STAT 5443: Homework 2

Daniel Kiser

February 21, 2018

Problem 1

- To use the inverse CDF transform method, we first find the cdf of $p(x)$ by taking the integral of $sin(x)$:

$$F(t) = \int_0^t sin(x) = 1 - cos(t)$$

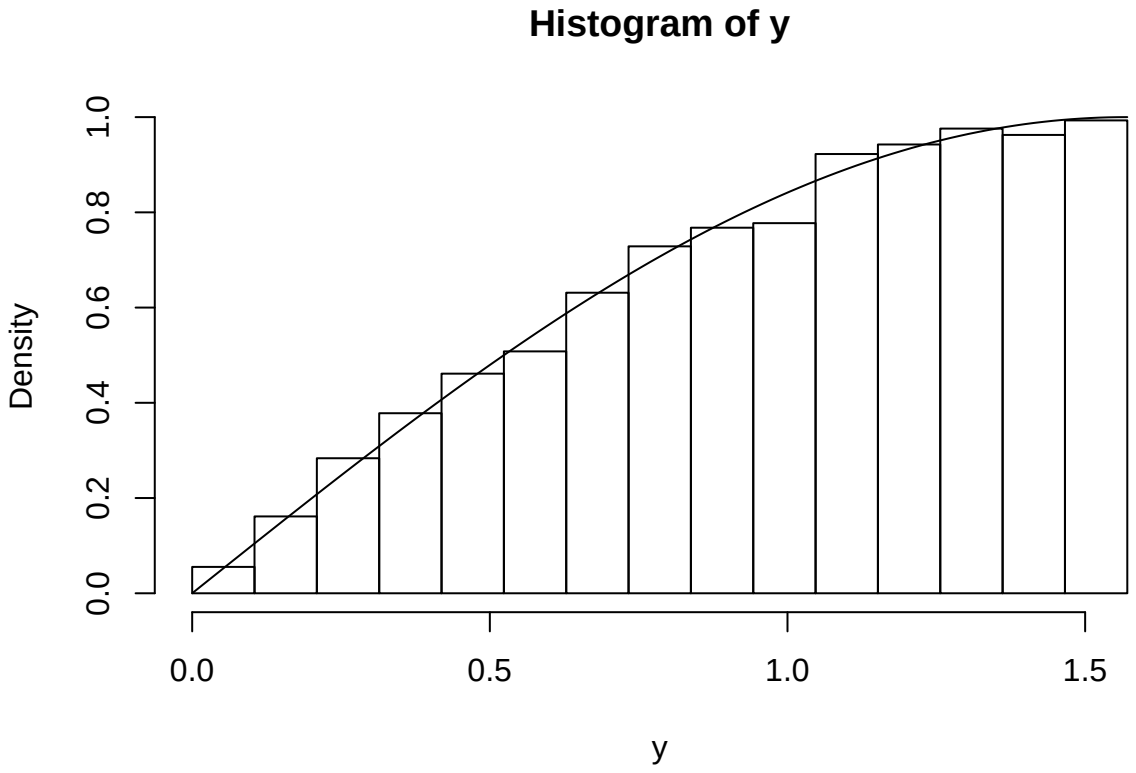
Then we find the inverse of $F(x)$:

$$u = 1 - cos(x) >> u - 1 = -cos(x) >> 1 - u = cos(x) >> x = cos^{-1}(1 - u)$$

which is equivalent to:

$$x = cos^{-1}(u)$$

```
sinrand <- function(n) {
    u <- runif(n)
    x <- acos(u)
    return(x)
}
y <- sinrand(10000)
hist(y, freq = F, breaks = seq(0, pi/2, pi/30))
curve(sin(x), from = 0, to = pi/2, add = T)
```



- If we use a uniform density on the interval $(0, \pi/2)$ as our proposal density, then we implement our rejection sampling method as follows:

```
g_density <- function(x) {
    u <- dunif(x, min = 0, max = pi/2)
    return(u)
}

g_rv <- function() {
    u <- runif(1, min = 0, max = pi/2)
    return(u)
}

mysin <- function(x) {
    y <- sin(x)
    return(y)
}

accept_reject <- function(n, known_g_density, known_g_value,
    p, M) {
    y <- numeric(n)
    for (i in 1:n) {
        x <- known_g_value()
        u <- runif(1, min = 0, max = M * known_g_density(x))
        while (u > p(x)) {
            u <- runif(1, min = 0, max = M * known_g_density(x))
            x <- known_g_value()
        }
        y[i] <- x
    }
    return(y)
}
```

To find our value for M, we first find the maximum value of our proposal density, which we designate as g(x):

```
max <- dunif(0, min = 0, max = pi/2)
max
```

```
## [1] 0.6366198
```

Since sin(x) is largest at $sin(\pi/2) = 1$, and since $g(x) = 0.6366198$ for all values of x, we calculate M as:

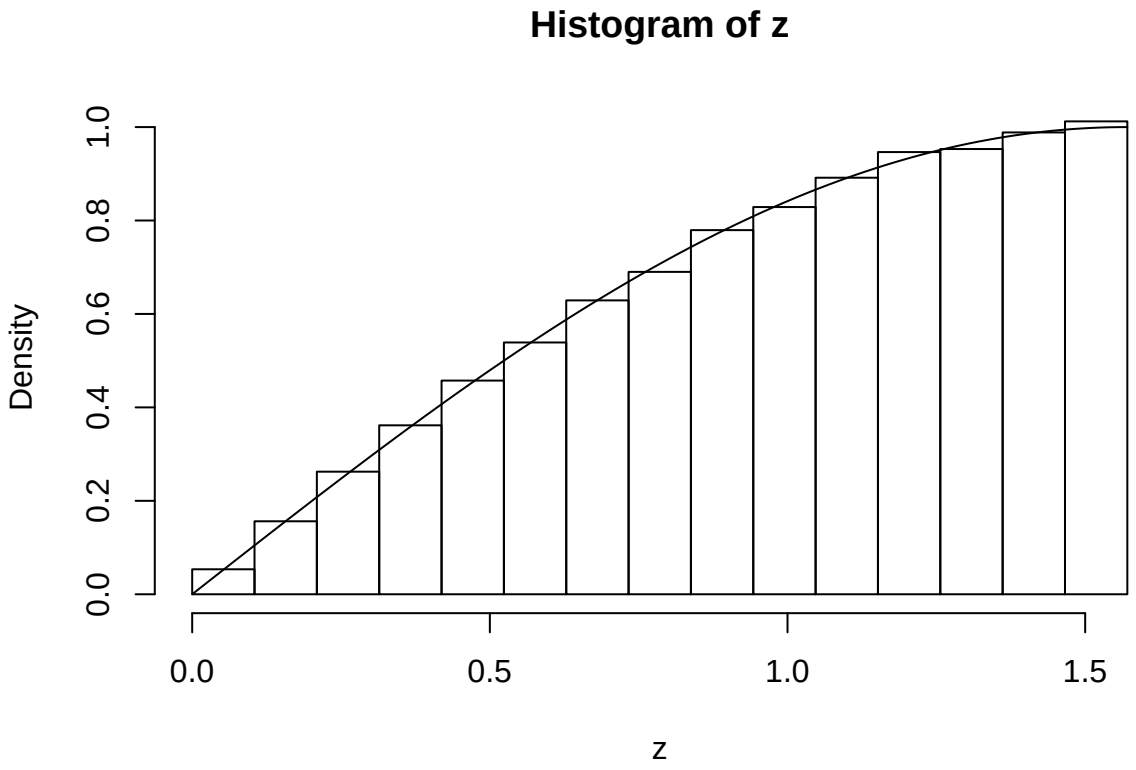
$$M = \frac{1}{g(x)}$$

```
M <- 1/max
M
```

```
## [1] 1.570796
```

We draw 100000 samples from out simulated distribution and compare the resulting histogram to the true density:

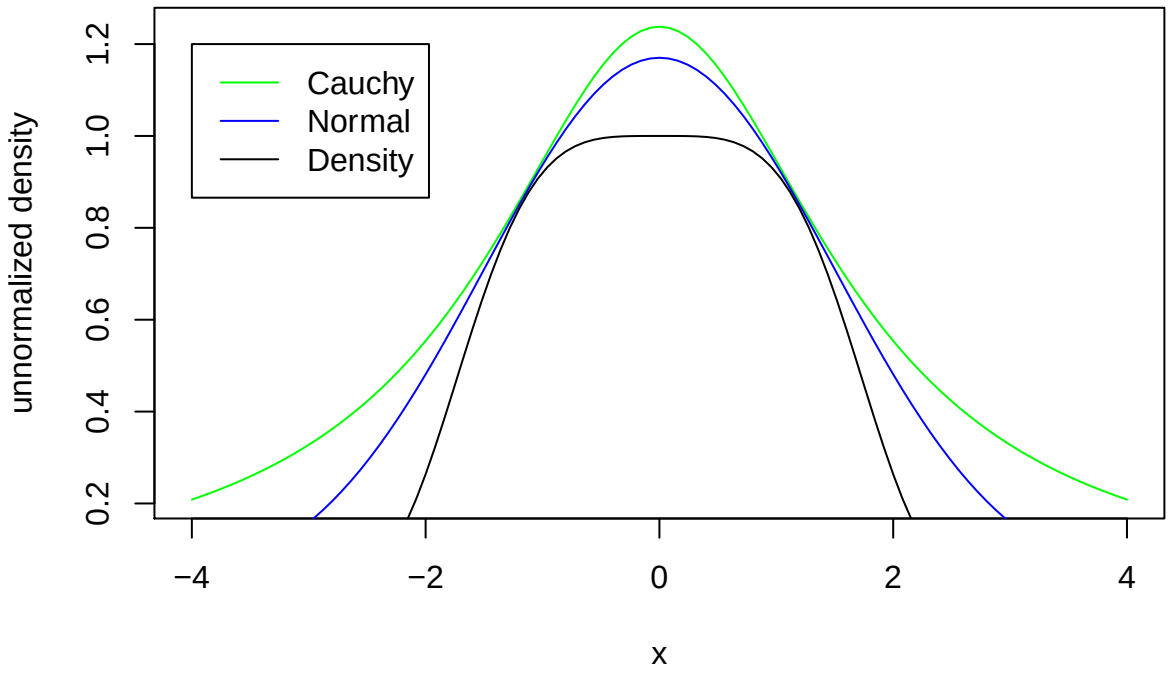
```
z <- accept_reject(1e+05, g_density, g_rv, sin, M)
hist(z, freq = F, breaks = seq(0, pi/2, pi/30))
curve(sin(x), from = 0, to = pi/2, add = T)
```



Problem 2

First, we find a good proposal density from which to draw samples. We consider the Cauchy and Normal distributions:

```
unscaled_pdf <- function(x) {
    exp((-x^4)/12)
}
norm_scaled_density <- function(x) {
    dnorm(x, sd = 1.5) * 4.4
}
cauchy_scaled <- function(x) {
    dcauchy(x, scale = 1.8) * 7
}
curve(cauchy_scaled, from = -4, to = 4, col = "green", ylab = "unnormalized density")
legend(x = -4, y = 1.2, lty = c(1, 1, 1), col = c("green", "blue",
    "black"), legend = c("Cauchy", "Normal", "Density"))
curve(norm_scaled_density, from = -4, to = 4, col = "blue", add = T)
curve(unscaled_pdf, from = -4, to = 4, add = T)
```



A normal distribution with a standard deviation of 1.5 and scaled by 4.4 appears to fit much more closely than the cauchy distribution with a scale of 1.8 and scaled by 7. We can verify this since the probability of a sampled value being accepted is 1/M. Thus, the probability of a sampled value being accepted using the normal distribution is $1/4.4 = 0.2272727$ while the probability of a sampled value being accepted using the cauchy distribution is $1/7 = 0.1428571$. Clearly, the normal distribution is the more efficient proposal density. Our accept-reject algorithm for an unnormalized density is as follows:

```
unnorm_accept_reject <- function(n, rproposed, dproposed, pdf,
    M) {
    x <- numeric(n)
    i = 1
    while (i <= n) {
        theta <- rproposed()
        u <- runif(1) * M * dproposed(theta)
        if (u <= pdf(theta)) {
            x[i] <- theta
            i <- i + 1
        }
    }
    return(x)
}
```

We also write a function to generate values from a normal distribution with a standard deviation of 1.5, and we set M = 4.4:

```
# function to generate normal value with sd = 1.5
norm_scaled_value <- function() {
    rnorm(1, sd = 1.5)
}

M <- 4.4

# generate samples
z <- unnorm_accept_reject(10000, norm_scaled_value, norm_scaled_density,
```

```
    unscaled_pdf, M)
```

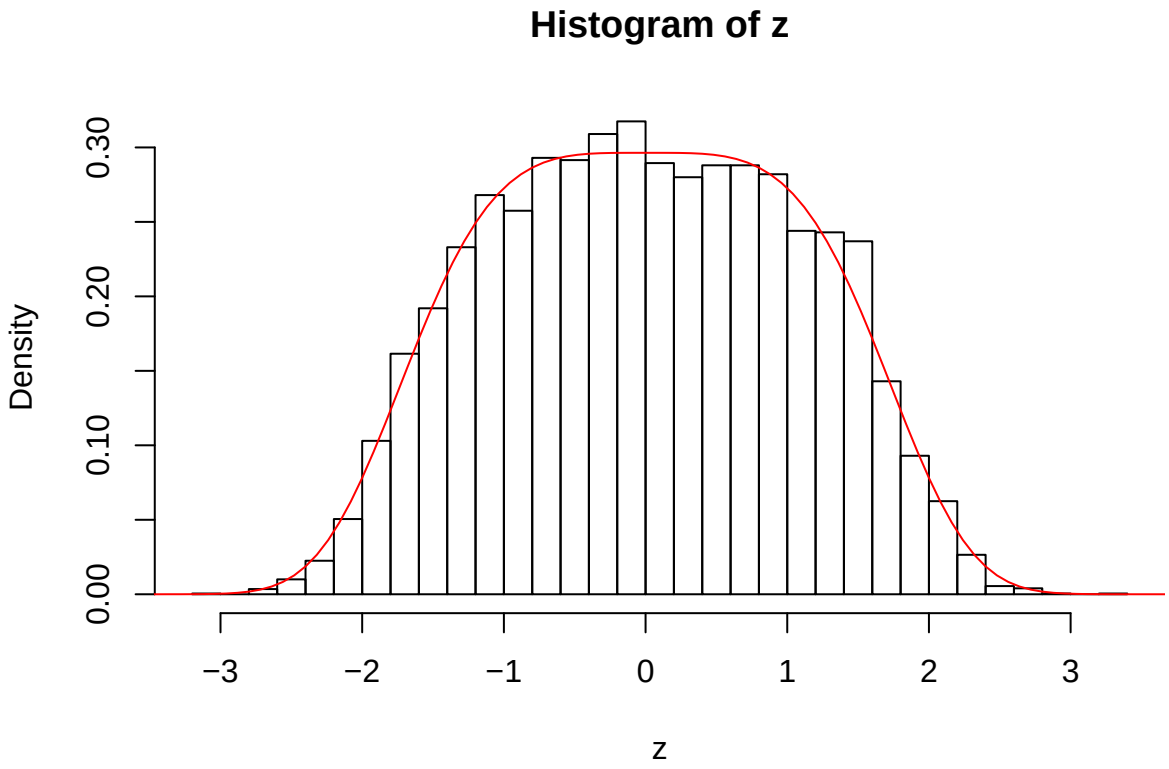
To verify that our algorithm works, we use the integrate() function in R to normalize our target density and compare our target density with our generated sample:

```
# plot histogram
hist(z, freq = F, breaks = 40)

# integrate and normalize target density and compare
# normalized density with generated sample
integrate(unscaled_pdf, lower = -Inf, upper = Inf)
```

```
## 3.37401 with absolute error < 0.00033
```

```
pdf <- function(x) {
    exp((-x^4)/12)/3.37401
}
curve(pdf, from = -4, to = 4, add = T, col = "red")
```



Problem 3

1) The method we use to sample from Y is Sampling Importance Sampling (SIR). First, we generate a sample from a known distribution, in this case the exponential distribution, since the exponential distribution has the same support as the target distribution:

```
set.seed(1234)
x <- rexp(10000)

# density function of Y
```

```
edelman <- function(y) {
    fy <- (1 + sqrt(y))/(2 * sqrt(y)) * exp(-(y/2 + sqrt(y)))
    return(fy)
}
```

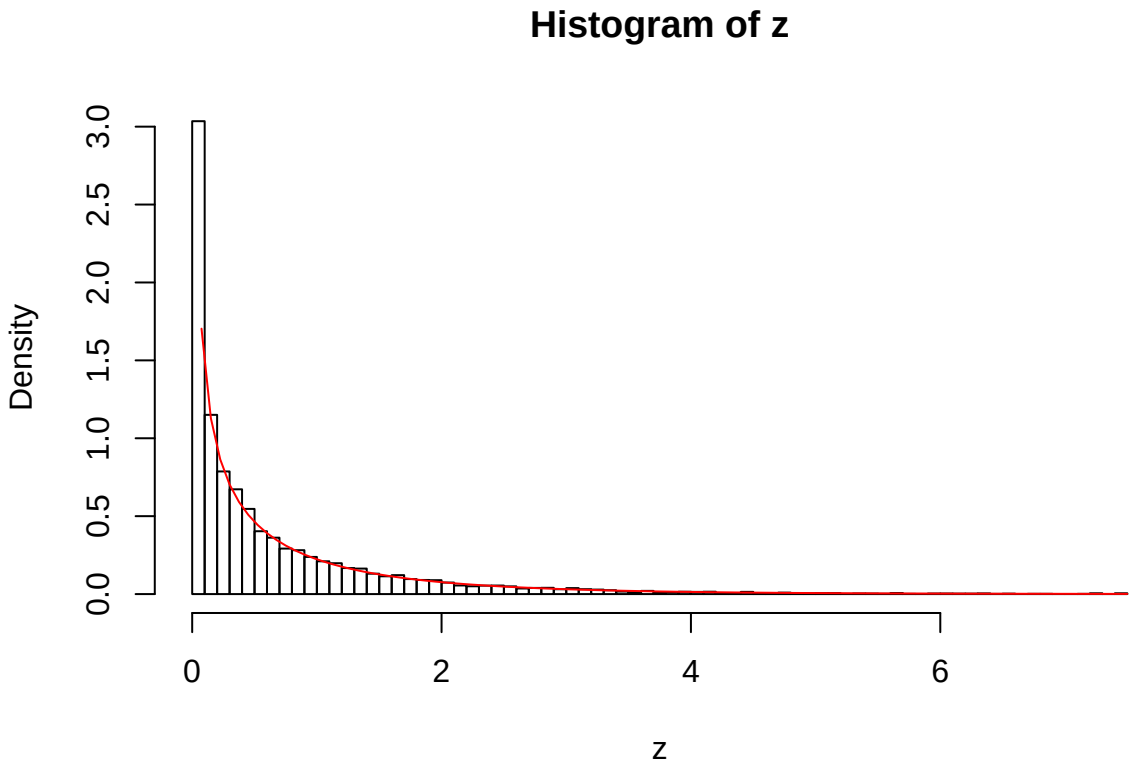
Next, we write a function to compute the importance weight for each value of x generated from the exponential distribution:

```
w <- function(y) {
    wi <- edelman(y)/dexp(y)
    wn <- sum(wi)
    wi <- wi/wn
    return(wi)
}
weights <- w(x)
```

We use those weights to sample from the values of the exponential distribution, with each weight being the probability that a particular value will be chosen:

```
z <- sample(x, 1e+05, replace = T, prob = weights)

# verify that sampling is correct by comparing histogram with
# true density
hist(z, freq = F, breaks = 100)
curve(edelman, add = T, col = "red")
```



2) To test our method, we estimate $E(log(Y))$ by simple Monte Carlo. We repeatedly sample the weighted values drawn from the exponential distribution, take the log, and find the mean. This creates an estimated distribution for $E(log(Y))$, which we use to find the estimated mean and confidence intervals.

```
E <- numeric(1000)
for (i in 1:1000) {
    z <- sample(x, 10000, replace = T, prob = weights)
    h <- log(z)
    E[i] <- mean(h)
}
mean(E)
```

```
## [1] -1.684035
```

```
CI <- c(quantile(E, 0.005), quantile(E, 0.995))
CI
```

```
##      0.5%     99.5%
## -1.741937 -1.630399
```

Thus, we find that our estimation of $E(log(Y))$ matches that of Edelman(1988), which was -1.68788.