      U <- sobol(N[i], randomize = "digital.shift")
      Xi <- qpareto(U, shape=a, scale = 1)
      Xi <- sort(Xi)
      mu[i] <- pareto_tail(max(x - (sum(Xi) - Xi[N[i]]),  Xi[N[i]-1]),a) / pareto_tail(Xi[N[i]-1],a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_order_RQMC_pareto(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with RQMC version code for Pareto Distribution

```
CMC_RWMC_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  U <- sobol(n, randomize = "digital.shift")
  N <- qpois(U, t)
  for (i in 1:n){
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    else if (N[i] >= 2){
      U <- sobol(N[i], randomize = "digital.shift")
      Xi <- qpareto(U, shape=a, scale = 1)
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*pareto_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_RWMC_Efficient(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with Stratification version code for Pareto Distribution

```
pareto_tail <- function(x, a){
  (1+x)^(-a)
}
CMC_STR <- function(n,a,t,x){
  if (t == 5){
    m <- 9
  }
  else if (t == 10){
    m <- 16
  }
  else{
   m <- 22
  }
  p <- vector(length=m)
  for (i in 1:(m-1)) {
    p[i] <- dpois(i-1, t)
  }
  p[m] <- 1 - sum(p)
  N <- vector(length=m)
  for (i in 1:m) {
    N[i] <- round(p[i]*n)
  }
  mu_mean_str <- vector(length=m)
  mu_var_str <- vector(length=m)
  for (j in 1:(m-1)) {
    mu_str <- vector(length = N[j])
    for (i in 1:N[j]) {
      Ni_T <- j-1
      Xi <- rpareto(Ni_T, shape=a, scale = 1)
      mu_str[i] <- 0
      if (Ni_T == 1){
      mu_str[i] <- pareto_tail(x,a)
```

```
      }
      else if (Ni_T >= 2){
      S_Ni <- sum(Xi[1:(Ni_T-1)])
      Xi <- sort(Xi[1:(Ni_T-1)])
      mu_str[i] <- Ni_T*pareto_tail(max(Xi[Ni_T-1], x - S_Ni),a)
    }
  }
  mu_mean_str[j] <- mean(mu_str)
  mu_var_str[j] <- 0
  if (length(mu_str) != 1){
    mu_var_str[j] <- sum((mu_str - mean(mu_str))^2) / (length(mu_str) - 1)
  }
  }
  mu_str <- vector(length = N[m])
    for (i in 1:N[m]) {
      Ni_T <- 0
      while(Ni_T < m-1){
       Ni_T <- rpois(1,t)
      }
      Xi <- rpareto(Ni_T, shape=a, scale = 1)
      mu_str[i] <- 0
      if (Ni_T == 1){
      mu_str[i] <- pareto_tail(x,a)
      }
      else if (Ni_T >= 2){
      S_Ni <- sum(Xi[1:(Ni_T-1)])
      Xi <- sort(Xi[1:(Ni_T-1))
      mu_str[i] <- Ni_T*pareto_tail(max(Xi[Ni_T-1], x - S_Ni),a)
    }
  }
  mu_mean_str[m] <- mean(mu_str)
  mu_var_str[m] <- 0
  if (length(mu_str) != 1){
    mu_var_str[m]<- sum((mu_str - mean(mu_str))^2) / (length(mu_str) - 1)
  }
mu_cmc_str <- (p %*% mu_mean_str)[1,1]
mu_str_var <- 1/n*((p %*% mu_var_str)[1,1])
half_width <- qnorm(1-0.05/2)*sqrt(mu_str_var)
nmc <- list(mc_mean = mu_cmc_str, HW=half_width)
nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_STR(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with Control Variate version code for Pareto Distribution

```
calc_beta <- function(num_runs, a, t, x){
mu <- vector(length=num_runs)
  N <- rpois(num_runs, t)
  for (i in 1:num_runs){
    Xi <- rpareto(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    else if (N[i] >= 2){
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*pareto_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
mu_mc_hat <- NMC_pareto(num_runs, a, t, x)$mc_mean
mu_N_i_hat <- mean(N)
var_N_i <- t
dot_prod <- mu%*%N
beta <- (dot_prod[1,1] - num_runs*(mu_mc_hat*mu_N_i_hat)) / (num_runs-1) * 1/ var_N_i
beta
}
CMC_CV_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rpareto(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- pareto_tail(x,a)
    }
    else if (N[i] >= 2){
```

```
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*pareto_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  beta <- calc_beta(1000, a, t, x)
  mu_N_i <- t
  Y_cv <- mu + beta*(mu_N_i - N)
  mu_mean <- mean(Y_cv)
  mu_var <- var(Y_cv)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_CV_Efficient(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Importance Sampling with Hazard Rate Twisting For Pareto code

```
Lambda <- function(x, a){
  a*log(1+x)
}

f_theta_inv(u, a, theta){
  exp((-log(1-u)) / (a*(1-theta))) - 1
}

IS_Hazard_Rate_Twist <- function(n,a,t,x){
  mu <- vector(length=n)
  theta <- 1 - 1 / Lambda(x,a)
  N <- rpois(n, t)
  for (i in 1:n){
    U <- runif(N[i])
    Xi <- f_theta_inv(U, a, theta)
    mu[i] <- 0
    if (sum(Xi) > x){
      mu[i] = 1/(1-theta)^N[i]*exp(-theta*sum(Lambda(Xi)))
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- IS_Hazard_Rate_Twist(n[i], alpha[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Weibull Case Code

Initialization of variables used in code.

Naive Monte Carlo Method code for Weibull Distribution

```
NMC_weibull <- function(n, b, t, x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rweibull(N[i], shape=b)
    if (sum(Xi) > x){
      mu[i] <- 1
    }
    else{
      mu[i] <- 0
    }
  }
  mu_mean<- mean(mu)
 mu_var<- var(mu)
  half_width<- qnorm(1-0.05/2)* sqrt(mu_var / n)
  nmc<- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
mc_mean <- vector(length = len)
mc_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- NMC_weibull(n[i], beta[j], t[k], x[m])
        mc_mean[s] <- round(nmc$mc_mean,digits=6)
        mc_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Antivariate Variate technique code for Weibull Distribution

```
AV_MOD_WEI <- function(n,a,t,x){
  mu <- vector(length=n/2)
  N <- rpois(n/2, t)
  for (i in 1:(n/2)){
    U <- runif(N[i])
    Xi <- qweibull(U, , shape=a, scale = 1)
```

```
    xi_tilde <- qweibull(1- U, shape=a, scale=1)
    mu[i] <- 0
    if (sum(Xi) > x){
        mu[i] <- 1
    }
    if (sum(xi_tilde) > x){
        mu[i] <- mu[i] + 1
    }
    mu[i] <- mu[i] / 2
  }
  mu_mean <- mean(mu)
  mu_var <- sum((mu - mu_mean)^2) / (n/2 - 1)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / (n/2))
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- AV_MOD_WEI(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

The Tail of the Pareto Distribution

```
weibull_tail <- function(x, b){
  e^(-x^b)
}
```

Conditional Monte Carlo Method conditioned on order statistics code for Weibull Distribution

```
CMC_order_weibull <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rweibull(N[i], shape=a, scale = 1)
    Xi <- sort(Xi)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
    if (N[i] >= 2){
      mu[i] <- weibull_tail(max(x - (sum(Xi) - Xi[N[i]]),  Xi[N[i]-1]),a) / weibull_tail(Xi[N[i]-1],a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_order_weibull(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient version code for weibull Distribution

```
CMC_Efficient_WEI <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rweibull(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
    else if (N[i] >= 2){
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*weibull_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_Efficient_WEI(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

RQMC code for weibull Distribution

```
RQMC_weibull <- function(n, a, t, x, m=25){
  mu <- vector(length=n)
    U <- sobol(n, randomize = "digital.shift")
    N <- qpois(U, t)
    for (i in 1:n){
      Xi <- c(0)
      if (N[i] >= 1){
        U <- sobol(N[i], randomize = "digital.shift")
        Xi <- qweibull(U, shape=a, scale = 1)
      }
      if (sum(Xi) > x){
        mu[i] <- 1
      }
      else{
        mu[i] <- 0
      }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mean(mu_mean), HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- RQMC_weibull(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

CMC Method conditioned on order statistics combined with RQMC code for weibull Distribution

```
CMC_order_RQMC_weibull <- function(n,a,t,x){
  mu <- vector(length=n)
  U <- sobol(n, randomize = "digital.shift")
  N <- qpois(U, t)
  for (i in 1:n){
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
    if (N[i] >= 2){
      U <- sobol(N[i], randomize = "digital.shift")
      Xi <- qweibull(U, shape=a, scale = 1)
      Xi <- sort(Xi)
      mu[i] <- weibull_tail(max(x - (sum(Xi) - Xi[N[i]]),  Xi[N[i]-1]),a) /weibull_tail(Xi[N[i]-1],a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_order_RQMC_weibull(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with RQMC version code for weibull Distribution

```
CMC_RQMC_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  U <- sobol(n, randomize = "digital.shift")
  N <- qpois(U, t)
  for (i in 1:n){
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
    else if (N[i] >= 2){
      U <- sobol(N[i], randomize = "digital.shift")
      Xi <- qweibull(U, shape=a, scale = 1)
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*weibull_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  mu_mean <- mean(mu)
  mu_var <- var(mu)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_RQMC_Efficient(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with Stratification version code for Weibull Distribution

```
CMC_STR <- function(n,a,t,x){
  if (t == 5){
    m <- 9
  }
  else if (t == 10){
    m <- 16
  }
  else{
   m <- 22
  }
  p <- vector(length=m)
  for (i in 1:(m-1)) {
    p[i] <- dpois(i-1, t)
  }
  p[m] <- 1 - sum(p)
  N <- vector(length=m)
  for (i in 1:m) {
    N[i] <- round(p[i]*n)
  }
  mu_mean_str <- vector(length=m)
  mu_var_str <- vector(length=m)
  for (j in 1:(m-1)) {
    mu_str <- vector(length = N[j])
    for (i in 1:N[j]) {
      Ni_T <- j-1
      Xi <- rweibull(Ni_T, shape=a, scale = 1)
      mu_str[i] <- 0
      if (Ni_T == 1){
      mu_str[i] <- weibull_tail(x,a)
      }
      else if (Ni_T >= 2){
      S_Ni <- sum(Xi[1:(Ni_T-1)])
```

```
      Xi <- sort(Xi[1:(Ni_T-1)])
      mu_str[i] <- Ni_T*weibull_tail(max(Xi[Ni_T-1], x - S_Ni),a)
    }
  }
  mu_mean_str[j] <- mean(mu_str)
  mu_var_str[j] <- 0
  if (length(mu_str) != 1){
    mu_var_str[j] <- sum((mu_str - mean(mu_str))^2) / (length(mu_str) - 1)
  }
  }
  mu_str <- vector(length = N[m])
    for (i in 1:N[m]) {
      Ni_T <- 0
      while(Ni_T < m-1){
       Ni_T <- rpois(1,t)
      }
      Xi <- rweibull(Ni_T, shape=a, scale = 1)
      mu_str[i] <- 0
      if (Ni_T == 1){
      mu_str[i] <- weibull_tail(x,a)
      }
      else if (Ni_T >= 2){
      S_Ni <- sum(Xi[1:(Ni_T-1)])
      Xi <- sort(Xi[1:(Ni_T-1))
      mu_str[i] <- Ni_T*weibull_tail(max(Xi[Ni_T-1], x - S_Ni),a)
    }
  }
  mu_mean_str[m] <- mean(mu_str)
  mu_var_str[m] <- 0
  if (length(mu_str) != 1){
    mu_var_str[m]<- sum((mu_str - mean(mu_str))^2) / (length(mu_str) - 1)
  }
mu_cmc_str <- (p %*% mu_mean_str)[1,1]
mu_str_var <- 1/n*((p %*% mu_var_str)[1,1])
half_width <- qnorm(1-0.05/2)*sqrt(mu_str_var)
nmc <- list(mc_mean = mu_cmc_str, HW=half_width)
nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_STR(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```

Conditional Monte Carlo Method efficient combined with Control Variate version code for Weibull Distribution

```
calc_beta <- function(num_runs, a, t, x){
mu <- vector(length=num_runs)
  N <- rpois(num_runs, t)
  for (i in 1:num_runs){
    Xi <- rweibull(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
    else if (N[i] >= 2){
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*weibull_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
mu_mc_hat <- NMC_weibull(num_runs, a, t, x)$mc_mean
mu_N_i_hat <- mean(N)
var_N_i <- t
dot_prod <- mu%*%N
beta <- (dot_prod[1,1] - num_runs*(mu_mc_hat*mu_N_i_hat)) / (num_runs-1) * 1/ var_N_i
beta
}
CMC_CV_Efficient <- function(n,a,t,x){
  mu <- vector(length=n)
  N <- rpois(n, t)
  for (i in 1:n){
    Xi <- rweibull(N[i], shape=a, scale = 1)
    mu[i] <- 0
    if (N[i] == 1){
      mu[i] <- weibull_tail(x,a)
    }
```

```
    else if (N[i] >= 2){
      S_Ni <- sum(Xi[1:(N[i]-1)])
      Xi <- sort(Xi[1:(N[i]-1)])
      mu[i] <- N[i]*weibull_tail(max(Xi[N[i]-1], x - S_Ni),a)
    }
  }
  beta <- calc_beta(1000, a, t, x)
  mu_N_i <- t
  Y_cv <- mu + beta*(mu_N_i - N)
  mu_mean <- mean(Y_cv)
  mu_var <- var(Y_cv)
  half_width <- qnorm(1-0.05/2)*sqrt(mu_var / n)
  nmc <- list(mc_mean = mu_mean, HW=half_width)
  nmc
}
```

Code used to create the table values.

```
options(scipen=999)
s <- 1
len <- 36
vrt_mean <- vector(length = len)
vrt_HW <- vector(length = len)
for (i in 1:2){
  for (j in 1:2){
    for (k in 1:3){
      for (m in 1:3){
        nmc <- CMC_CV_Efficient(n[i], beta[j], t[k], x[m])
        vrt_mean[s] <- round(nmc$mc_mean,digits=6)
        vrt_HW[s] <- round(nmc$HW,digits=6)
        s <- s + 1
      }
    }
  }
}
```