Homework-3

Helene Behrens

1 november 2019

The code is written in R. Seeds are set arbitrarily to ensure cohesion between text and code.

Exercise 1

We will show that in estimating $z = \mathbf{E}[\sqrt{1-U^2}]$, where U is a standard uniform random variable, U^2 is a better choice of control variable than simply U. We will show this by calculating the variance for each new estimator;

$$\mathrm{Var}[\hat{z} + \alpha(\hat{w} - \mathbf{E}[W])] = \mathrm{Var}[\hat{z}] + \alpha^2\mathrm{Var}[\hat{w}] + 2\alpha\mathrm{Cov}[\hat{z}, \hat{w}],$$

where $\hat{z}$ and $\hat{w}$ are the Crude Monte Carlo estimator and the sample mean of the control variable, respectively. α is calculated as

$$\alpha = -\frac{\mathrm{Cov}(Z, W)}{\mathrm{Var}(W)}.$$

Note that we have used that $\mathrm{Var}[\hat{z}] = \mathrm{Var}[Z]/n$, where n is the sample size. We then get:

```r
n <- 100
u <- runif(n)
z <- sqrt(1 - u^2)
zhat <- mean(z)
varz <- var(z)/n

w1 <- u
w1hat <- mean(w1)

w2 <- u^2
w2hat <- mean(w2)

var1 <- var(w1)/n
var2 <- var(w2)/n

cov1 <- cov(z,w1)/n
cov2 <- cov(z,w2)/n

alpha1 <- -cov(z,w1)/var(w1)
alpha2 <- -cov(z,w2)/var(w2)

totvar1 <- varz + alpha1^2*var1 + 2*alpha1*cov1
totvar2 <- varz + alpha2^2*var2 + 2*alpha2*cov2
cat("The variance with control varable U: ",totvar1,"\nThe variance with control varable U^2: ", totvar
```

```
## The variance with control varable U:  7.437979e-05
## The variance with control varable U^2:  1.524788e-05
```

We see that the variance is lower when U^2 is the control variable, so we conclude that this is the better choice. ## Exercise 2

We will find the value of p, the probability that there are at least 20 claims a given week by simulation. However, we note that, given the rate, we can find an analytical value for p. We use this as a reference to validate our simulation results. For a Poisson process with a given rate λ, we have that

$$\mathbf{P}(\text{no of claims} \geq 20) = 1 - \mathbf{P}(\text{no of claims} \leq 10) = 1 - \sum_{i=0}^{19} \lambda^i \frac{e^{-\lambda}}{i!}$$

We will then use this expression, together with a Crude Monte Carlo simulation for the rate λ only, to get a good approximation of the true answer:

```
set.seed(200)

Poiss <- function(i, lamb){
  return(lamb**i*exp(-lamb)/factorial(i))
}

Theoretical <- function(R){
  ps = c()
  for (r in 1:R){
    u = runif(1)
    lamb = 15/(0.5 + u)
    x = c(0:19)
    p = 1 - sum(lamb**x*exp(-lamb)/factorial(x))
    ps = c(ps,p)
  }
  return(ps)
}

conf_int_mean <- function(R, z){
  zhat = sum(z)/R
  err = 1.96*sd(z)/sqrt(R)
  rhs = zhat - err
  lhs = zhat + err
  return(list("rhs" = rhs, "lhs" = lhs))
}

ps <- Theoretical(10000)
conf_int_mean(10000,ps)
```

```
## $rhs
## [1] 0.2796378
##
## $lhs
## [1] 0.2922149
```

Above, we see a 95% confidence interval for the value of p.

2.1

For a Poisson porcess we have

$$S_n := \sum_{i=1}^{n} X_i,$$

where X_i are exponential random variables with mean $1/\lambda$. S_n is the time it takes for n claims to arrive, when the rate parameter of the process is λ. (Recall that the expected waiting time between arrivals in a Poisson process is exponential distributed.) So, S_{20} is the time for the first 20 claims to arrive. Thus, if $S_{20} \leq 1$, we have 20 claims in one time unit, in our case one week. This leads to the Crude Monte Carlo estimator

$$\hat{p} = \frac{1}{R}\sum_{r=1}^{R} \mathbf{1}\{S_{20,r} \leq 1\},$$

for the probability that the branch receives 20 or more claims per week. Recall that an exponential variable with mean $1/\lambda$ can be simulated by

$$\exp = -\frac{\log(U)}{\lambda},$$

where U is a standard uniform random variable. So, for each repetition, we sample one uniform random variable, hereafter referred to as u, to calculate the rate parameter λ_r and 20 uniform random variables, $\mathbf{U}$ to calculate $S_{20,r}$. We then get:

```r
set.seed(201)

CMC <- function(R){
  Z = c()
  for(r in 1:R){
    u <- runif(1) #Note: we resample the rate for each repetition
    U <- runif(20)
    lamb <- 15/(0.5+u)
    if (sum(-log(U)/lamb) <= 1){
      Z = c(Z,1)
    }
    else {
      Z = c(Z,0)
    }
  }
  return(list("Z" = Z, "R" = R))
}

CMC <- CMC(1000)
cat("The Crude Monte Carlo estimator gives the 95% confidence interval for p: \n")
```

```
## The Crude Monte Carlo estimator gives the 95% confidence interval for p:
```

```r
conf_int_mean(CMC$R, CMC$Z)
```

```
## $rhs
## [1] 0.2453738
##
## $lhs
## [1] 0.3006262
```

```r
var_CMC <- var(CMC$Z)/CMC$R #The variance of the Crude Monte Carlo method; this is the one we are tryin
cat("The variance is then ", var_CMC)
```

```
## The variance is then  0.0001986697
```

CMC is now our Crude Monte Carlo estimator of p, the probability that the branch gets more than 20 claims in a week. We sample this a number of times, and use the sample variance to approximate the variance of this estimator: 1.9866967×10^{-4}. This is the variance we are trying to improve.

2.2

We will try two approaces to reducing the variance; the method of conditional expectation combined with either a control vairable or antithetic sampling.

We will start by implementing conditional expectation. This method entails conditioning on one of the random variables used for sampling, to reduce the number of variables that need to be sampled and to reduce variance.

We use that the density $f^{*n}(x)$ of the convolution $S_n := \sum_{i=1}^{n} X_i$ can be conditionally estimated by $f_X(x - S_{n-1})$, given S_{n-1}. Since we are interested in $\mathbf{P}(S_{20} \leq 1)$, we look at the exponential cumulative distribution $F_X(x - S_{19})$. So, we sample only S_{19}, not S_{20} as before, and find

$$\mathbf{P}(S_{20} \leq 1) = F_X(1 - S_{19}) = 1 - e^{-\lambda(1-S_{19})}.$$

In our code, we include that if $S_{19} > 1$, there is no chance that there will be more than 20 claims in one week, and set p_r to zero. We get:

```r
set.seed(202)

ConditMC <- function(R){
  Z = c()
  us <- c()
  for(r in 1:R){
    u <- runif(1) #Note: we resample the rate at each repetition
    us = c(us,u)
    U <- runif(19)
    lamb <- 15/(0.5+u)
    S19 = -(1/lamb)*sum(log(U))
    if (S19 >= 1){
      Z = c(Z,0)
    }
    else {
      p = 1 - exp(-lamb*(1 - S19))
      Z = c(Z,p)
    }
  }
  return(list("Z" = Z, "us" = us, "R" = R))
}

Z_cnd <- ConditMC(1000)

cat("This leaves us with a new confidence interval for p: \n")
```

```
## This leaves us with a new confidence interval for p:
```

```r
conf_int_mean(Z_cnd$R, Z_cnd$Z)
```

```
## $rhs
## [1] 0.2551248
##
## $lhs
## [1] 0.3080184
```

```r
cat("The variance of Monte Carlo with only Conditional Expectation is ", var(Z_cnd$Z)/Z_cnd$R, "\n")
```

```
## The variance of Monte Carlo with only Conditional Expectation is  0.000182068
```

As only implementing conditional expectation did not yield much of a variance reduction, we implement this in combination with the method of control variates and the method of antithetic sampling.

We start with the method of control variates, where the method in principle is the same as the one implemented in Exercise 1.

A natural choice for the control variate is u, the rate factor, as this will be correlated with the value of p. We will then use the estimator

$$\tilde{p} = \hat{p} + \alpha(\hat{u} - \mathbf{E}[U]).$$

The α is decided by the formula

$$\alpha = -\frac{\text{Cov}(p, U)}{\text{Var}(U)}.$$

We use the sample variance and sample covariance to calculate α, and get

```r
set.seed(203)

CtrlVar <- function(R){
  condMC <- ConditMC(R) #ps sampled from the method of Conditional Expectation
  ps = condMC$Z
  us = condMC$us
  alpha = - cov(ps,us)/var(us) #Calculating the ideal value for alpha
  ptilde = sum(ps)/R + alpha*(sum(us)/R - 0.5)
  return(list("ptilde" = ptilde, "ps" = ps, "us" = us, "alpha" = alpha, "R" = R))
}

Z_ctrlv <- CtrlVar(1000)
cat("This gives us the new estimation for p: ", Z_ctrlv$ptilde, "\n")
```

```
## This gives us the new estimation for p:  0.2909479
```

```r
ps <- Z_ctrlv$ps; us <- Z_ctrlv$us; alpha <- Z_ctrlv$alpha; R <- Z_ctrlv$R
varCtrl <- var(ps)/R + alpha^2*var(us)/R + 2*alpha*cov(ps, us)/R
cat("For this estimator, we get the variance ", varCtrl)
```

```
## For this estimator, we get the variance  9.222061e-05
```

Here, we se that we have get a variance of one order lower magnitude compared to only the Crude Monte Carlo estimator. The variance is calculated by the same formula as the one described in Exercise 1.

2.3

In antithetic sampling, we consider i.i.d pairs $(Z_1, Z_2), (Z_3, Z_4), ...$ instead of i.i.d samples as before. While the pairs are mutually independent, the Z_i inside a pair may now be dependent. Our estimator is still

$$\hat{z}_{\text{ant}} = \sum_{r=1}^{R} Z_i,$$

but the variance is now

$$\text{Var}[\hat{z}_{\text{ant}}] = \frac{\text{Var}[Z]}{R}(1 + \text{Corr}[Z_1, Z_2]),$$

where $\text{Corr}[z_1, Z_2]$ is the correlation coefficient of the variables in one of the pairs. We see that we get variance reduction when each pair is individually negatively correlated, and the correlation coefficient is as small as possible. We know that for a standard uniform random variable U, U and $1-U$ are perfectly negatively correlated, and following from that, that $g(U)$ and $g(1-U)$ are negatively correlated as well, for any function g. We use that our p_r are functions of u, the rate factor. So, instead of sampling R independent u_r, we sample $R/2$ repetitions, and create pairs of rate factors, $(u_r, 1-u_r)$. This gives the estimator

$$\hat{z}_{\text{ant}} = \frac{1}{R}\sum_{r=1}^{R/2} 2 - e^{-\lambda_r^{(1)}(1-S_{19,r}^{(1)})} - e^{-\lambda_r^{(2)}(1-S_{19,r}^{(2)})},$$

where $\lambda_r^{(1)} = 15/(0.5 + u_r)$ and $\lambda_r^{(2)} = 15/(0.5 + 1 - u_r)$, and $S_{19,r}^{(1)}$ and $S_{19,r}^{(2)}$ are calculated from their respective λ_r's. This gives us:

```r
set.seed(204)
AntSamp <- function(R){
  Z = c()
  z1s <- c()
  z2s <- c()
  #us <- c()
  for(r in 1:(R/2)){
    u1 <- runif(1) #Note: we resample the rate at each repetition
    u2 <- 1 - u1
    lamb1 <- 15/(0.5+u1)
    lamb2 <- 15/(0.5+u2)
    #us = c(us,u)
    U <- runif(19)
    S19_1 = -(1/lamb1)*sum(log(U))
    if (S19_1 >= 1){
      Z = c(Z,0)
      z1s = c(z1s, 0)
    }
    else {
      p = 1 - exp(-lamb1*(1 - S19_1))
      Z = c(Z,p)
      z1s = c(z1s, p)
    }
    S19_2 = -(1/lamb2)*sum(log(U))
    if (S19_2 >= 1){
      Z = c(Z,0)
      z2s = c(z2s, 0)
    }
    else {
      p = 1 - exp(-lamb2*(1 - S19_2))
      Z = c(Z,p)
      z2s = c(z2s, p)
    }
  }

  return(list("Z" = Z, "R" = R, "z1s" = z1s, "z2s" = z2s))
}
Z_as <- AntSamp(1000)
var_as <- var(Z_as$Z)/Z_as$R*(1 + cor(Z_as$z1s, Z_as$z2s))
cat("Antithetic sampling gives the estimator ", sum(Z_as$Z)/Z_as$R, "with variance", var_as)
```

```
## Antithetic sampling gives the estimator  0.2798565 with variance 0.0001389562
```

We see that antithetic sampling does not give a variance reduction as big as the control variate for the same $R = 1000$. However, we note that we have only sampled our u_r and $\mathbf{U}_r$ $R/2$ times. This means that we can "afford", in measures of run time, to sample twice as many repetitions:

```r
set.seed(204)
Z_as2K <- AntSamp(2000)
var_as2K <- var(Z_as2K$Z)/Z_as2K$R*(1 + cor(Z_as2K$z1s, Z_as2K$z2s))
cat("With 2000 repetitions, the variance is ", var_as2K)
```

```
## With 2000 repetitions, the variance is  6.800058e-05
```

Now, we observe a substantial variance reduction with antithetic sampling as well. We should note that by choosing to not resample the $\mathbf{U}_r$ for each $S_{19,r}^{(i)}$, meaning that both $S_{19,r}^{(1)}$ and $S_{19,r}^{(2)}$ depend on the same $\mathbf{U}_r$, we do necessarily increase the in-pair correlation coefficient a little. A method to avoid this, would be to sample $\mathbf{U}_r$ twice at each repetition, one for each of $S_{19,r}^{(1)}$ and $S_{19,r}^{(2)}$. I tried this as well, and it did indeed provide a larger variance reduction than the original procedure with 1000 repetitions. However, with this strategy we do no longer get the run-time benefit of only sampling half of the repetitions, and it turned out that our original procedure with 2000 repetitions had far better variance reduction. So this is what is implemented here.

We will now compare the variances for the different methods;

```r
set.seed(205)

cat("Estimated variance for each of our discussed procedures: \n")
```

```
## Estimated variance for each of our discussed procedures:
```

```r
cat("Crude Monte Carlo: ", var_CMC, "\n")
```

```
## Crude Monte Carlo:  0.0001986697
```

```r
cat("Control Variate: ", varCtrl, "\n")
```

```
## Control Variate:  9.222061e-05
```

```r
cat("Antithetic sampling 1000 repetitions", var_as, "\n")
```

```
## Antithetic sampling 1000 repetitions 0.0001389562
```

```r
cat("Antithetic sampling 2000 repetitions", var_as2K, "\n")
```

```
## Antithetic sampling 2000 repetitions 6.800058e-05
```

We see that both the Control Variate method and the Anithetic Sampling with run-time taken into account, yield substantial variance reduction. Of these two, the Anithetic Sampling seems to be slightly more efficient, but this difference might not be significant.

Exercise 3

In the stratification method, we want to divide the sample space Ω into distinct strata, $\Omega_1, ..., \Omega_S$ for some S. A natural way to perform this division is to apply some restriction to the variable X if the target is the function $h(X)$. In our case, we want to estimate the expectation, $\mathbf{E}[h(Z_1, ..., Z_n)]$, of a function of n independent standard normal varibles, that is increasing for each of its coordinates. One method of defining the strata is to part each Z_i into some $\alpha\%$ intervals, which would be easily accessible since the Z_i are standard normals. We could then form an n-dimensional grid that defines the different strata. The quite obvious problem with this approach however, is that the number of strata would be very rapidly increasing with n, which would make it unpractical to implement. Instead, we will consider the sum

$$W = \sum_{i=1}^{n} a_i Z_i$$

where the a_i are some non-negative constants. Since both this sum and $h(Z_1, ..., Z_n)$ can be seen as increasing functions of $(Z_1, ..., Z_n)$, we have that

$$W(\mathbf{Z}^{(1)}) > W(\mathbf{Z}^{(2)}) \quad \Leftrightarrow \quad h(\mathbf{Z}^{(1)}) > h(\mathbf{Z}^{(2)}).$$

This means that the set of $\mathbf{Z}$ that create a partition of W will necessarily also create a partition of h, and that the probability of being in each partition of W or h will be the same. We now look at the distribution of W. Since the Z_i are independent standard normals, the sum of these will also have a normal distribution, with

$$\mathbf{E}[W] = \mathbf{E}[\sum_{i=1}^{n} a_i Z_i] = \sum_{i=1}^{n} a_i \mathbf{E}[Z_i] = 0$$

$$\mathrm{Var}[W] = \mathrm{Var}[\sum_{i=1}^{n} a_i Z_i] = \sum_{i=1}^{n} a_i^2 \mathrm{Var}[Z_i] = \sum_{i=1}^{n} a_i^2.$$

So, we get that

$$W \sim \mathbf{N}(0, \sum_{i=1}^{n} a_i^2).$$

We use this to make strata, each with probability $\alpha\%$ of occuring:

$$\Omega_s := \{Z_{s\alpha}\sqrt{\sum_{i=1}^{n} a_i^2} \leq W < Z_{(s+1)\alpha}\sqrt{\sum_{i=1}^{n} a_i^2}\}$$

for $s = 1, ..., \frac{100\%}{\alpha} - 1$, where $Z_{s\alpha}$ are the $s \cdot \alpha\%$ standard normal quantile values. We can now sample from the different strata like this; W is sampeled by sampling all Z_i by inversion, in total R times. For each W_r we check which strate it belongs to, and then add the corresponding $h(\mathbf{Z}_r)$ to that strata, $\mathbf{Z}_r$ being the same $\mathbf{Z}_r$ that provided the W_r. Since α, the probability of being in a paricular strata Ω_s, is the same for all strata, each R_s should approximately fulfull the criterion of proportional allocation of R_s, namely that $p_s = R_s/R$. This should lead to variance reduction, at least when R is sufficiently large. After sorting each sample $h(\mathbf{Z}_r)$ into different strata, while keeping track of the corresponding R_s, we can perform the stratification procedure as usual:

1. for each strata:

$$\hat{z}_s = \frac{1}{R_s}\sum_{r=1}^{R_s} h(Z_{s,r})$$

2. We can then find the stratified estimator:

$$\hat{z}_{strat} = \sum_{s=1}^{S} \alpha \hat{z}_s.$$

Note: I do realise that I did not really use the hint, and that it is probably some other way to do this as well, perhaps should I have found a better way to precicely deifine the R_s, but I do think my method might still work.

Exercise 4

4.1

Using the Crude Monte Carlo method, we get:

```r
set.seed(130)

R = 1000
z = c()
for(r in 1:R){
  U <- runif(4)
  X <- c(1:4)
  zr <- sapply(X, function(x) -log(U[x])*(x + 2))
  s = sum(zr)
  if(s > 62){
    z = c(z,s)
    cat("Non-zero value!")
  }
  else{
    z = c(z,0)
  }
}

zhatCMC = sum(z)/R
#95% Confidence interval:
errCMC = 1.96*sd(z)/sqrt(R)
rhsCMC = zhatCMC - errCMC
lhsCMC = zhatCMC + errCMC

cat("We get the confidence interval ", rhsCMC, ", ", lhsCMC)
```

```
## We get the confidence interval  0 ,  0
```

We see that the Crude Monte Carlo method with 1000 repetition in this case actually gives no non-zero samples. This makes it difficult to make a reliable confidence interval, as we have no observations in the target area.

4.2

In importance sampling, we use that $z = \mathbf{E}[h(X)] = \mathbf{E}_g[h(X)\frac{f(X)}{g(X)}]$ to estimate an r.v. $z = \mathbf{E}[h(\mathbf{X})]$ by $\hat{z}_{imp} = \frac{1}{R}\sum_{r=1}^{R} h(\mathbf{X}_r)L(\mathbf{X}_r)$, where $L(\mathbf{X}_r) = f(\mathbf{X}_r)/g(\mathbf{X}_r)$. Here, $f(\cdot)$ is the original distribution of the r.v. $\mathbf{X}$ in our Z, and the $\mathbf{X}_r$ are sampled as if they were from the distribution $g(\cdot)$. $g(\cdot)$ is usually a distribution that is similar to $f(\cdot)$, but more heavily weighted towards a target area. In our case, we have

$$z = \mathbf{E}[h(x_1, x_2, x_3, x_4)] = \mathbf{E}[S\mathbf{1}\{S > 62\}]; \quad S = x_1 + x_2 + x_3 + x_4$$

We will start by finding the joint distribution of $x_i; i = 1, ..., 4$. Since they are independent, we get that the cumulative distribution function is

$$F_{X_{all}}(\mathbf{X}) = \Pi_{i=1}^{4}(1 - e^{-\lambda_i x_i}),$$

since

$$F_{X_i} = 1 - e^{-\lambda_i x_i}$$

is the cumulative distribtion of exponential variables with mean $1/\lambda_i$. We will then find the joint probability density function by derivating this:

$$f_{X_{all}}(\mathbf{X}) = \frac{\partial^4 F_{X_{all}}(\mathbf{X})}{\partial x_1 \partial x_2 \partial x_3 \partial x_4}$$
$$= \frac{\partial^4}{\partial x_1 \partial x_2 \partial x_3 \partial x_4}\Pi_{i=1}^{4}(1 - e^{-\lambda_i x_i})$$
$$= \frac{\partial^3}{\partial x_1 \partial x_2 \partial x_3}\Pi_{i=1}^{3}(1 - e^{-\lambda_i x_i})\frac{\partial}{\partial x_4}(1 - e^{-\lambda_4 x_4}) = \lambda_4 e^{-\lambda_4 x_4}\frac{\partial^3}{\partial x_1 \partial x_2 \partial x_3}\Pi_{i=1}^{3}(1 - e^{-\lambda_i x_i}) = \dots$$
$$= \lambda_1\lambda_2\lambda_3\lambda_4 e^{-(\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4)}$$

With this distribution, the expectation of S is

$$\mathbf{E}[S] = \int_0^\infty\int_0^\infty\int_0^\infty\int_0^\infty (\sum_{i=1}^{4} x_i)\lambda_1\lambda_2\lambda_3\lambda_4 e^{-(\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4)} dx_1 dx_2 dx_3 dx_4 = \dots = \frac{1}{\lambda_1} + \frac{1}{\lambda_2} + \frac{1}{\lambda_3} + \frac{1}{\lambda_4}.$$

A natural choice for g is then on the same form, but yielding an s with expectation 62, our critical value. So, we choose the importance distribution

$$g(\mathbf{X}) = \lambda_g^4 e^{-\lambda_g(x_1 + x_2 + x_3 + x_4)},$$

where

$$4 \cdot \frac{1}{\lambda_g} = 62 \Leftrightarrow \lambda_g = \frac{4}{62},$$

which will give

$$\mathbf{E}[S|\mathbf{X} \text{ from } g(\cdot)] = 62.$$

4.3

The implementation of this importance sampling is the following:

```r
R = 1000
z <- c()

for (r in 1:R){
  U <- runif(4)
  X <- sapply(U, function(x) -log(x)*31/2)
  s = sum(X)
  f = (1/360)*exp(-(X[1]/3 + X[2]/4 + X[3]/5 + X[4]/6))
  g = (2/31)^4*exp(-2/31*s)
  if(s > 62){
    z = c(z,s*f/g)
  }
  else{
    z = c(z,0)
  }
}

zhat_is = sum(z)/R
zhat_is
```

```
## [1] 0.07180488
```

```r
#95% Confidence interval:
err = 1.96*sd(z)/sqrt(R)
rhs = zhat_is - err
lhs = zhat_is + err

rhs; lhs
```

```
## [1] 0.0493295
```

```
## [1] 0.09428026
```

We see that this method does produce a much more precice confidence interval for z.

4.4

In the Crude Monte Carlo procedure, we estimate z by sampling values from $f(\cdot)$ and calculate the probability from the rate of which it satisfies the criterion $S > 62$, which, since the probability of the rate being fulfulled is very low is very rarely. In importance sampling, we instead sample $\mathbf{X}$ from a distribution which more often provides values close to the critical value, and then find the probability that this would have been observed. We can then get a much more precise estimation, as each sample is closer to the actual value of z, and we do not have to deal with too many null samples.