Assignment 2

Bjørn Christian Weinbach

12th October, 2020

Clear R environment

```
rm(list = ls())
```

Problem 1

Consider the integral

$$\int_{-1}^{1}\int_{-1}^{1} 1_D(x,y)dxdy$$

Where $1_D(x,y)$ is the indicator function defined so that

$$1_D(x,y) = \begin{cases} 1 & \text{if} \quad x^2 + y^2 \leq 1, 0 \quad \text{otherwise.} \end{cases}$$

As a crude first attempt, consider the estimator

$$\theta_{CMC} = \frac{4}{N}\sum_{i=1}^{n} 1_D(X_i, Y_i)$$

A.) Argue for why θ_{CMC} is a monte carlo estimator for the integral above.

According to this wikipedia article which was accessed the 9th October, 2020, the monte carlo estimate for a multidimensional definite integral

$$I = \int_{\Omega} f(\bar{\boldsymbol{x}})d\bar{\boldsymbol{x}}$$

where Ω is a subset of R^m, has volume

$$V = \int_{\Omega} d\bar{\boldsymbol{x}}$$

The naive Monte Carlo approach is to sample points uniformly on Ω given N uniform samples,

$$\bar{\boldsymbol{x}}_{\boldsymbol{1}}, \ldots, \bar{\boldsymbol{x}}_{\boldsymbol{n}} \in \Omega,$$

I can be approximated by

$$I \approx \Omega_N \equiv V\frac{1}{N}\sum_{i=1}^{N} f(\bar{\boldsymbol{x}}_{\boldsymbol{i}})$$

Which is true due to the law of large numbers.

In our case, which is also very similar to the example on the wikipedia article for monte carlo integration, $\Omega = [-1, 1] \times [-1, 1]$ with $V = \int_{-1}^{1} \int_{-1}^{1} dxdy = 4$ which gives the following crude way to estimate I

$$I = \frac{4}{N} \sum_{i=1}^{N} 1_D(X_i, Y_i)$$

Which is the proposed estimator θ_{CMC}.

A.) Show that $1_D(X_i, Y_i)$ has a bernoulli distribution with $p = \frac{\pi}{4}$

The function returns success or failure, and is therefore has aa potential ' bernoulli distribtion. To calculate $P(X^2 + Y^2 \leq 1)$ we need to calculate the

c) Implementation of monte carlo estimate of θ_{CMC} with $N = 1000$

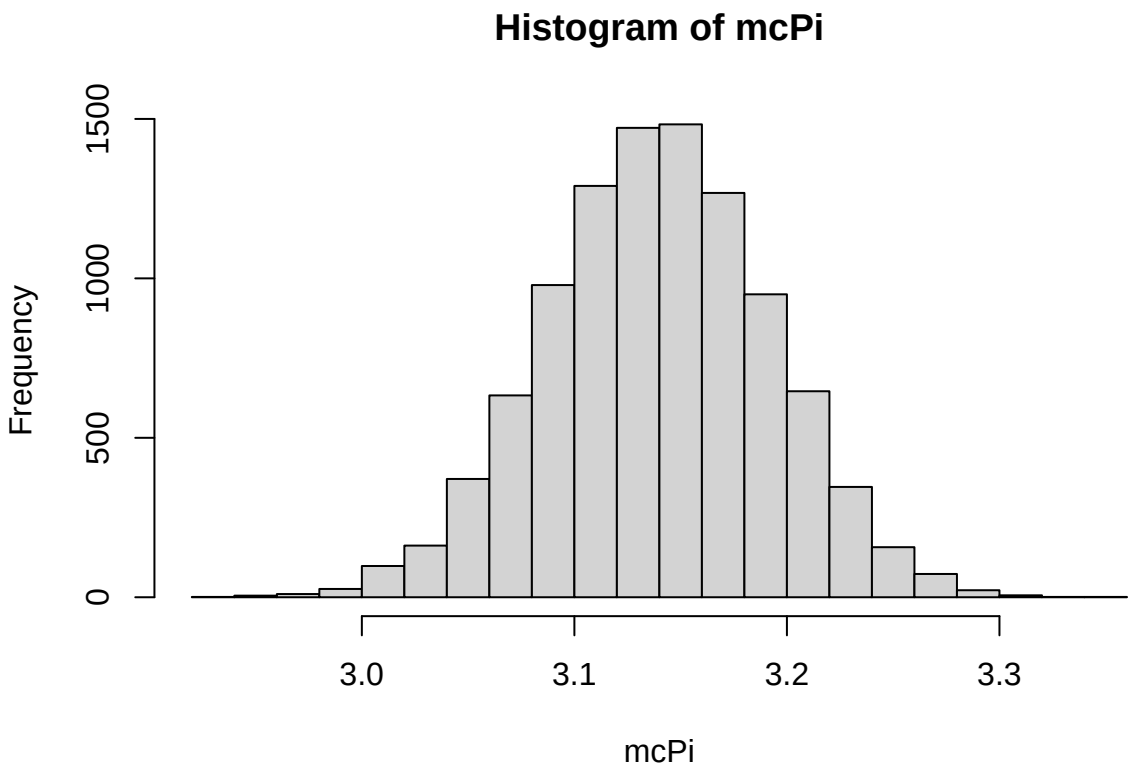
```
indicator1 <- function (x, y) {          # Indicator function for unit circle
  return((x^2+y^2 <= 1))
}

mcEstimatePi <- function (Sims) {     # function for monte carlo estimate
  mcPi <- numeric(Sims)
  for (i in 1:Sims) {
    N <- 1000                           # Number of samples
    x <- runif(N, -1, 1)                # x values from uniform
    y <- runif(N, -1, 1)                # y values from uniform
    mcPi[i] <- (4/N)*sum(indicator1(x, y)) # return monte carlo estimate
  }
  return(mcPi)
}

Sims <- 10000                           # Simulations of pi
mcPi <- mcEstimatePi(Sims)        # Function call
summary(mcPi)                           # Summary stats for MC estimate
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.936   3.104   3.140   3.141   3.176   3.352
```

```
hist(mcPi)                              # Histogram
```




```
mean(mcPi)
```

```
## [1] 3.141076
```

```
var(mcPi)
```

```
## [1] 0.002782377
```

We see that the distribution is approxamately normally distrobuted with sample mean at approx $Mean = 3.141$ and sample variance at approx $Variance = 0.00260$. This is du to the central limit theorem because our estimate of pi is based on a sum of several samples.

d.) Calculate probability of correctly estimating to two decimal places

```
mean(mcPi < 3.15) - mean(mcPi <= 3.14)
```

```
## [1] 0.0604
```

e.) Introducing antithetic variables

The assignment proposes two antithetic variables $V = a + b - X = -X$ and $W = -Y$

These will not reduce the monte carlo variance since X and Y are independent uniform random variables and due to their independence there is no negative correlation gained by just flipping the sign of both variables from X and Y to
$-X$ and $-Y$.

```
x <- runif(1000, -1, 1)
y <- runif(1000, -1, 1)
cov(-x, -y)
```

```
## [1] -0.005408746
```

f.) Let's see if our variance is reduced by doing exercise c. with the new variables.

```
mcEstimatePi <- function (Sims) {     # function for monte carlo estimate
  mcPi <- numeric(Sims)
  for (i in 1:Sims) {
    N <- 1000                         # Number of samples
    x <- runif(N, -1, 1)              # x values from uniform
    y <- runif(N, -1, 1)              # y values from uniform
    mcPi[i] <- (4/N)*sum(indicator1(-x, -y)) # mcEstimate with new variables
  }
  return(mcPi)
}

Sims <- 10000                         # Simulations of pi
mcPi <- mcEstimatePi(Sims)            # Function call
summary(mcPi)                         # Summary stats for MC estimate
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.952   3.108   3.144   3.142   3.176   3.340
```

```
mean(mcPi)
```

```
## [1] 3.141616
```

```
var(mcPi)
```

```
## [1] 0.002688787
```

g.) Check if shift function reduces variance

```
shift <- function(u) {                # Introducing shift function
  return(((u+2.0) %% 2.0) - 1.0)
}

mcEstimatePi <- function (Sims) {     # function for monte carlo estimate
  mcPi <- numeric(Sims)
  for (i in 1:Sims) {
    N <- 1000                         # Number of samples
    x <- runif(N, -1, 1)              # x values from uniform
    y <- runif(N, -1, 1)              # y values from uniform
    sx <- shift(x)
    sy <- shift(y)
    mcPi[i] <- (4/N)*sum(indicator1(sx, sy)) # mcEstimate with new variables
  }
  return(mcPi)
}

Sims <- 10000                         # Simulations of pi
mcPi <- mcEstimatePi(Sims)            # Function call
summary(mcPi)                         # Summary stats for MC estimate
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.920   3.108   3.140   3.141   3.176   3.344
```

```
mean(mcPi)
```

```
## [1] 3.141421
```

```r
var(mcPi)
```

```
## [1] 0.002631304
```

h.) Use important sampling

```r
f <- function (x, y, sigma) {
  return((1/(2*pi*sigma^2))*exp(-(x^2)/(2*sigma^2))*exp(-(y^2)/(2*sigma^2)))
}

mcEstimatePi <- function (Sims, sigma) {  # function for monte carlo estimate
  mcPi <- numeric(Sims)
  for (i in 1:Sims) {
    N <- 1000                                  # Number of samples
    x <- rnorm(N, 0, sigma^2)                  # x values from uniform
    y <- rnorm(N, 0, sigma^2)                  # y values from uniform
    mcPi[i] <- mean(indicator1(x, y) / f(x, y, sigma))
  }
  return(mcPi)
}
```

```r
mcPi <- mcEstimatePi(10000, 0.3)
summary(mcPi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6146  0.6201  0.6214  0.6214  0.6227  0.6297
```

Problem 2

Define $\lambda(t)$ in R

```r
# Specify the intensity function for storms
lambdastorm <- function(t) {
 (297 / 10)*(1 + cos(2*pi*(t + (1/10)))) * (1 - (exp(-t/10)/2)) + 3/5
}

lambdastorm(0.5)
```

```
## [1] 3.574416
```

a.) Calculating number of storms, expected value and variance

According to Rizzo on page 103. A poisson proccess with a intensity function $\lambda(t)$ has the property that the number of events $N(t)$ in interval $[0, t]$ has the poisson distribution with mean

$$E[N(t)] = \int_0^t \lambda(y)dy$$

Which in our case gives

```r
integrate(lambdastorm, 0, 1)
```

```
## 16.29763 with absolute error < 1.8e-13
```

Which we have confirmed by the simulation above which has a simulated mean of approximately 16.3

Let's find the expected number of storms in 2025 by calculating the integral

```r
integrate(lambdastorm, 5, 6)
```

```
## 21.80713 with absolute error < 2.4e-13
```

And let's find the expected value and standard deviation of storms in 2020 and 2021 combined

Expected value is calculated using the integral below

```r
integrate(lambdastorm, 0, 2)
```

```
## 33.92776 with absolute error < 2.3e-08
```

Which means the number of events in 2020 and 2021 combined is poisson distrubuted with $\lambda = 39.72776$. The variance of a poisson distribution is $Var(X) = \lambda$ (according to Rizzo on page 44).

$$SD(X) = \sqrt{Var(X)} = \sqrt{33.92776} = 5.824754$$

b.) Find smallest possible λ_{max} for all $\lambda(t), \quad t \geq 0$

The function $\lambda(t)$ does not have a global maximum because it is modeled with a increasing winter intensity due to climate change that does not stop increasing. Solving for $\frac{d}{dt}\lambda(t) = 0$ gives an infinite number of potential maximum or minimum points and there is no λ_{max} for all $\lambda(t)$ values.

c.) Validate previous points by simulation

simtNHPP borrowed from lectures on stochastic processes

```r
# Function for simulating arrival times for a NHPP between a and b using thinning
simtNHPP <- function(a,b,lambdamax,lambdafunc){
  # Simple check that a not too small lambdamax is set
  if(max(lambdafunc(seq(a,b,length.out = 100)))>lambdamax)
    stop("lambdamax is smaller than max of the lambdafunction")
  # First simulate HPP with intensity lambdamax on a to b
  expectednumber <- (b-a)*lambdamax
  Nsim <- 3*expectednumber  # Simulate more than the expected number to be certain to exceed stoptime
  timesbetween <- rexp(Nsim,lambdamax) # Simulate interarrival times
  timesto <- a+cumsum(timesbetween)   # Calculate arrival times starting at a
  timesto <- timesto[timesto<b] # Dischard the times larger than b
  Nevents <- length(timesto) # Count the number of events
  # Next do the thinning. Only keep the times where u<lambda(s)/lambdamax
  U <- runif(Nevents)
  timesto <- timesto[U<lambdafunc(timesto)/lambdamax]
  timesto  # Return the remaining times
}

Nsim <- 1000
a <- 0
b <- 1
NHPPnumbers <- vector(length=Nsim)
for(i in 1:Nsim) {
  NHPPnumbers[i] <- length(simtNHPP(a=a,b=b,
                           lambdamax=max(lambdastorm(seq(a, b, 0.01))),
                           lambdafunc=lambdastorm))
}

# Exepcted number of storms in 2020
mean(NHPPnumbers)
```

```
## [1] 16.274
```

```r
Nsim <- 1000
a <- 5
b <- 6
NHPPnumbers <- vector(length=Nsim)
for(i in 1:Nsim) {
  NHPPnumbers[i] <- length(simtNHPP(a=a, b=b,
                           lambdamax=max(lambdastorm(seq(a, b, 0.01))),
                           lambdafunc=lambdastorm))
}

# Exepcted number of storms in 2025
mean(NHPPnumbers)
```

```
## [1] 21.807
```

```r
Nsim <- 1000
a <- 0
b <- 2
NHPPnumbers <- vector(length=Nsim)
for(i in 1:Nsim) {
  NHPPnumbers[i] <- length(simtNHPP(a=a, b=b,
                           lambdamax=max(lambdastorm(seq(a, b, 0.01))),
                           lambdafunc=lambdastorm))
}

# Expected number of storms in 2020 and 2021
mean(NHPPnumbers)
```

```
## [1] 33.661
```

```r
# Variance of number of storms in 2020 and 2021
var(NHPPnumbers)
```

```
## [1] 29.55964
```

```r
# Standard deviation of number of storms in 2020 and 2021
sd(NHPPnumbers)
```

```
## [1] 5.436878
```