Project 4

Benjamin Draves

Exercise 1

(a)

Suppose that $X \sim \text{Gumbel}(\mu, \sigma)$ then

$$F_X(x) = \mathbb{P}(X \leq x) = \exp\left\{-\exp\left\{-\frac{x-\mu}{\sigma}\right\}\right\}$$

From this we can find $F^{-1}(x)$ as follows

$$\begin{aligned} x &= \exp\left\{-\exp\left\{-\frac{y-\mu}{\sigma}\right\}\right\} \\ -\log(x) &= \exp\left\{-\frac{y-\mu}{\sigma}\right\} \\ -\log(\log(x^{-1})) &= \frac{y-\mu}{\sigma} \\ \mu - \sigma\log(\log(x^{-1})) &= y \end{aligned}$$

Hence we see that

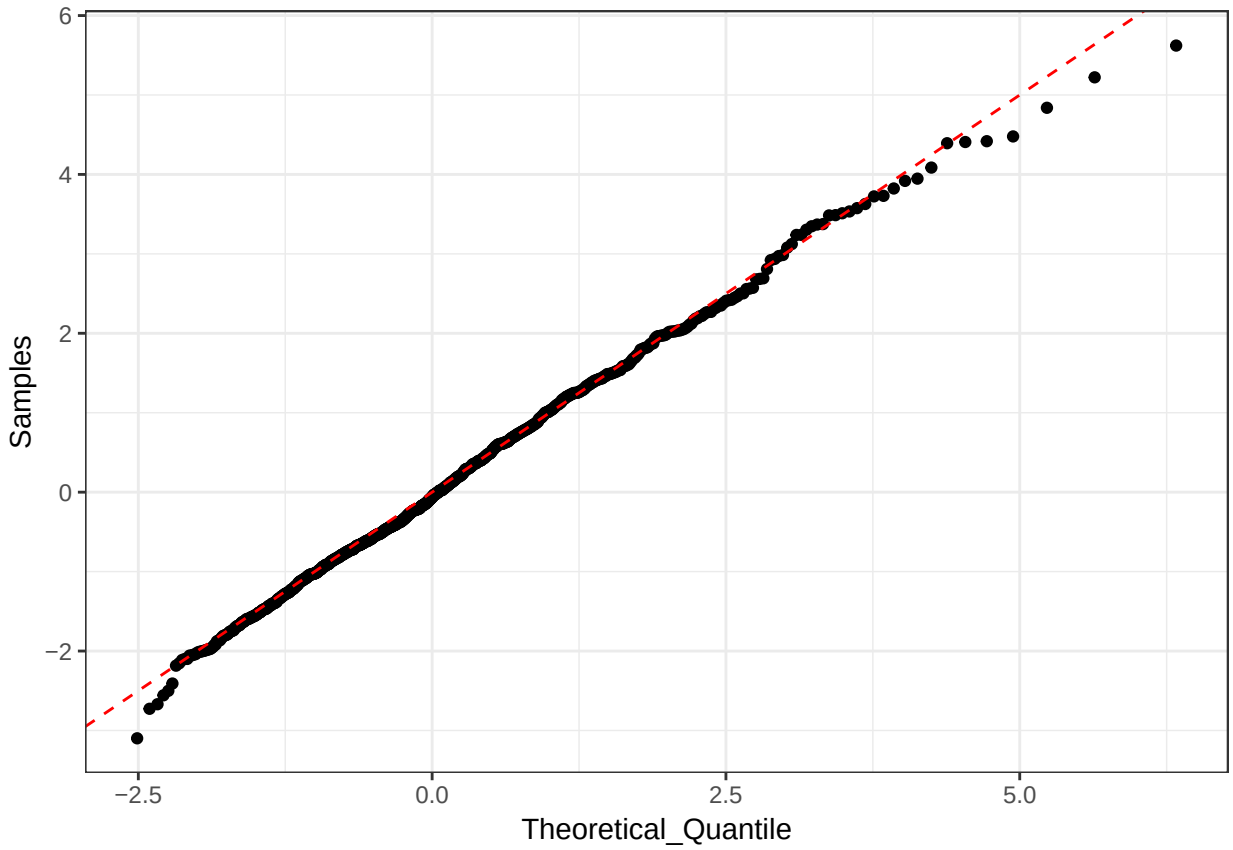
$$F_X^{-1}(x) = \mu - \sigma\log(\log(x^{-1}))$$

Using this expression of the inverse of the cumulative distribution function, we see that $X \overset{D}{=} F_X^{-1}(U) = \mu + \sigma\log(\log(U^{-1}))$ where $U \sim Unif(0,1)$. With this we can build a sampler as follows.

```
rgumbel <- function(mu = 0, sigma = 1, n = 100) mu - sigma * log(log(1/runif(n)))
```

Next we implement this sampler as with mean $\gamma = -\psi(1)$ where $\psi(x) = \frac{d}{dx}\log(\Gamma(x))$ is the digamma function. Here we plot the QQ-plot to show these samples follow the theoretical quantile curve.

```
set.seed(1985)
samp <- rgumbel(mu = digamma(1), sigma = 1, n = 1000)

#make qqplot
p <- seq(0, 1, length.out = 1000 + 2)[-c(1, 1000 + 2)]
quant_theo <- digamma(1) - 1*log(log(1/p))
df <- data.frame(Samples = sort(samp), Theoretical_Quantile = quant_theo)
library(ggplot2)
ggplot(aes(x = Theoretical_Quantile, y = Samples), data = df) +
  geom_point()+
  geom_abline(slope = 1, color="red", linetype="dashed")+
  theme_bw()
```



(b)

Suppose that $Y \sim \text{Exp}(\lambda)$ and that $X \sim F_X$ where

$$\begin{aligned} F_X(x) = \mathbb{P}(X \leq x) = \mathbb{P}(Y \leq x | a \leq Y < b) I(x \geq a) = \frac{\mathbb{P}(Y \leq x, a \leq Y < b)}{\mathbb{P}(a \leq Y < b)} I(x \geq a) \\ = \frac{\mathbb{P}(a \leq Y \leq \min(x, b))}{\mathbb{P}(a \leq Y \leq b)} I(x \geq a) = \frac{F_Y(\min(x, b)) - F_Y(a)}{F_Y(b) - F_Y(a)} I(x \geq a) \\ = \begin{cases} 1 & b \leq x \\ \frac{F_Y(x) - F_Y(a)}{F_Y(b) - F_Y(a)} & a \leq x < b \\ 0 & x < a \end{cases} \end{aligned}$$

Using this expression we can find the inverse of F_X as follows

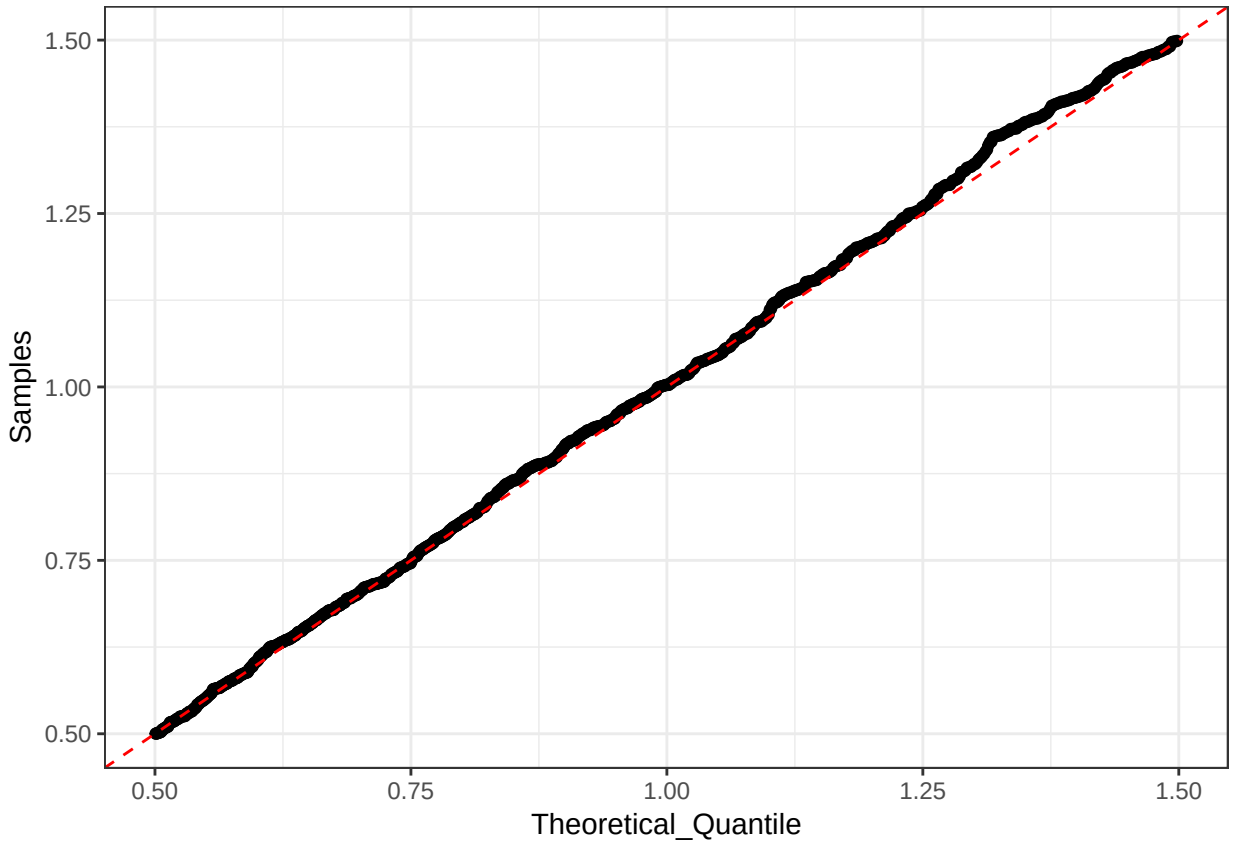
$$F_X^{-1}(x) = \begin{cases} b & b \leq x \\ F_Y^{-1}\left(F_Y(a) + (F_Y(b) - F_Y(a)) * x\right) & a \leq x < b \end{cases}$$

Recall that as $Y \sim \text{Exp}(\lambda)$ we have $F_Y(x) = 1 - \exp(-\lambda x)$ and $F_Y^{-1}(x) = -\frac{\log(1-x)}{\lambda}$. Therefore, all together, we have the following inverse CDF.

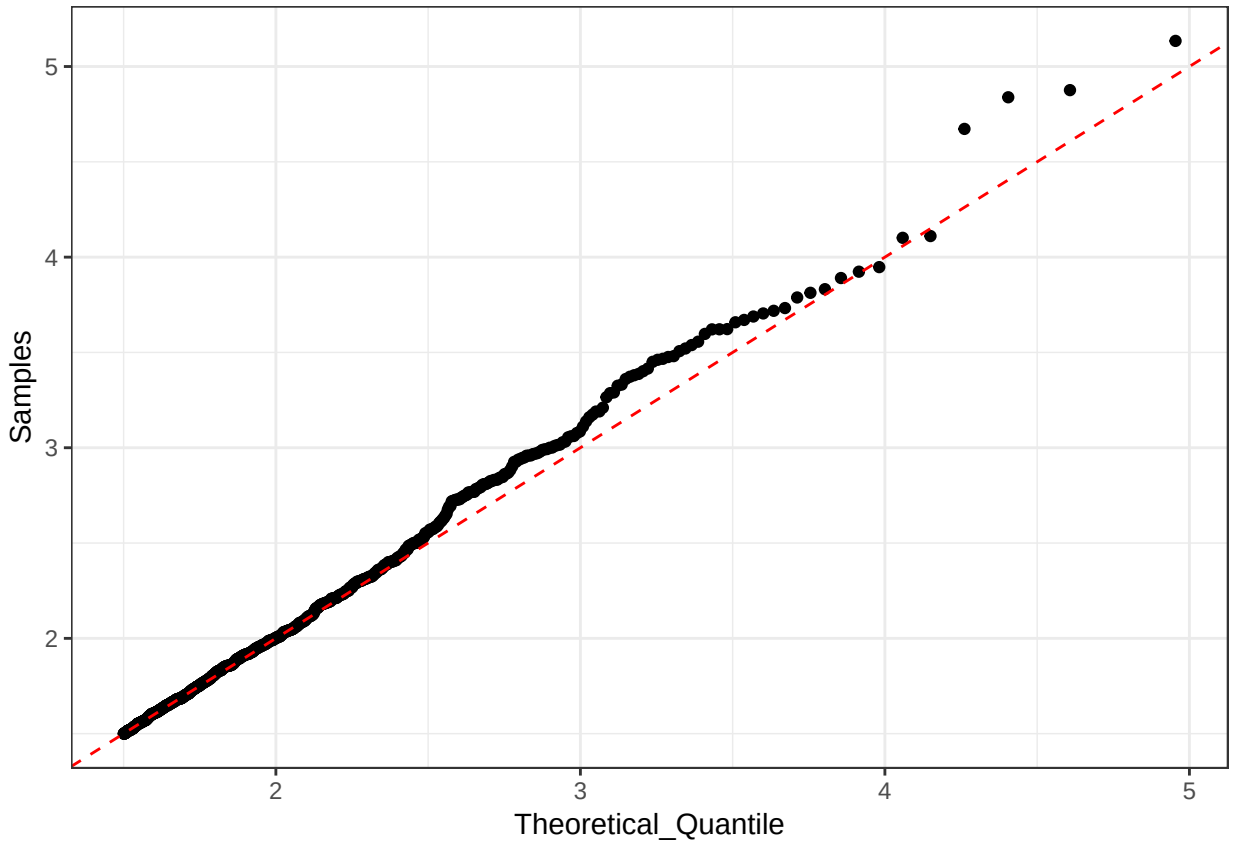
$$F_X^{-1}(x) = \begin{cases} b & b \leq x \\ -\frac{1}{\lambda}\log\left[1-(1-\exp(-\lambda a)+(1-\exp(-\lambda b)-1+\exp(-\lambda a))*x)\right] & a \leq x < b \end{cases}$$
$$= \begin{cases} b & b \leq x \\ -\frac{1}{\lambda}\log\left[\exp(-\lambda a)-(\exp(-\lambda a)-\exp(-\lambda b))*x\right] & a \leq x < b \end{cases}$$

Using this expression, we can build our sampler as follows.

```
#set up lse
LOGEPS <- log(.Machine$double.eps / 2)
lse <- function (x) {
  m <- max(x); x <- x - m
  m + log(sum(exp(x[x > LOGEPS])))
}#


F.inv <- function(x, a, b , lambda){
  if(b > 1/.Machine$double.eps) return(a - log(1 - x)/lambda) # If b = infity
  ifelse(x > b, b,-log(x*(exp(-lambda*b)- exp(-lambda*a))+exp(-lambda  *a))/lambda) # inverse function
}#F inverse function

rTrunExp <- function(lambda, a,b,n) F.inv(runif(n), a, b,lambda)

#(1, .5, 1.5)
set.seed(1234)
samp <- rTrunExp(lambda = 1, a = .5, b = 1.5, n = 1000)

p <- seq(0, 1, length.out = 1000 + 2)[-c(1, 1000 + 2)]
quant_theo <- F.inv(p, a = .5, b = 1.5, lambda = 1)
df <- data.frame(Samples = sort(samp), Theoretical_Quantile = quant_theo)

ggplot(aes(x = Theoretical_Quantile, y = Samples), data = df) +
  geom_point()+
  geom_abline(slope = 1, color="red", linetype="dashed")+
  theme_bw()
```




```r
#(2, 1.5, Inf)
set.seed(1234)
samp <- rTrunExp(lambda = 2, a = 1.5, b = Inf, n = 1000)

p <- seq(0, 1, length.out = 1000 + 2)[-c(1, 1000 + 2)]
quant_theo <- F.inv(p, a = 1.5, b = Inf, lambda = 2)
df <- data.frame(Samples = sort(samp), Theoretical_Quantile = quant_theo)

ggplot(aes(x = Theoretical_Quantile, y = Samples), data = df) +
  geom_point()+
  geom_abline(slope = 1, color="red", linetype="dashed")+
  theme_bw()
```



Exercise 2

(a)

First note that we can write the p_i as follows.

$$p_i = \frac{w_i}{\sum_{i=1}^{n} w_i} = \frac{\exp(l_i)}{\sum_{i=1}^{n} \exp(l_i)}$$

Using thism we can write the $\log(p_i)$ as follows.

$$\log(p_i) = l_i - \log \sum_{i=1}^{n} \exp(l_i) = l_i - lse(l)$$

Recall the c_i is defined to be $c_i = \sum_{k=1}^{i} p_k$ so we see that the $\log(c_i)$ is given by the following.

$$\log c_i = \log \sum_{k=1}^{i} \exp \log(p_k) = \log \sum_{k=1}^{i} \exp(l_k - lse(l)) = \otimes_{k=1}^{i} (l_k - lse(l))$$

Having calculating the $log(c_i))$ we can use the same scheme as above but this time considering the log of a sample from a uniform $U(0, 1)$. This scheme is implemented below.

```
#add lse code
LOGEPS <- log(.Machine$double.eps / 2)
lse <- function (x) {
  m <- max(x); x <- x - m
```

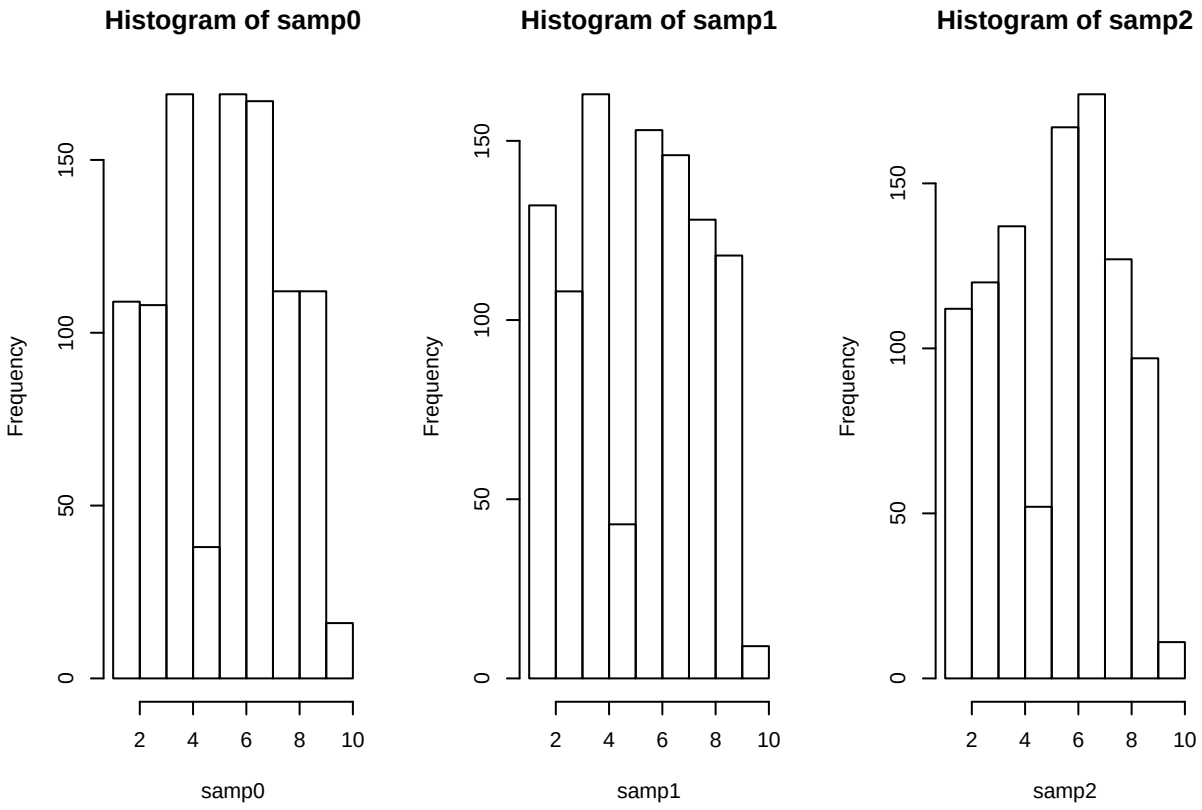
```
  m + log(sum(exp(x[x > LOGEPS])))
}

#write new rcal function
rcatl <- function(n,l){
  lsel <- lse(l) #define lse(l)
  logc <- numeric(length(l))
  for(i in 1:length(l)){
    logc[i] <- lse(l[1:i] - lsel) #calculate log(c_i)
  }

  findInterval(log(runif(n)),logc) + 1 #find interval of log(U(0,1))
}
```

To verify this sampler, we compare the methods in a, b, and the classic *rcart* function below.

(b)

Here we simply implement the Gumbel Max Trick in the sampler below.

```
rcat2 <- function(n,w)replicate(n, which.max(rgumbel(mu = digamma(1), sigma = 1, n = length(w)) + log(w
rcat <- function (n, w) findInterval(runif(n), cumsum(w / sum(w))) + 1 #define the standard rcat functi

set.seed(1)
w <- runif(10)
samp0 <- rcat(1000, w)
samp1 <- rcatl(1000, log(w))
samp2 <- rcat2(1000, w)
par(mfrow = c(1,3)); hist(samp0); hist(samp1);hist(samp2)
```



Exercise 3

(a)

$$
\begin{aligned}
(z-x)^2 &\geq 0 \\
z^2 - 2zx + x^2 &\geq 0 \\
\frac{z^2}{2} - zx + \frac{x^2}{2} &\geq 0 \\
\frac{z^2}{2} - zx &\geq -\frac{x^2}{2} \\
\frac{1}{\sqrt{2\pi}} \exp\left\{\frac{z^2}{2} - zx\right\} &\geq \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2}\right\} \\
e_z(x) &\geq \phi(x)
\end{aligned}
$$

(b)

First, we derive the density if a truncated exponential. Recall that we showed for $X \sim TruncExp(\lambda, a, b)$ that for $a \leq x < b$ that the distribution funciton is given by

$$F_X(x) = \frac{1 - e^{-\lambda x} - 1 + e^{-\lambda a}}{1 - e^{-\lambda b} - 1 + e^{-\lambda a}} = \frac{e^{-\lambda a} - e^{-\lambda x}}{e^{-\lambda a} - e^{-\lambda b}} = \frac{1 - e^{-\lambda(x-a)}}{1 - e^{-\lambda(b-a)}}$$

Now, as this function is differentiable with respect to x, we can its density as follows.

$$f_X(x) = \frac{\partial}{\partial x} F_X(x) = \frac{\lambda e^{-\lambda(x-a)}}{1 - \lambda e^{-\lambda(b-a)}}$$

We will use this function in the derivation of the following mixture distribution.

$$\begin{aligned}
Mg(x) &= e_0(x)I(0 \le x < x_0^*) + e_1(x)I(x_0^* \le x < x_1^*) + e_2(x \ge x_1^*) \\
&= \frac{1}{\sqrt{2\pi}} \left(I(0 \le x < \frac{1}{2}) + e^{1/2-x} I(\frac{1}{2} \le x < \frac{3}{2}) + e^{2-2x} I(\frac{3}{2} \le x) \right) \\
&= \frac{1}{\sqrt{2\pi}} \left(\frac{1}{2} U(0, 1/2) + (1 - e^{-1}) \frac{e^{-(x-1/2)}}{1 - e^{-(1.5-.5)}} I(\frac{1}{2} \le x < \frac{3}{2}) + \frac{2e^{-2(x-1-.5+.5)}}{2(1 - e^{-(\infty-3.2)})} I(\frac{3}{2} \le x) \right) \\
&= \frac{1}{\sqrt{2\pi}} \left(\frac{1}{2} U(0, 1/2) + (1 - e^{-1}) \frac{e^{-(x-1/2)}}{1 - e^{-(1.5-.5)}} I(\frac{1}{2} \le x < \frac{3}{2}) + \frac{e^{-1}}{2} \frac{2e^{-2(x-3/2)}}{1 - e^{-(\infty-3/2)}} I(\frac{3}{2} \le x) \right) \\
&= \frac{1}{\sqrt{2\pi}} \left(\frac{1}{2} U(0, 1/2) + (1 - e^{-1}) TruncExp(1, 0.5, 1.5) + \frac{e^{-1}}{2} TruncExp(2, 1.5, \infty) \right)
\end{aligned}$$

Let $g_0(x) = 2$ be the density of the uniform on $[0, 1/2]$, $g_1(x) = \frac{e^{-(x-1/2)}}{1-e^{-(3/2-1/2)}}$ be the desensity of $TruncExp(1, 0.5, 1.5)$, and lastly $g_2(x) = \frac{2e^{-2(x-3/2)}}{1-e^{-(\infty-3/2)}}$ be the desensity of $TruncExp(2, 1.5, \infty)$. Morever, letting $w_0 = \frac{1}{2}$, $w_1 = (1 - e^{-1})$, and $w_2 = e^{-1}/2$, and their sum be $w = w_0 + w_1 + w_2$ we can rewrite this mixture density as follows.

$$Mg(x) = \frac{w}{\sqrt{2\pi}} \left(\frac{w_0}{w} g_0(x) + \frac{w_1}{w} g_1(x) + \frac{w_2}{w} g_2(x) \right)$$

Using this we can write M and the mixture weights as follows.

$$\begin{aligned}
M &= \frac{3 - e^{-1}}{2\sqrt{2\pi}} \\
\frac{w_0}{w} &= \frac{1}{3 - e^{-1}} \\
\frac{w_1}{w} &= \frac{2(1 - e^{-1})}{3 - e^{-1}} \\
\frac{w_2}{w} &= \frac{e^{-1}}{3 - e^{-1}}
\end{aligned}$$

(c)

```
rbern <- function (n,p = 0.5) runif(n) < p

rNorm <- function(n = 100){
  #set up accepet samples
  samp <- numeric(n)
  samp.no <- 1

  #set up weights and mixture densities
```

```
  w <- c(1/2, 1 - 1/exp(1), 1/(2*exp(1)))
  M <- sum(w)/sqrt(2 * pi)
  w <- w/sum(w)

  g0 <- function(x) 2
  g1 <- function(x) exp(-(x -1/2))/(1 - 1/exp(1))
  g2 <- function(x) 2*exp(-2*(x -3/2))
  phi <- function(x) 1/sqrt(2*pi)*exp(-(x^2/2))

  for(i in 1:n){

  repeat{
    U <- runif(1)
      if(U<w[1]){
        #Accept/Reject from uniform (0,1/2)
        x <- .5 * runif(1)
        if(runif(1) < phi(x)/(M*w[1]*g0(x))){
          samp[i] <- x
          break
        }
      }else if(U < w[1] + w[2]){
        #Accept/Reject from RTrunc(1,.5,1.5)
        x <- rTrunExp(1, .5, 1.5, 1)
        if(runif(1) < phi(x)/(M*w[2]*g1(x))){
          samp[i] <- x
          break
        }
      }else{
        #Accept/Reject from RTrunc(2,1.5,infity)
        x <- rTrunExp(2, 1.5, Inf, 1)
        if(runif(1) < phi(x)/(M*w[3]*g2(x))){
          samp[i] <- x
          break
        }
      }
    }
  }
  (2*rbern(n) - 1)*samp
}
```

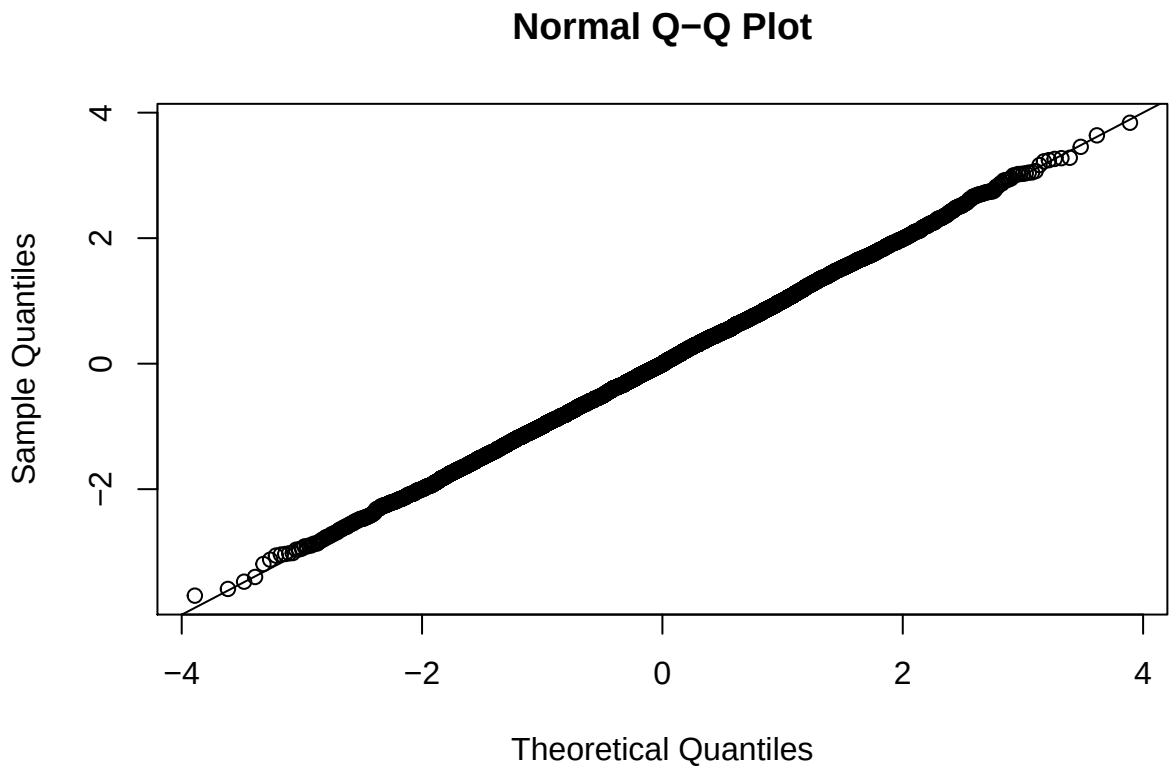
(d)

Here we see that we reject whenever $U \leq \frac{f(x)}{Mg(x)}$ hence we see that the expected proprotion of rejections is given by the area between $Mg(x)$ and $f(x)$ over $\mathbb{R}^+$ divided by the area uner the $Mg(x)$ curve over $\mathbb{R}^+$. Mathematiaclly, we have

$$\text{Prop. Rej} = \frac{\int_{\mathbb{R}^+}[Mg(x) - f(x)]dx}{\int_{\mathbb{R}^+} Mg(x)dx} = \frac{M - 1/2}{M} = 1 - \frac{1}{2M}$$

(e)

```r
samp <- rNorm(10000)
qqnorm(samp);abline(a = 0, b = 1)
```



Exercise 4

(a)

We run 100,000 simulations to obtain a Monte Carlo esimate of the probability of reaching the BU Pub. Treating this simulation as a Bernoulli experiment, we see that $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} I(X_{final}^{(i)} = 20)$ where $I(X_{final}^{(i)} = 20)$ are indepedent random variables with comment mean and finite variance. Hence we have by the classical central limit theorem, $\sqrt{n}(\hat{p} - p) \xrightarrow{D} N(0, \sigma^2)$ where $\sigma^2 = \text{Var}(I(X_{final}^{(i)} = 20)) = p(1-p)$ which we estimate by $\hat{p}(1-\hat{p})$. Therefore, using this infromation, we can define the following $(1-\alpha) \times 100\%$confidence interval for $\hat{p}$

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

We will use this interval for $\alpha = .05$.

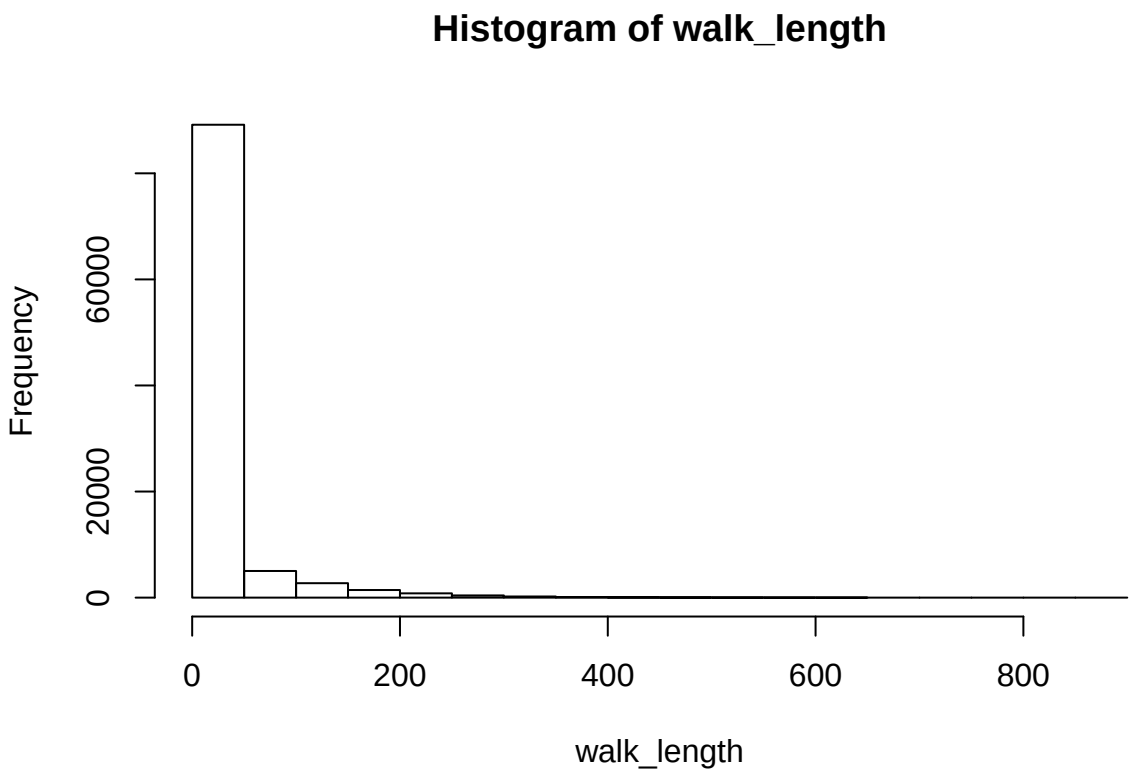
```r
rwalk <- function (p) {
  j <- 1 # start
  walk <- c() # store movements
  repeat {
  j <- j + (2 * rbinom(1, 1, p) - 1) # move
  walk <- append(walk, j)
  if (j == 0 || j == 20) return(walk)
  }
}
```

```r
n <- 100000
set.seed(1985)
res <- replicate(n, rwalk(.5)) #run simulation
end <- sapply(res, function(x) tail(x, 1))
phat <- unname(table(end)[2])/n # estimate p

upper <- phat + 1.96 * sqrt(phat*(1 - phat)/n)
lower <- phat - 1.96 * sqrt(phat*(1 - phat)/n)
knitr::kable(data.frame(Lower = lower, Estimate = phat, Upper = upper)) #print CI
```

Lower	Estimate	Upper
0.0495969	0.05096	0.0523231

(b)

```r
walk_length <- sapply(res, length)
hist(walk_length)
```



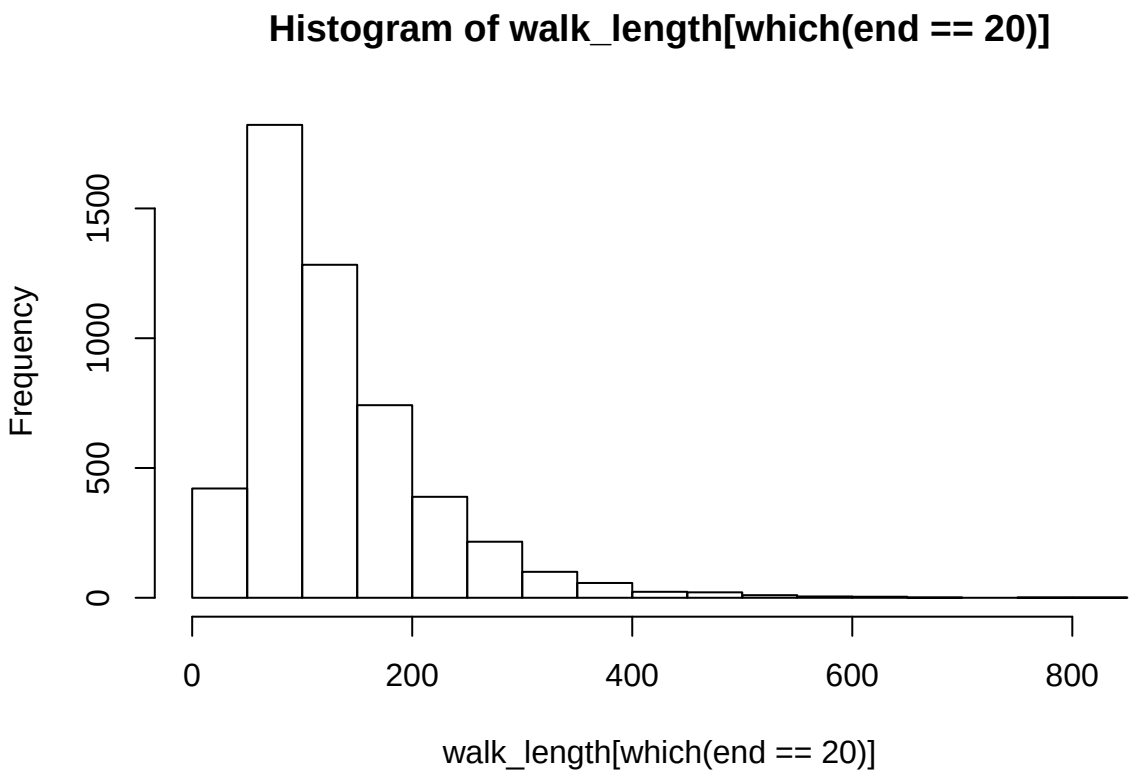

```r
length(which(walk_length>200))/n
```

```
## [1] 0.01662
```

Here we see clearly that the distribution of walk lenght is zero - inflated with a long right tail suggesting gamma behavior. Morever it appears that 1.662% of walks

(c)

```r
hist(walk_length[which(end == 20)])
```



Here we see a dramatic shift to the right. That is we see that the mode of the distribution is in the range roughly [50,100]. Moreover, we see a dramatic decrease in small walk lengths (this makes sense as the minimum walk length is 19). Moreover, we see that this decaying tail still presits. This shapes looks like a truncated exponential distribution with $a = 19$, $b = Inf$, and λ near 50.

(d)

```r
dugout <- function(x) length(which(x == 18))
ndugout <- sapply(res, dugout)
summary(ndugout)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.2052  0.0000 35.0000
```

The mean number of times a random walker would be in front of Dugout is .2052 times. We note, however, that the third quartile is zero, suggesting that it is quite rare to pass in front of Dugout.

(e)

Here we design a function that returns the importance sampling weights of the random walk. Notice here that a walk to the right happens with probability p and left with probability $1 - p$. Hence the probability of a given walk is $p^{n_r}(1-p)^{n_l}$ where n_r is the number of steps to the right and n_l is the number of steps to the left. Hence we see that by chaning the probability of a step to the right from p_1 to p_2 corresponds with the following importance sampling weights.

$$\frac{(p_1)^{n_r}(1-p_1)^{n_l}}{(p_2)^{n_r}(1-p_2)^{n_l}}$$

The function below takes a walk and calculates these weights.

```
ISW <- function(p1,p2,x){
  step_dir <- diff(x)
  nr <- length(which(step_dir>0))+1
  nl <- length(which(step_dir<0))
  (p1^nr * (1 - p1)^nl)/(p2^nr * (1 - p2)^nl)
}

ISW(.5, .55, rwalk(.55)) #example
```

```
## [1] 0.9090909
```

(f)

Now we turn to using importance sampling for this same experiment. We change the probability to $p_2 = .55$ to encourage more trips to BP. We then control for this shift by using the weights given in the previous part.

```
res.55 <- replicate(n, rwalk(.55)) #simulate
ind <- which(sapply(res.55, function(x) tail(x, 1)) == 20) #reached BP
w <- mean(sapply(res.55[ind], function(x) ISW(.5, .55, x))) # weight calculations
phatMC <- mean(sapply(res.55, function(x) ifelse(tail(x,1) == 20, 1, 0))) #MC estimate
phatIS <- w * phatMC #IS estimate
phatIS
```

```
## [1] 0.05010593
```

Next we turn to comparing the standard deviation of the two methods. Now notice that for MC estimate we have the following $\text{Var}(\hat{p}_{MC}) = \frac{p_1(1-p_1)}{n}$ which we estimate by $\frac{\hat{p}_1(1-\hat{p}_1)}{n}$. For importance sampling we see that

$$\begin{aligned}
\text{Var}(\hat{p}_{IS}) &= \text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} wI_{p_2}(X_{last}^{(i)} = 20)\right) \\
&= \frac{1}{n}\text{Var}\left(wI_{p_2}(X_{last}^{(i)} = 20)\right) \\
&= \frac{w^2}{n}p_2(1-p_2)
\end{aligned}$$

We then estimate this quantity by $\frac{w^2}{n}\hat{p}_2(1-\hat{p}_2)$. Together we see that the ratio of the standard deviations can be written as follows

$$\frac{\sqrt{\hat{p}_1(1-\hat{p}_1)}}{w * \sqrt{\hat{p}_2(1-\hat{p}_2)}} = \frac{\widehat{sd(MC)}}{w * \widehat{sd(IS)}}$$

```
MC <- sapply(res, function(x) ifelse(tail(x, 1) == 20, 1, 0))
IS <- sapply(res.55, function(x) ifelse(tail(x, 1) == 20, 1, 0))
sd(IS)/sd(MC)
```

```
## [1] 1.769305
```

```
sd(MC)/ (w *sd(IS))
```

```
## [1] 2.097963
```

Hence we see that importance sampling gives more than twice the reduction in variance as compared to the traditional Monte Carlo estimate.