Assignment I - CompStat2023

Aldo Giovanni e Giacomo

Anuja Saira Abraham 5204982, Alessia Marzotti 5108443, Miriam Mercuri 5207057

Set up

```
library(ggplot2)
library(patchwork)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(mixtools)
```

```
## mixtools package, version 2.0.0, Released 2022-12-04
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

Simulation problem: `RISIKO!`

Exeercise 1

Point a

Compute $P[\max(X_1, X_2) > Y_1]$.

We know that

$$Z = max(X_1, X_2) \in (1, 2, 3, 4, 5, 6)$$

and that

$$P(Z = z) = \frac{(2(z-1)+1)}{36}$$

. In other word we want to find $P(Z > y) = 1 - P(Z \leq y)$. $P(Z \leq y) =$
$P(Z = 1 \cap y = 1) +$
$P(Z = 1 \cap y = 2) + P(Z = 2 \cap y = 2) +$

$P(Z=1 \cap y=3)+P(Z=2 \cap y=3)+P(Z=3 \cap y=3)+$
$P(Z=1 \cap y=4)+P(Z=2 \cap y=4)+P(Z=3 \cap y=4)+P(Z=4 \cap y=4)+$
$P(Z=1 \cap y=5)+P(Z=2 \cap y=5)+P(Z=3 \cap y=5)+P(Z=4 \cap y=5)+P(Z=5 \cap y=5)+$
$P(Z=1 \cap y=6)+P(Z=2 \cap y=6)+P(Z=3 \cap y=6)+P(Z=4 \cap y=6)+P(Z=5 \cap y=6)+P(Z=6 \cap y=6)$

So we have $P(Z \leq y)=6\frac{1}{36}\frac{1}{6}+5\frac{3}{36}\frac{1}{6}+4\frac{5}{36}\frac{1}{6}+3\frac{7}{36}\frac{1}{6}+2\frac{9}{36}\frac{1}{6}+\frac{11}{36}\frac{1}{6} \approx 0,421$ and

$$P(Z>y)=1-0,421=0,579$$

Point b

Here there are some code to simulate a generic `Risiko!` game for different values of competing units. Starting with a function, *combat_round*, that take as arguments: *att_units*, the number of attacking units, *def_units*, the number of defending units and *sim*, the number of scenarios to simulate.

```
set.seed(123)
combat_round <- function(att_units,def_units,sim=10000) {
  Results = rep(NA,sim)
  AS<-att_units
  DS<-def_units
  for(i in 1:sim){
    while(def_units>0 & att_units>0){
      Dnum <- sort(sample(1:6, min(def_units,3),replace = TRUE),decreasing = T)
      Anum <- sort(sample(1:6, min(att_units,3),replace = TRUE),decreasing = T)
      for (j in 1:min(length(Dnum),length(Anum))){
        if(Anum[j]>Dnum[j]){
          def_units<-def_units-1
        }
        else{
          att_units<-att_units-1
        }
      }
    }
    Results[i]<- ifelse(att_units>0,1,0)
    att_units<-AS
    def_units<-DS
  }
  return(mean(Results))
}
```

Point c

Now we replicate the results a sufficient amount of time equal to 10 to estimate the probability that the attacker has to win the battle for different values of *att_units* and *def_units*.

```
prob_att_win <- function(){
  res<-sapply(1:10, function(x) {
  sapply(1:10, function(y) {
    combat_round(y,x,1000)
    })
  })
  return(res)
}
```

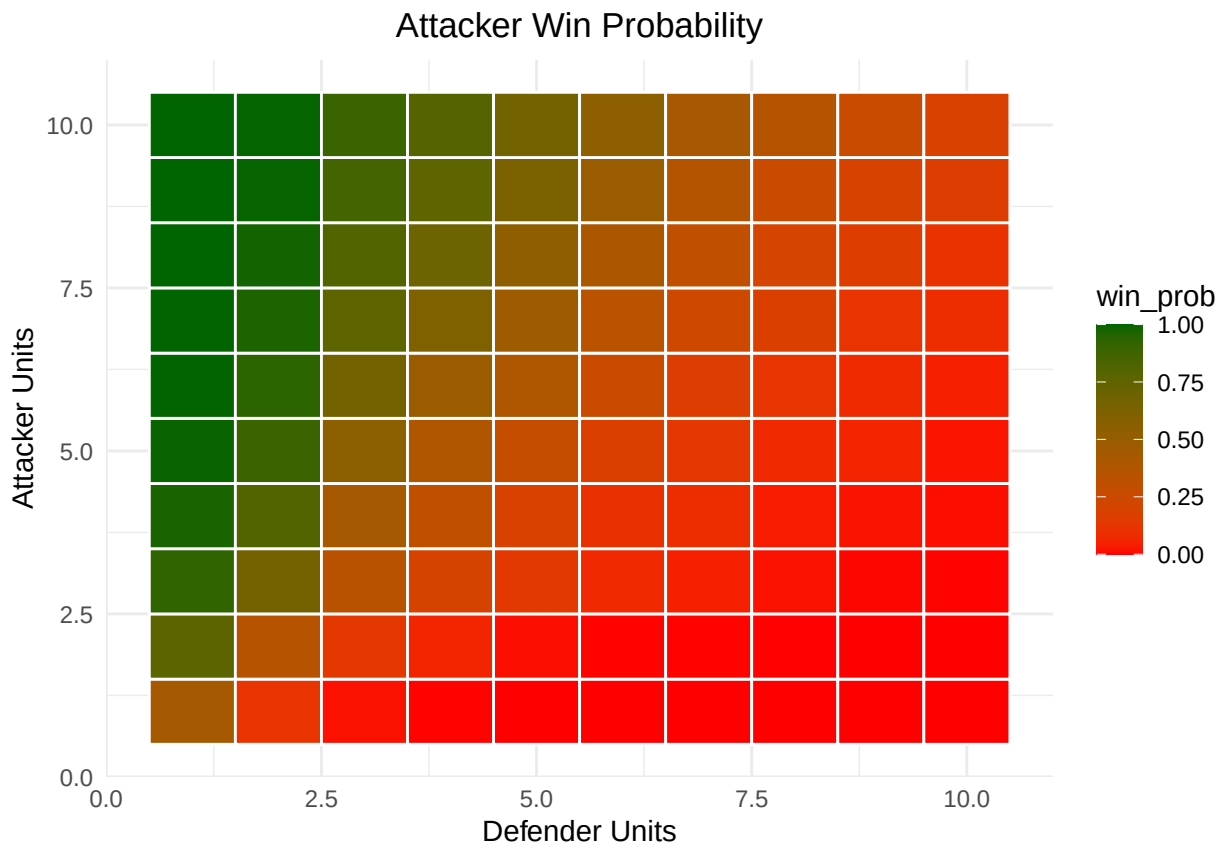
```
att_prob<- data.frame(
  attacker_unit<- rep(1:10,10),
  defender_unit<- rep(1:10,each= 10),
  win_prob<- as.vector(prob_att_win())
)
```

Point d

Finally, we display the results in a plot.

```
ggplot(att_prob,aes(x=defender_unit,y=attacker_unit,fill=win_prob))+
  geom_tile()+
  scale_fill_gradient(low="red",high="darkgreen")+
  labs(title = "Attacker Win Probability",x="Defender Units",y="Attacker Units")+
  geom_tile(color = "white",lwd = 0.5,linetype = 1)+
  theme_minimal()+
  theme(plot.title = element_text(hjust = 0.5))
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



The probability we obtain when *att_units* = 2 and *def_units* = 1 in the *combat_round* function is different from the one computed analytically in point a.

```
print(combat_round(def_units=1,att_units=2,sim=10000))
```

```
## [1] 0.7561
```

This discrepancy arises due to a difference in the calculations. In point a, we specifically calculated the probability of the attacker winning the battle in the first round. However, in point d, we calculate the probability of the attacker winning the battle overall, without necessarily considering it happening in the first round alone.

Monte Carlo simulations I

Point a

An old and naive algorithm for the generation of Normally distributed random numbers is the following:
1. Generate independent variables as $U_1, ..., U_{12} \sim U[-\frac{1}{2}, \frac{1}{2}]$
2. Set $Z = \sum_{i=1}^{12} U_i$
The rational here is that 12 realizations are usually enough to exploit the CLT.

We want to show analytically that $E(X) = 0$ and $V(Z) = 1$.

We know that

$$E(X) = \int_a^b \frac{x}{b-a} dx = \frac{b^2 - a^2}{2(b-a)}$$

Reference: Wikipedia

So $E(Z) = E(\sum_{i=1}^{12} U_i)$
$= \sum_{i=1}^{12} E(U_i)$ independent
$= 12E(U_i)$ identically distributed
$12\frac{\frac{1}{2}^2 - (-\frac{1}{2})^2}{2(\frac{1}{2} - (-\frac{1}{2}))} = 0$

And

$$V(Z) = E[(X - E(X))^2] = \int_a^b (x - \frac{a+b}{2})^2 \frac{dx}{b-a} = \frac{(b-a)^2}{12}$$

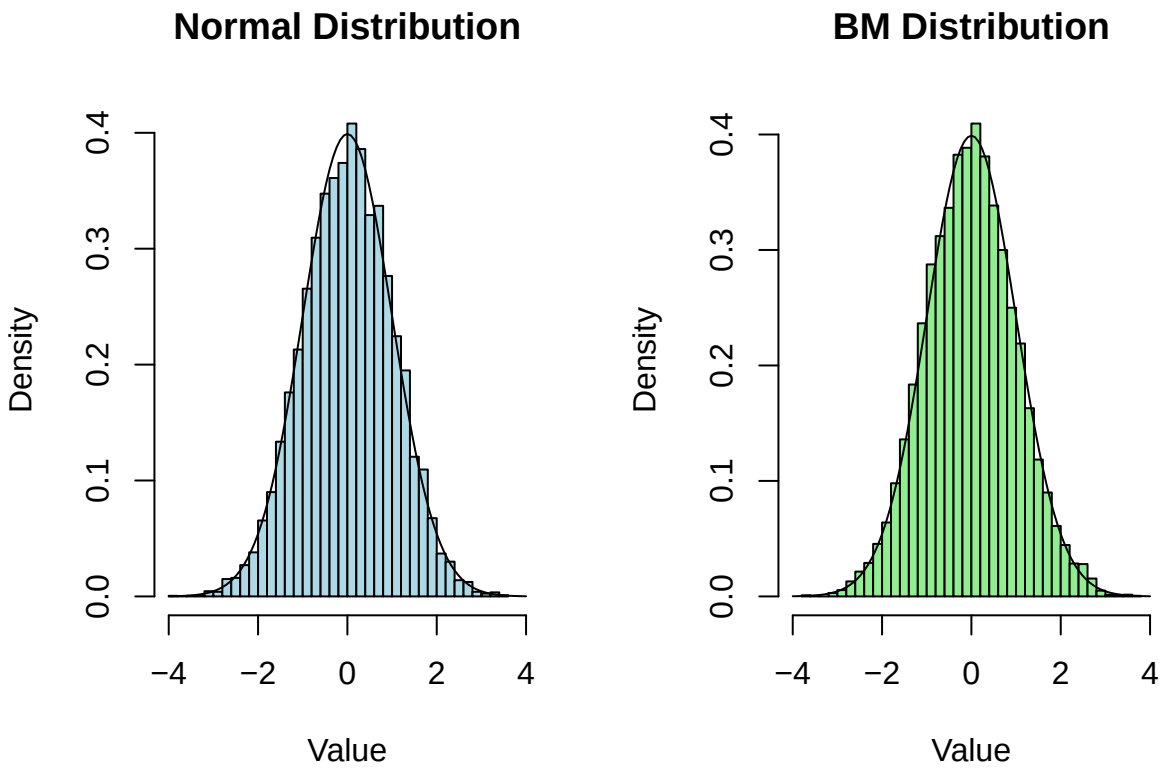
Reference: Wikipedia

So $V(Z) = V(\sum_{i=1}^{12} U_i)$
$= \sum_{i=1}^{12} V(U_i)$ variable independent
$= 12V(U_i)$ U_i identically distributed
$= 12\frac{(\frac{1}{2} - (-\frac{1}{2}))^2}{12}$
$= 12\frac{(\frac{1}{2} + \frac{1}{2})^2}{12} = 1$

Point b

```
#Using histograms to compare the above Normal generator with the Box-Muller algorithm

set.seed(13)
```

```r
normal_functn_gen = function(sim){
  n<-12
  Uz<-runif(sim*n,min = -0.5,max = 0.5)
  Uz<-matrix(Uz,nrow=N,ncol=n)
  Z<-apply(Uz,1,sum)
  return(Z)
}

N <- 10000

gen_norm<-normal_functn_gen(sim=10000)

U1 <- runif(N)
U2 <- runif(N)
X1 <- sqrt( -2*log(U1) )*cos(2*pi*U2)

par(mfrow=c(1,2))
hist(gen_norm, main="Normal Distribution", xlab="Value",col="lightblue" ,breaks=30,freq = F,xlim=c(-4,4
curve(dnorm(x,0,1),add=T)
hist(X1, main="BM Distribution", xlab="Value",col="lightgreen" ,breaks=30,freq = F,xlim=c(-4,4))
curve(dnorm(x,0,1),add=T)
```

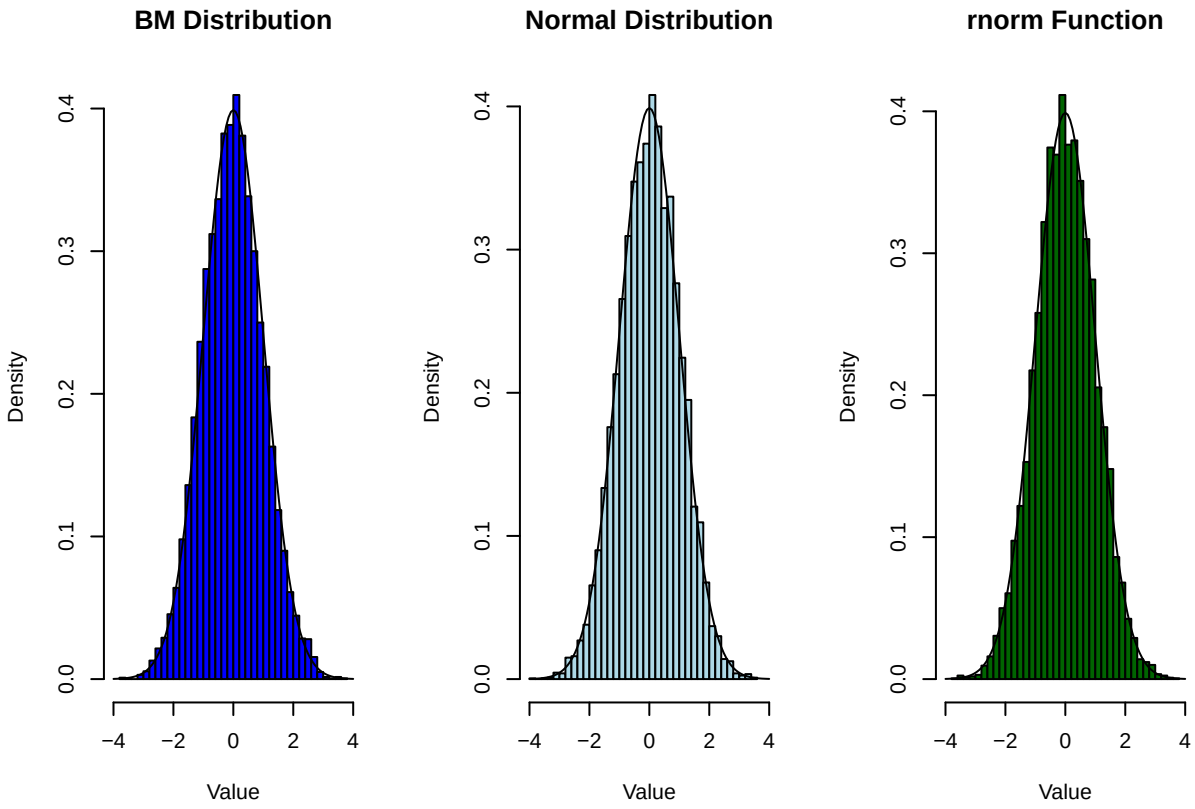


The histograms and curves indicate that both the Normal generator and the Box-Muller algorithm produce samples that closely follow a standard normal distribution and the tail probabilities for extreme values like 3 are expected to be very small in both cases.

Point c

Compare both of the generators above with rnorm shows that is a normal distribution that looks almost similar.

```
normgen <- rnorm(10000)
par(mfrow=c(1,3))
hist(X1, main="BM Distribution", xlab="Value",col="blue" ,breaks=30,freq = F,xlim=c(-4,4))
curve(dnorm(x,0,1),add=T)
hist(gen_norm, main="Normal Distribution", xlab="Value",col="lightblue" ,breaks=30,freq = F,xlim=c(-4,4
curve(dnorm(x,0,1),add=T)
hist(normgen,xlab="Value",col="darkgreen" ,breaks=30,freq = F,xlim=c(-4,4), main = "rnorm Function")
curve(dnorm(x,0,1),add=T)
```



Point d

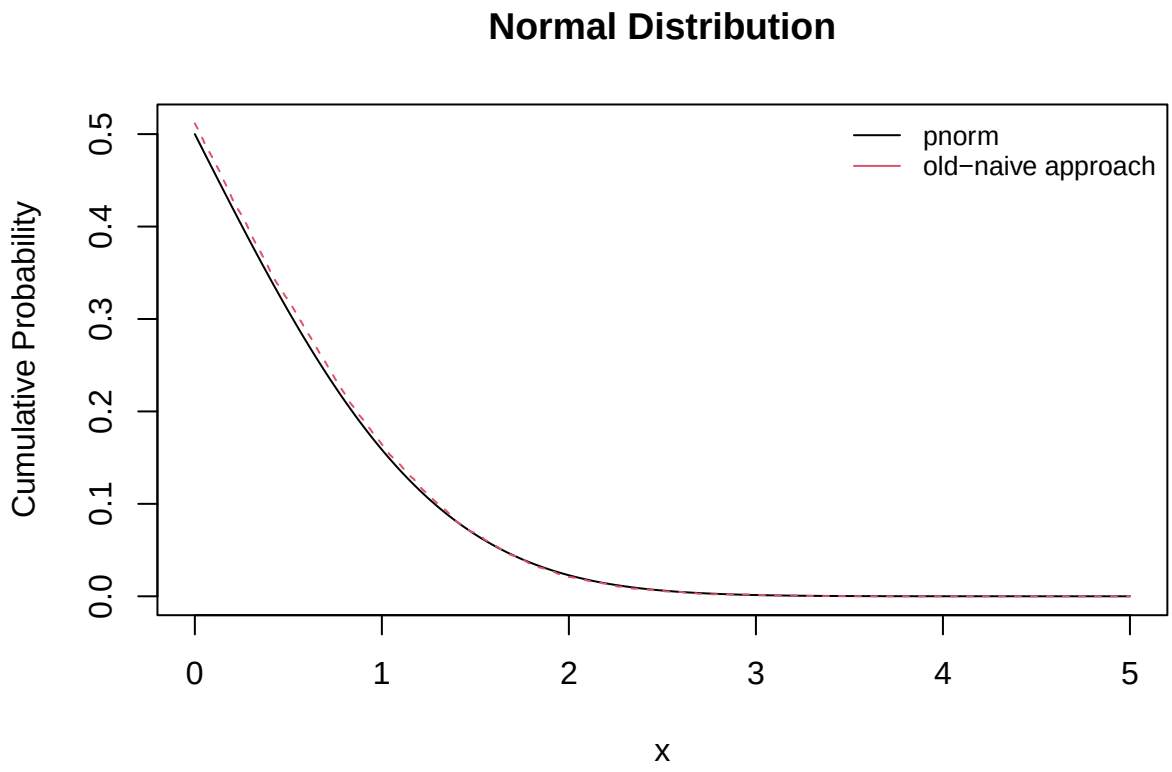
The below code estimate P (Z >= θ) for $\theta \in (0, 5)$

```
upper_tail_prob<- function(gen_norm,theta){
  prob=sum(gen_norm>theta)/length(gen_norm)
  return(prob)
}
x <- seq(0, 5, by = 0.01)
R<- sapply(seq(0, 5, by = 0.01), function(x) upper_tail_prob(gen_norm,x))
# Calculate cumulative probabilities using pnorm
prob <- 1-pnorm(x,0,1)
```

Point e

The below graph shows the variation of estimated probability and actual probability.For larger θ the estimate and truth is equal.

```
matplot(x, cbind(prob,R),
        type = "l",
        xlab = "x",
        ylab = "Cumulative Probability",
        main = "Normal Distribution")
legend("topright", legend = c("pnorm", "old-naive approach"),
       col = 1:3, lty = 1, bty = "n", cex = 0.8)
```



Monte Carlo simulations II

Point a

The Pareto distribution is defined by a density $f(x;\gamma) = \gamma x^{-(\gamma+1)}$ over $(1;+\infty)$, with $\gamma > 0$.

It can be generated as the $-\frac{1}{\gamma}$ power of a uniform random variable.

Cumulative distribution function of Pareto distribution

$$\int_1^x \gamma z^{-(\gamma+1)} dz = \gamma \int_1^x z^{-1-\gamma} dz = -[x^{-\gamma} - 1^{-\gamma}] = -x^{-\gamma} + 1 = 1 - (\frac{1}{x})^{\gamma}$$

Reference: Wikipedia

We will use the following theorem: if $X \sim F(x)$ then $U = F(x) \sim U(0,1)$
$F(X) = 1 - (\frac{1}{x})^{\gamma} = U$
$(1-U)^{-\frac{1}{\gamma}} = (x^{\gamma})^{-\frac{1}{\gamma}}$
$x = (1-U)^{-\frac{1}{\gamma}} = U^{-\frac{1}{\gamma}}$

Pont b

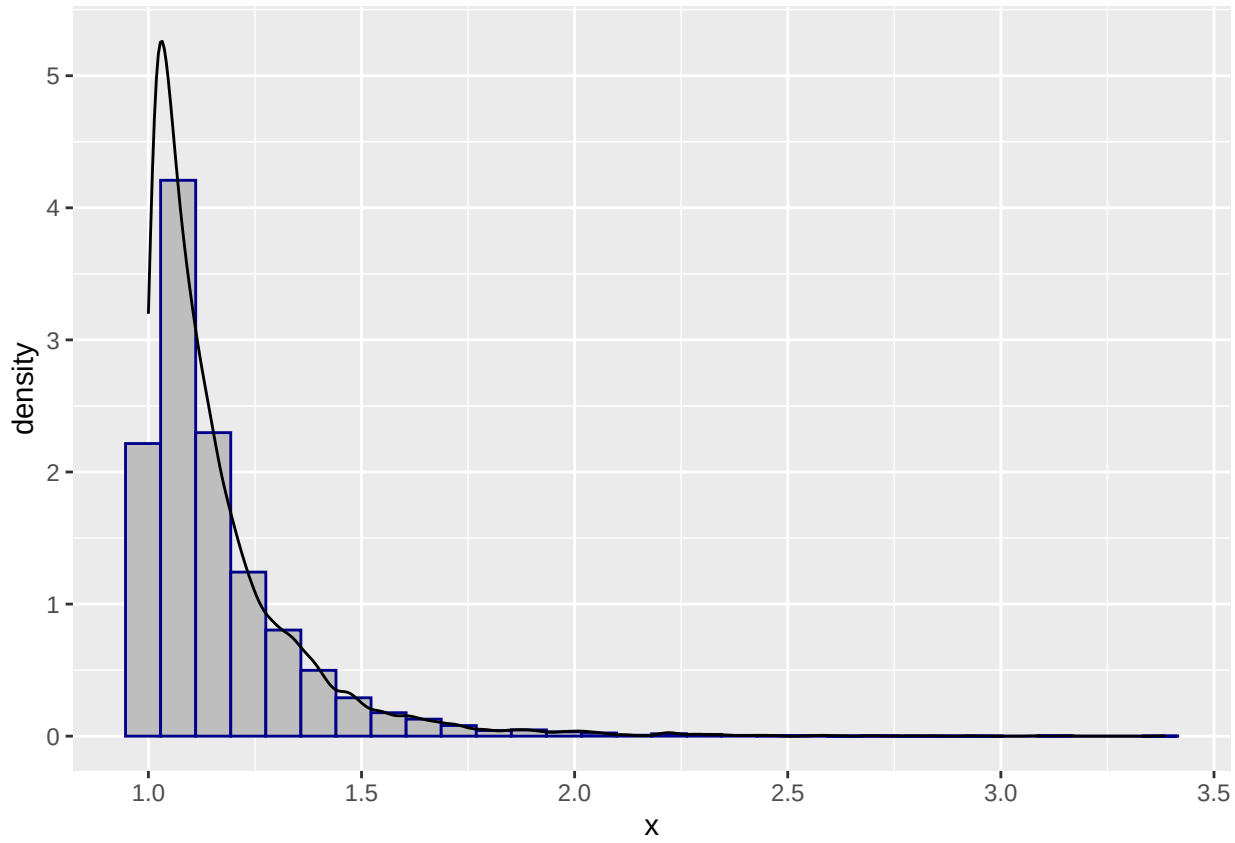
The Pareto distribution is related to the exponential distribution as follows. If X is Pareto-distributed with minimum x_m and index α, then $Y = log(\frac{X}{x_m})$ is exponentially distributed with rate parameter α.

$Y = log(\frac{X}{x_m}) \sim Exp(\gamma)$
$Y = log(X) \sim Exp(\gamma)$

Point c

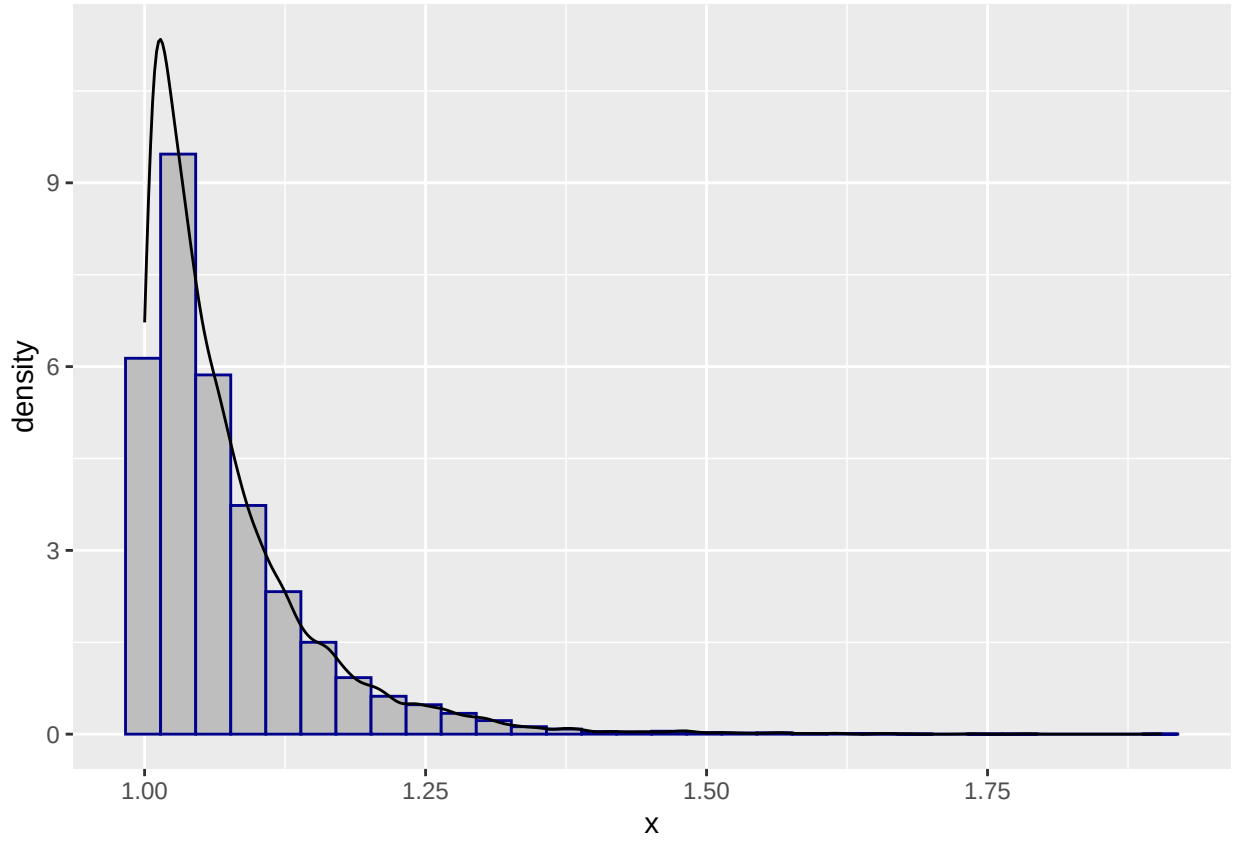
```
#sampler for Pareto
xsampler<- function(n, gamma){
  u<- runif(n)
  x<-1/((1-u)^(1/gamma))
  return(x)
}

#sampler for Exponential
ysampler<- function(x){
  y<- log(x)
  return(y)
}
i=0
gamma = c(7,15,25)

x1=xsampler(10000,10)
y1=ysampler(x1)
x2<-xsampler(10000,10)
y2<- ysampler(x2)

#Histograms and densities for different value of gamma
ggplot(data.frame(x=xsampler(10000,gamma[1])), aes(x)) +
  geom_histogram(aes(y=..density..), fill="gray",col="darkblue")+
  geom_density()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



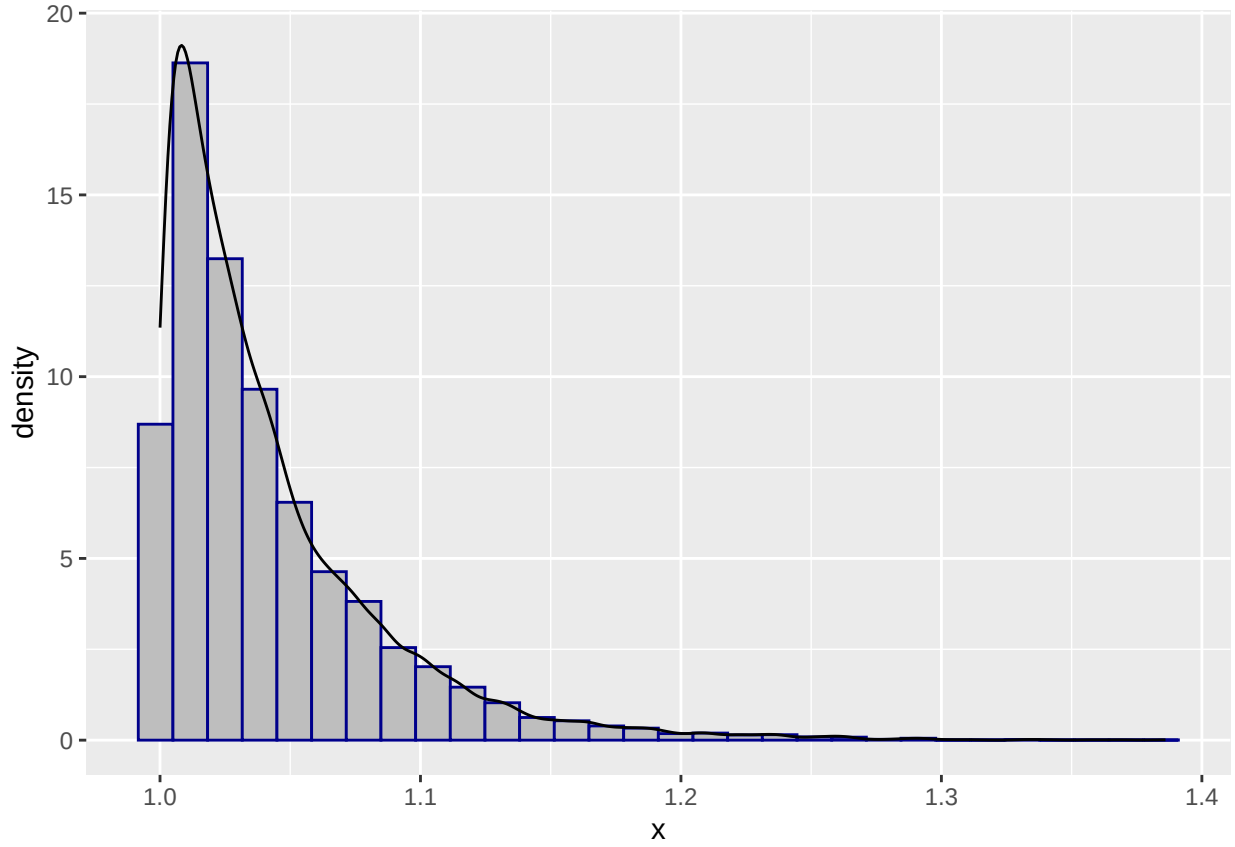

```
ggplot(data.frame(x=xsampler(10000,gamma[2])), aes(x)) +
  geom_histogram(aes(y=..density..), fill="gray",col="darkblue")+
  geom_density()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(data.frame(x=xsampler(10000,gamma[3])), aes(x)) +
  geom_histogram(aes(y=..density..), fill="gray",col="darkblue")+
  geom_density()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The plots above show histograms and density plots in the Pareto distribution for γ respectively equal to 7, 15 and 25. As the value of γ increases, the Pareto distribution becomes less skewed and more concentrated around the mode. Higher gamma values result in distributions with shorter tails of extreme values. On the other hand, lower gamma values lead to distributions with pronounced right-skewness and longer tails.

Point d

MC integration

Point a

We write the probability of a standard Normal r.v. X as an integral thanks to its probability density function.

$$P(X > 20) = \int_{20}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

Reference: Wikipedia

The crude Monte Carlo estimation of this quantity is deemed to fail because the region of integration is so far out in the tails of the standard normal distribution.

Point b

Rewrite the integral employing the change of variable $Y = \frac{1}{X}$.

$dy = -\frac{1}{x^2}dx$
$dx = -x^2 dy = -\frac{1}{y^2}dy$
$y = Y_{20} = 0,05$
$y = Y_{\infty} = 0$

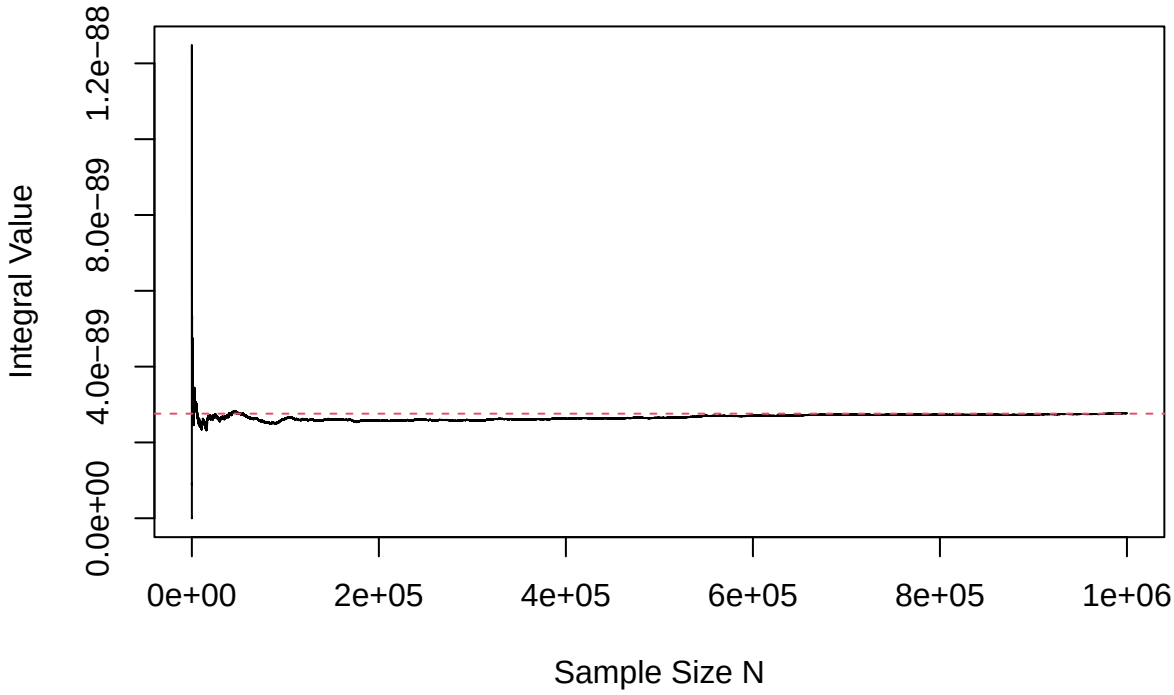
So $\int_{20}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$
$= \int_{0,05}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2y^2}} - \frac{1}{y^2} dy$
$= \int_{0}^{0,05} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2y^2}} \frac{1}{y^2} dy$

Express the integral as the expected value of a Uniform distribution over (0, 0.05). $0,05 \int_0^{0,05} \frac{f_y(y)}{0,05} dy$ with $0 < y < 0,05$

```
theme_set(theme_bw())

#Crude method to estimate the integral
set.seed(155523)
n <- 1000000
a <- 0
b <-  0.05
u <-runif(n, a, b)

intfunc <- function(y){
  f = (1/sqrt(2*pi)) * (1/(y^2)) * (exp(-1/(2*(y^2))))
}



H1 = (b-a)* intfunc(u)
psi_hat_MC <- mean(H1)
psi_hat_MC
```

```
## [1] 2.767355e-89
```

```
#Corrisponding ergotic plot
set.seed(1234)
i_1n <- 1:n
plot( i_1n, cummean(H1), type='l', lwd=1,
      xlab="Sample Size N", ylab="Integral Value",col=1)
abline(h=integrate(intfunc, a, b)$value, lwd=1, col=2, lty=2)
```



Point c

we consider estimator given by H2 below. We can construct an antithetic estimator as $\hat{\theta}_{AV} = \frac{\hat{\theta}_{MC}(1)+\hat{\theta}_{MC}(2)}{2}$

```
H2 <- (b-a)*intfunc(0.05-u)
MC1 <- mean(H1)
MC2 <- mean(H2)

MCA <- (MC1+MC2)/2


MCE_MC1 <- sd(H1)/sqrt(n)
MCE_MC1
```

```
## [1] 3.908867e-91
```

```
MCE_MC2 <- sd(H2)/sqrt(n)
MCE_MC2
```

```
## [1] 3.943859e-91
```

The above shows the Monte Carlo error in MC1 and MC2. The equation below shows the Monte Carlo error in Antithetic Variable.

```
MCE_MCA <- sqrt(1+cor(H1,H2))*sd(H1)/sqrt(2*n)
MCE_MCA
```

```
## [1] 2.757021e-91
```

```
effgain<- MCE_MCA/MCE_MC1
effgain
```

```
## [1] 0.7053249
```

The efficiency gain is given above. ## Point d

```
integrate(intfunc, a, b)
```

```
## 2.759158e-89 with absolute error < 5.4e-89
```

```
pnorm(20,0,1,lower.tail = F)
```

```
## [1] 2.753624e-89
```

The value of both the results is equal.