Homework 04 - STAT440

Joseph Sepich (jps6444)

09/19/2020

```
set.seed(42)
```

Problem 1

Use R to simulate samples from a normal distribution. Let Y be a random variable with chi-squared distribution with 5 degrees of freedom.

Part a

Since we can only sample from the standard normal distribution, I would use the transformation that the Chi-square distribution is really the sum of squares of standard normal variables. The degrees of freedom is the number of standard normal random variables that make up the Chi-square distribution. In this case we would use the following transformation:

$$\chi^2 = \Sigma_{i=1}^{5} Z_i^2$$

Part b

Variance of Y:

$$Var(Y) = (\int x f(x) dx)^2 - (\int x^2 f(x) dx)$$

Foruth moment of Y:

$$\int x^4 f(x) dx$$

We can approximate all these integrals using Monte Carlo, where $h(x) = x^n$ (depending on the moment) and $f(x)$ is the density of Y that we are sampling from. Therefore we can use our samples X_i from the standard normal and transform them (5 for 5 degrees of freedom) into a chi-square sample Y_i. These samples can give us the value $\frac{1}{n}\Sigma_{i=1}^{n} h(Y_i) = \frac{1}{n}\Sigma_{i=1}^{n} Y_i^k$, which can approximate the k^{th} moment integral.

Part c

Use R to estimate the above quantities using Monte Carlo with N = 10,000 samples and report the results.

```r
n <- 10000
df <- 5
norm_samples <- matrix(rnorm(n*df), nrow=n, ncol=df)
norm_samples <- apply(norm_samples, c(1,2), function(x) {x^2})
chi_samples <- rowSums(norm_samples)

# variance
second <- sum(chi_samples^2) / n
expect_square <- (sum(chi_samples) / n) ^ 2
variance <- second - expect_square
print(variance)
```

```
## [1] 9.983276
```

```r
# compare approx with sd function
print(sd(chi_samples)^2)
```

```
## [1] 9.984275
```

```r
# fourth moment
fourth <- sum(chi_samples^4) / n
print(fourth)
```

```
## [1] 3449.642
```

Problem 2

Part a

Express the probability as an expectation and as an integral. For this we can use an indication function $g(x)$, which is 1 when $|\overline{X}_n - \mu| < \epsilon$ and 0 when it is not . ($|\overline{X}_n - \mu| \geq \epsilon$) We can use this indication function to express the probaility as an expectation:

$$E[g(x)] = P(|\overline{X}_n - \mu| < \epsilon)$$

We represent this as an integral (formula for finding the first moment) where $f(x)$ is the pdf of $|\overline{X}_n - \mu|$.

$$E[g(x)] = \int g(x)dP(x) = \int g(x)f(x)dx$$

Part b

Here $I_n = \int h(x)f(x)dx$ where $h(x)$ is the indicator function that we described above. This integral is in the form of an expectation of the indicator function, therefore all we need to do is sample from this indicator function random variable and find that sample mean, which is the Monte Carlo integration process.

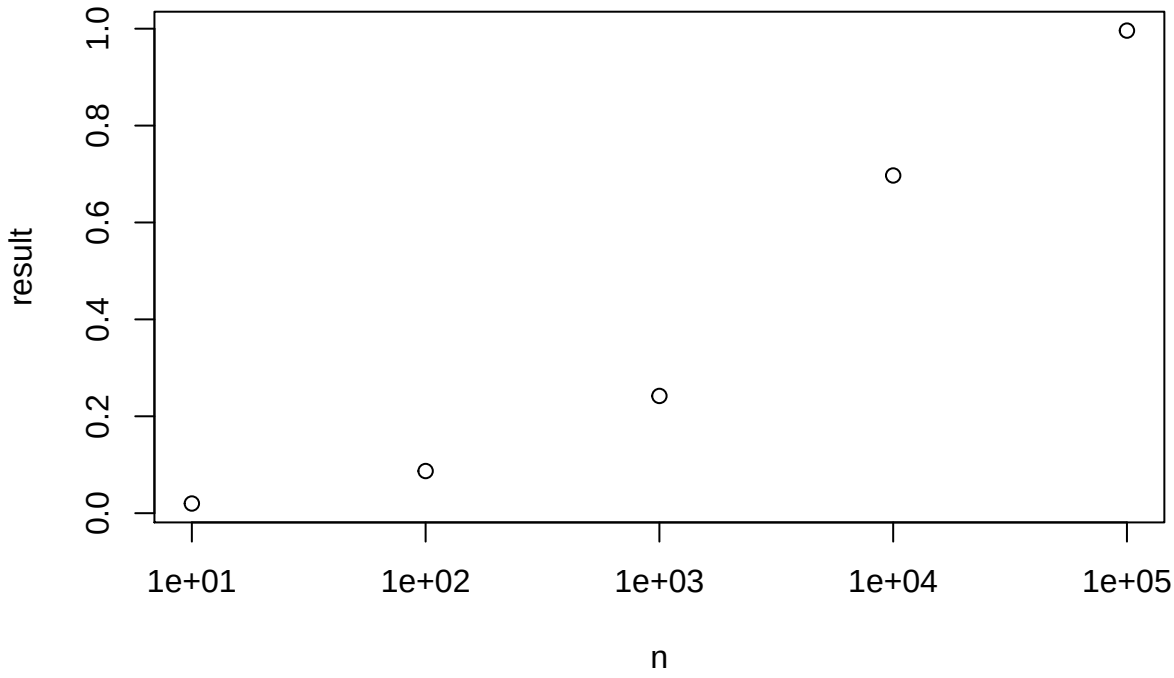
Part c

```r
epsilon <- 0.01
m <- 1000
n <- c(10, 100, 1000, 10000, 100000)
result <- vector("numeric", length(n))

mu <- 2
sigma <- 1

for (i in 1:length(n)) {
  samples <- matrix(rnorm(n[i] * m, mu, sigma), nrow = m, ncol = n[i])
  samples <- rowSums(samples)
  x_n <- samples / n[i]
  comp_val <- abs(x_n - mu)
  result[i] <- sum(as.integer(comp_val < epsilon)) / m
}

plot(n, result, log='x')
```



Part d

As you can see in the plot above the correct behavior is shown. As the sample size of the sample mean gets larger, the expected value of our indicator function gets closer to 1, which means the numerical approximation of the $P(|\overline{X}_n - \mu| < \epsilon)$ gets closer to 1. This is exactly what the limit states in the weak law of large numbers.

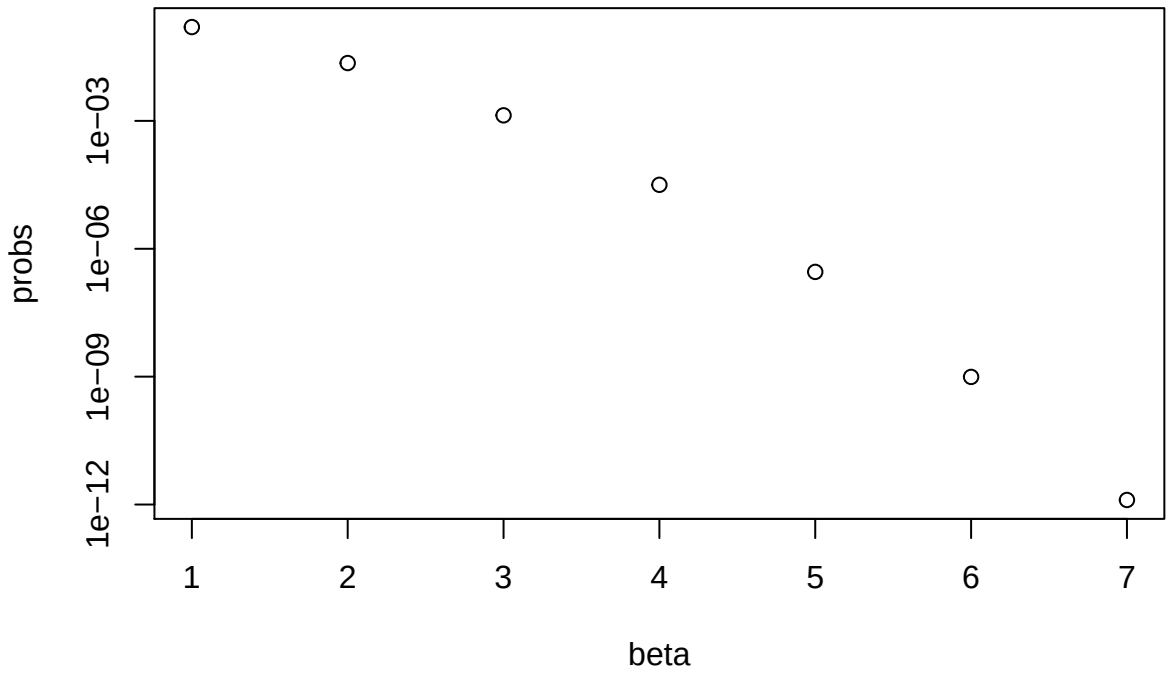
Problem 3

Part a

```
beta <- c(1:7)
mu <- 0
sigma <- 1

probs <- 1 - pnorm(beta, mu, sigma)

plot(beta, probs, log="y")
```



It would not be very reasonable to sample from the normal distribution and only take these large samples. These samples occur so sparsely that you would have to sample an incredible number of times. For example, when beta is 1, you could only use 15.8% of the samples to try to estimate the value. For beta of 2, it would be 2.3% and with beta of 7 you would be lucky to find 1 in 1,000,000 to use to estimate the value.

Part b

To find the pdf of $P(Z|Z > \beta)$ we will use the density version of Bayes theorem described where $S = Z > \beta$:

$$f_{Z|Z>\beta}(x) = \frac{f_Z(x)P(Z > \beta|Z = x)}{1 - \Phi(\beta)}$$

In the numerator we have $P(Z > \beta | Z = x)$, which is either 1 when $x > \beta$ or 0 otherwise, which makes sense, because you can't condition on the event that the value is greater than β unless it is. This gives use the density:

$$f_{Z|Z>\beta}(x) = \frac{f_Z(x)}{1 - \Phi(\beta)}$$

With the support $\beta < x < \infty$.

We can use this to find expectation:

$$\theta_\beta = \int_\beta^\infty x f_{Z|Z>\beta}(x) dx = \int_\beta^\infty \frac{x f_Z(x)}{1 - \Phi(\beta)} dx = \frac{1}{1 - \Phi(\beta)} \int_\beta^\infty x f_Z(x) dx$$

$$\theta_\beta = \frac{1}{1 - \Phi(\beta)} \int_\beta^\infty \frac{x}{\sqrt{2\pi}} e^{-x^2/2} dx = \frac{-1}{\sqrt{2\pi}(1 - \Phi(\beta))} (e^{-x^2/2})|_\beta^\infty$$

$$\theta_\beta = \frac{e^{-\beta^2/2}}{\sqrt{2\pi}(1 - \Phi(\beta))}$$

Part c

We can use a rejection sampling approach to sample values from the distribution to estimate the expectation using the sample mean. The candidate density to be used will be that of the standard normal. Recall that we want to find a value that fits the following:

$$\frac{f(x)}{g(x)} \leq c$$

To do this we can define $h(x) = \frac{f(x)}{g(x)}$,

$$h(x) = \frac{f(x)}{g(x)} = \frac{\frac{f_Z(x)}{1 - \Phi(\beta)}}{f_Z(x)} = \frac{1}{1 - \Phi(\beta)}$$

This means our scaling value $c = \frac{1}{1 - \Phi(\beta)}$. For a value of $\beta = 6$ we find our probability of acceptance to be:

```
beta <- 6
c <- 1 / (1 - pnorm(6))
1 / c
```

```
## [1] 9.865877e-10
```

This makes the standard normal a very poor density to use since we would have to wait for about 1013594635 samples to get 1 accepted. This is so poor, since the support of the conditional density only exists in the area greater than β while the standard normal support covers negative infinity to infinity. Since the values in the desired support will be drawn so rarely from the candidate function this makes it a very poor choice.

Part d

We must find C so that our function is equal to 1 under the area of its curve. Recall that we only integrate between β and ∞ due to the indicator function.

$$\int_{\beta}^{\infty} C\lambda e^{-\lambda y} dy = -Ce^{-\lambda y}|_{\beta}^{\infty} = -C(0 - e^{-\lambda\beta}) = Ce^{-\lambda\beta}$$

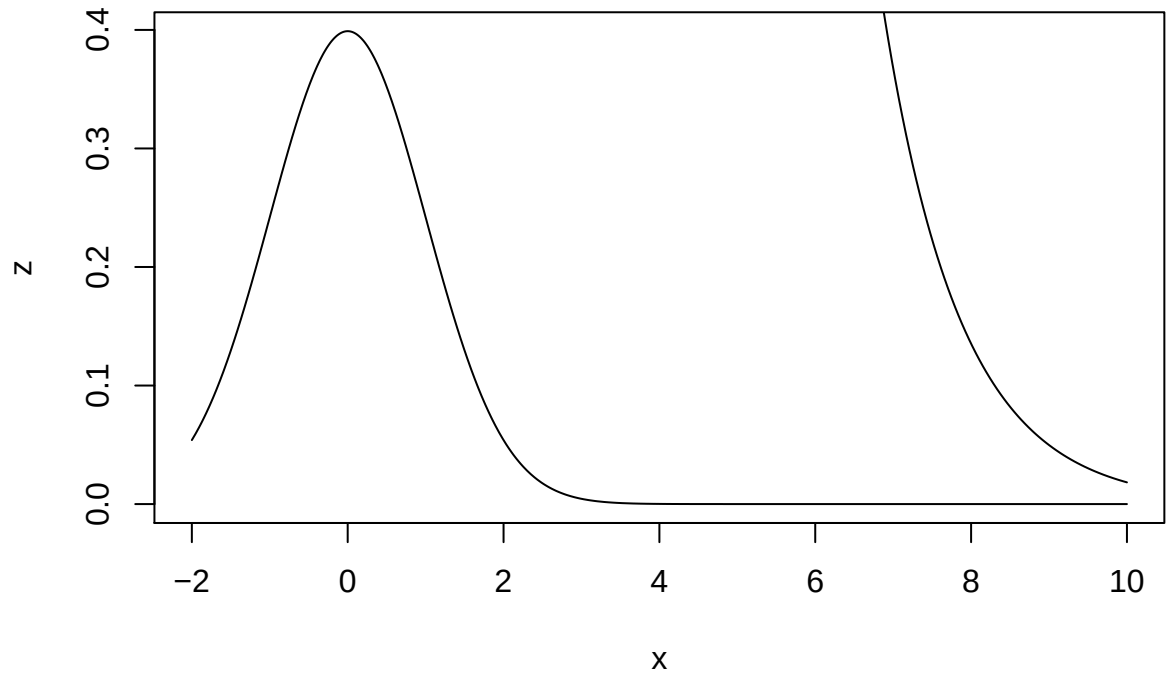
Let's set this value equal to 1 to find C:

$$Ce^{-\lambda\beta} = 1$$
$$C = e^{\lambda\beta}$$

This gives us the density $f_Y(y) = e^{\lambda\beta}\lambda e^{-\lambda y}$ for values greater than β.

```
x <- seq(-2, 10, 0.01)
z <- dnorm(x)
beta <- 6
lambda <- 1
x_2 <- seq(beta, 10, 0.01)
y <- dexp(x_2, lambda) * exp(beta)

plot(x,z, type='l')
lines(x_2,y)
```



Above is an example for beta = 6 again. It is clear that using an exponential function as a candidate density is not a good idea, because there is a large rejection area compared to the acceptance area under the exponential density curve. This will not solve the problem we have of just sampling from the normal density, since it will still require a huge amount of samples.

Part e

To compute the inverse CDF of Y let's first define the CDF:

$$F_Y(a) = \int_{\beta}^{a} e^{\lambda\beta}\lambda e^{-\lambda y}dy = -e^{\lambda\beta}(e^{-\lambda y})|_{\beta}^{a} = -e^{\lambda\beta}(e^{-\lambda a} - e^{-\lambda\beta}) = 1 - e^{\lambda(\beta-a)}$$

Now find the inverse of the CDF:

$$\begin{aligned} y &= 1 - e^{\lambda(\beta-x)} \\ e^{\lambda(\beta-x)} &= 1 - y \\ \lambda(\beta - x) &= ln(1-y) \\ \beta - x &= \frac{ln(1-y)}{\lambda} \\ x &= \beta - \frac{ln(1-y)}{\lambda} \end{aligned}$$

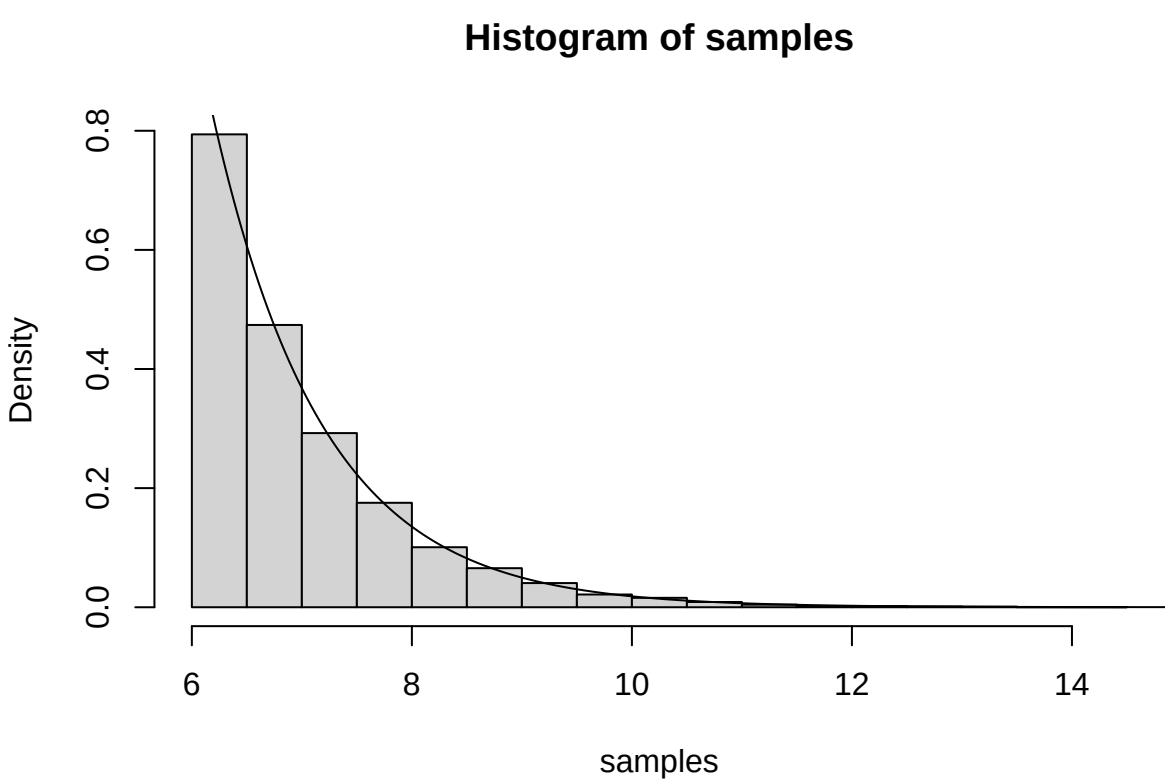
This is our inverse CDF $F_Y^{-1}(x) = \beta - \frac{ln(1-x)}{\lambda}$. Now we can use the inverse CDF method to sample from f_Y.

```
cdf_inverse <- function(x, beta, lambda) {
  beta - (log(1 - x)) / lambda
}

fy_dens <- function(x, beta, lambda) {
  exp(beta * lambda) * lambda * exp(-1 * lambda * x)
}

fy_inverse_samples <- function(n, beta, lambda) {
  samples <- runif(n)
  samples <- cdf_inverse(samples, beta, lambda)
  return(samples)
}

n <- 10000
beta <- 6
lambda <- 1
samples <- fy_inverse_samples(n, beta, lambda)
hist(samples, freq=FALSE)
x <- seq(6, 15, 0.01)
lines(x, fy_dens(x, beta, lambda))
```



Part f

Now we want to find a scaling factor c, so we can use f_Y as a candidate for rejection sampling of $f_{Z|Z>\beta}$. Recall that c must satisfy $\frac{f(x)}{g(x)} \leq c$. Lets define $h(x) = \frac{f(x)}{g(x)}$.

$$h(x) = \frac{f(x)}{g(x)} = \frac{f_{Z|Z>\beta}(x)}{f_Y(x)}$$

$$h(x) = \frac{e^{-x^2/2}}{e^{\lambda\beta}\lambda e^{-\lambda x}\sqrt{2\pi}(1-\Phi(\beta))} = \frac{e^{-x^2/2-\lambda\beta+\lambda x}}{\lambda\sqrt{2\pi}(1-\Phi(\beta))}$$

To find the max value, let's calculate the first derivative and set it to zero.

$$h'(x) = \frac{(-x+\lambda)e^{-x^2/2-\lambda\beta+\lambda x}}{\lambda\sqrt{2\pi}(1-\Phi(\beta))} = 0$$
$$(-x+\lambda)e^{-x^2/2+\lambda\beta-\lambda x} = 0$$
$$-x+\lambda = 0$$
$$x = \lambda$$

Plug it back in to $h(x)$:

$$h(\lambda) = c = \frac{e^{-\lambda^2/2-\lambda\beta+\lambda^2}}{\lambda\sqrt{2\pi}(1-\Phi(\beta))}$$

Part g

In the last part we saw that $c = h(\lambda)$. We can find the min of this function to get the desired value of λ.

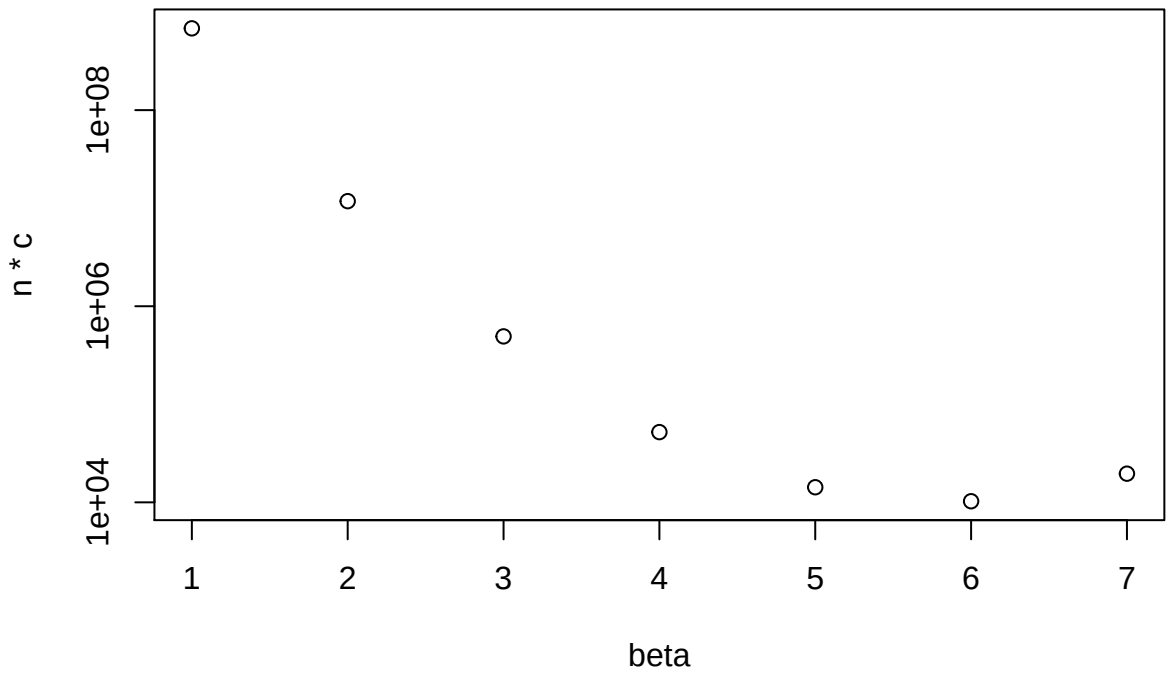
$$h'(\lambda) = \frac{(\lambda - \beta)e^{\lambda^2/2 - \lambda\beta}}{\lambda\sqrt{2\pi}(1 - \Phi(\beta))} = 0$$
$$\lambda - \beta = 0$$
$$\lambda = \beta$$

To minimize our scaling factor c we should then use the value $h(\beta)$.

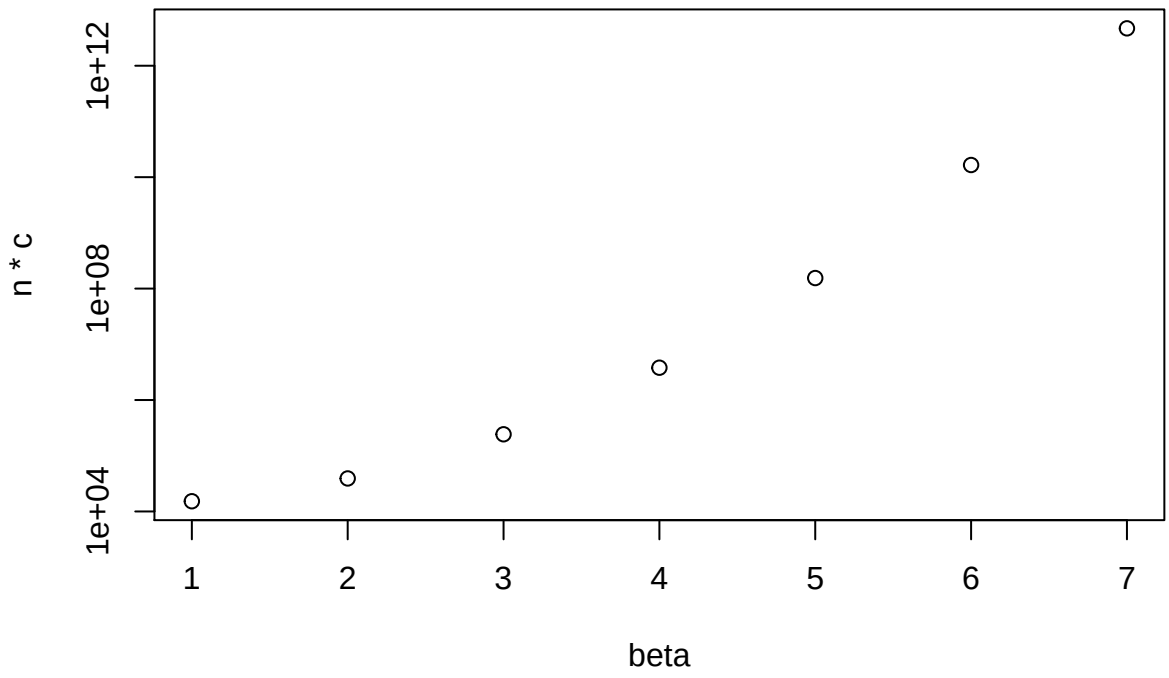
Part h

Below we can see the difference in using the proper parameter to minimize the scaling factor c. When $\lambda = 6$ when $\beta = 6$ we only expect to reject about 264 samples when getting 10,000 accepted samples. For the same *beta*, but $\lambda = 1$ we expect to have to sample over 16 billion times.

```
n <- 10000
lambda <- c(6, 1)
beta <- c(1:7)
# expected number of iterations is c * n
for (i in 1:length(lambda)) {
  c <- dnorm(lambda[i]) / (1 - pnorm(beta)) / fy_dens(lambda[i], beta, lambda[i])
  plot(beta, n*c, log="y")
  print(n*c[6])
}
```




```
## [1] 10264.14
```




```
## [1] 16525504184
```

Part i

We want to estimate:

$$\theta_\beta = \int x f_{Z|Z>\beta}(x) dx$$

We do this by performing rejection sampling here and finding the sample mean as our estimate for θ_β.

```
beta <- 3
lambda <- 3
n <- c(10, 100, 1000, 10000)
err <- vector("numeric", length(n))
theta_actual <- exp(-1 * beta ^2 / 2) / (sqrt(2 * pi)*(1 -pnorm(beta)))
# actual expected value
print(theta_actual)
```

```
## [1] 3.283099
```

```
c <- dnorm(lambda) / (1 - pnorm(beta)) / fy_dens(lambda, beta, lambda)
# scaling factor
print(c)
```

```
## [1] 1.094366
```

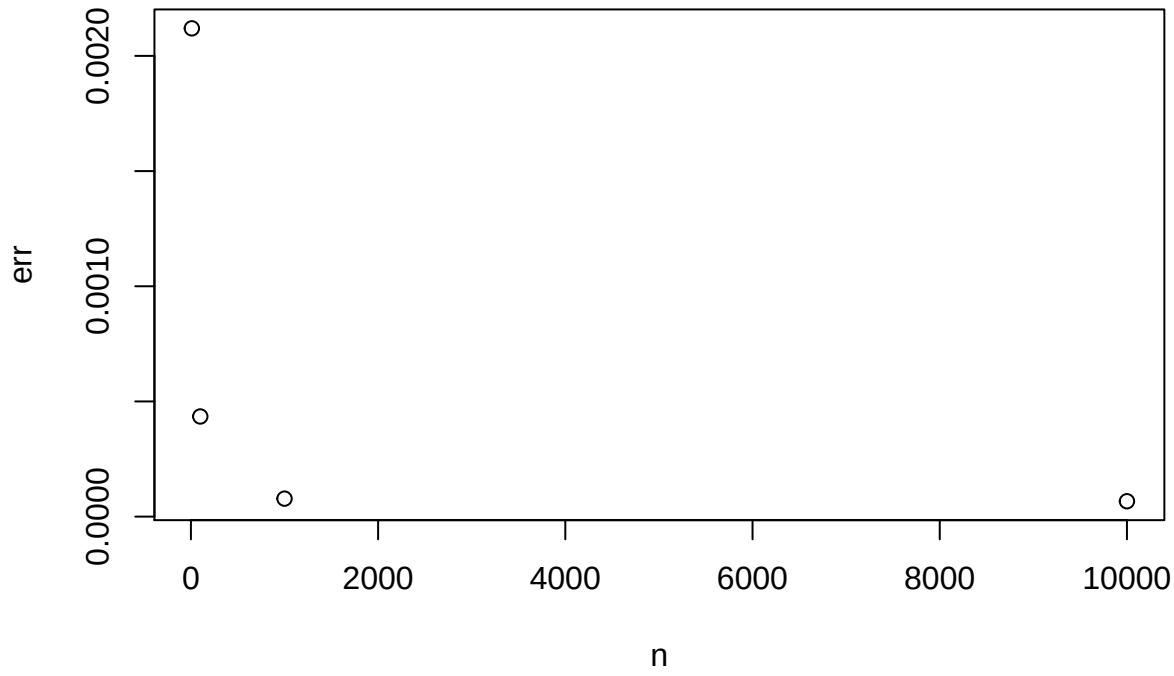
```
# conditional density
f_dens <- function(x, beta) {
  dnorm(x) / (1 - pnorm(beta))
}

for (i in 1:length(n)) {
  counter <- 0
  found_count <- 0
  found_samps <- vector("numeric", n[i])

  while (found_count < n[i]) {
   g_samp <- fy_inverse_samples(1, beta, lambda)
   g_yval <- runif(1, 0, c * fy_dens(g_samp, beta, lambda))
   if (g_yval <= f_dens(g_samp, beta)) {
     found_count <- found_count + 1
     found_samps[found_count] <- g_samp
   }
   counter <- counter + 1
  }
  # number of iterations required
  print(counter)
  theta_est <- sum(found_samps) / n[i]
  err[i] <- (theta_actual - theta_est)^2
  # estimated expectation
  print(theta_est)
}
```

```
## [1] 10
## [1] 3.23706
## [1] 111
## [1] 3.303949
## [1] 1108
## [1] 3.291932
## [1] 10921
## [1] 3.274928
```

```
plot(n, err)
```



The above results are exactly what we expected. The scaling factor c was about 1.094, meaning each estimation with rejection sampling would take about n * c iterations, which proved to be true and you can see how the expected iteration count is closer to the actual iteration count with more iterations.

Part j

This problem illustrated how crucial it is to pick a good candidate function. It also showed how there are two different ways to change your candidate density function. You can either change the actual function itself or you can change the parameter value to the density function. Clearly both the function and proper parameter are crucial for finding a minimum value of c (the scaling factor). We could see how if we didn't pay attention to what this scaling factor was we would have to sample for an outrageously large amount of time/iterations, whereas if we take the time to minimize this scaling factor, then we can reject very few samples, which is ideal.

Problem 4

Part a

$$\theta_\beta = \int h(x)f(x)dx = \int x f_{Z|Z>\beta}(x)dx$$

Here $h(x) = x$ and $f(x) = f_{Z|Z>\beta}(x)$, the density we are looking at.

Part b

$$\theta_\beta = \int \frac{h(x)f(x)}{g(x)} g(x)dx$$

This integral as an expectation with respect to g that is: $E[\frac{h(x)f(x)}{g(x)}]$

Part c

To estimate θ_β with importance sampling I can use Monte Carlo integration to approximate this integral by using the sample mean of samples drawn from the candidate density $g(x)$. Samples drawn from $g(x)$ will be "weighted" according to the ratio $\frac{f(x)}{g(x)}$, so just like in the previous problem (3) it would be important to choose a candidate density $g(x)$ that is similar to our pdf $f(x)$. Candidate functions that behave more like $f(x)$ will produce a better estimate for the same sample size compared to a candidate function that does not match as well.