Project 1 - Computational Statistics

Andreas Matre, Amir Ahmed

1/23/2020

Problem A

Problem A.1

We want to draw from the exponential pdf.

$$X \sim f(x) = \lambda e^{-\lambda x}$$

We use the inverse transform technique to draw from the exponential.

Find the cdf:

$$F(x) = \int_{-\infty}^{x} f(t)dt = 1 - e^{-\lambda x}$$

The inverse of the cdf is:

$$x = -\frac{\log(F(x))}{\lambda}$$

So if you draw:

$$U \sim unif(0, 1)$$

And let:

$$X = -\frac{\log(U)}{\lambda}$$

Then:

$$X \sim exponential(\lambda)$$

Implement code to sample:

```
# Use the inverste transform technique to draw from exponential
my.rexp <- function(lambda = 1, n = 1) {
  x = runif(n) # Sample from uniform
  x = log(x) # Tranform using inverse
  return(-x/lambda) # Return scaled resilt
}
```

```
# Draw from exponential with lambda = 1 to validate results
n = 10000 # Number of points to draw
lambda = 1
draws = my.rexp(lambda , n = n) # Drawing using inverse transform technique

# Compare variance to real variance
obs_var <- round(var(draws),4)
expected_var <- round(1/lambda^2,4)
print(paste0("Observed variance is: ",
             obs_var, " Theoretical variance is: ",
             expected_var, " With lambda: ",
             lambda))
```

```
## [1] "Observed variance is: 1.0405 Theoretical variance is: 1 With lambda: 1"
```

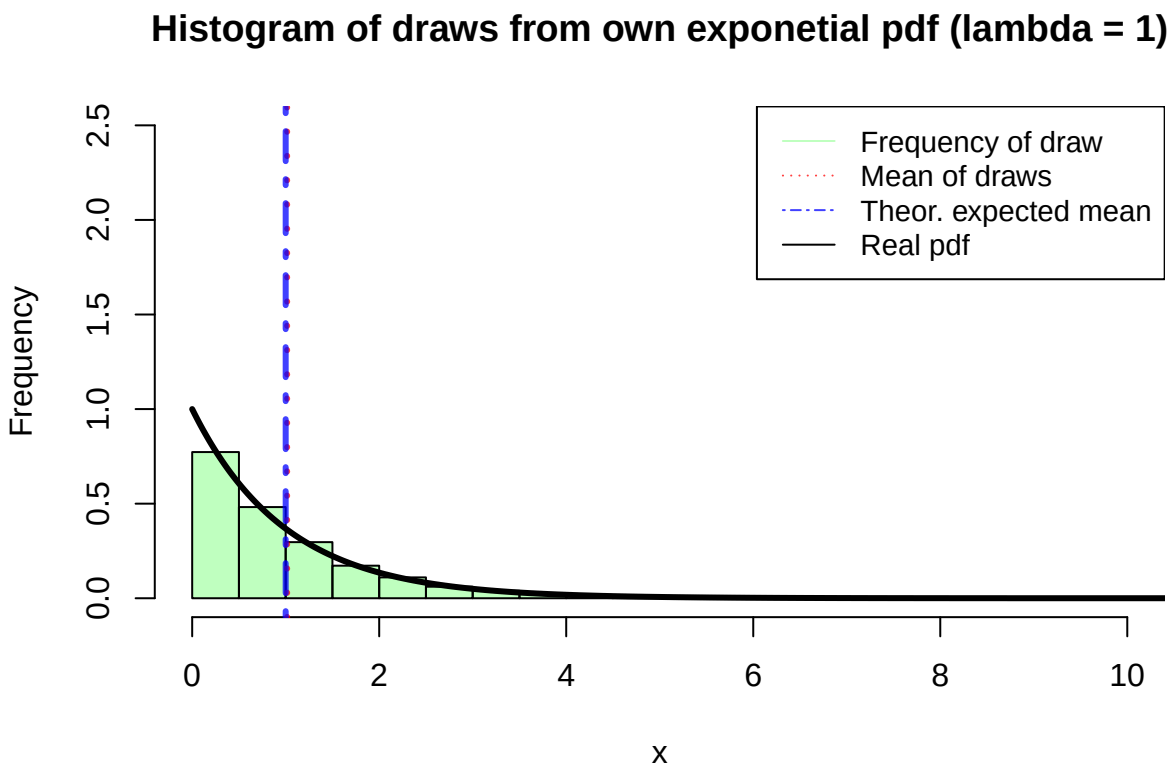

Figure 1: Draw from exponential with lambda = 1, n=10000.

With n=10^4 and $\lambda = 1$ draws we observe a variance of 1.0405 the theoretical variance is $\frac{1}{\lambda^2} = 1$ which seems to be matching. Results from sample is illustrated in Figure 1. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf.

```
# Draw from exponential with lambda = 3 to validate results
n = 10000 # Number of points to draw
lambda = 3
draws = my.rexp(lambda , n = n) # Drawing using inverse transform technique

# Compare variance to real variance
obs_var <- round(var(draws),4)
expected_var <- round(1/lambda^2,4)
print(paste0("Observed variance is: ",
             obs_var,
             "Theoretical variance is: ",
             expected_var, " With lambda: ",
             lambda))
```

```
## [1] "Observed variance is: 0.1107Theoretical variance is: 0.1111 With lambda: 3"
```

With n=10^4 and $\lambda = 3$ draws we observe a variance of 0.1107 the theoretical variance is $\frac{1}{\lambda^2} = \frac{1}{9}$ which seems to be matching. Results from sample is illustrated in Figure 2. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf.

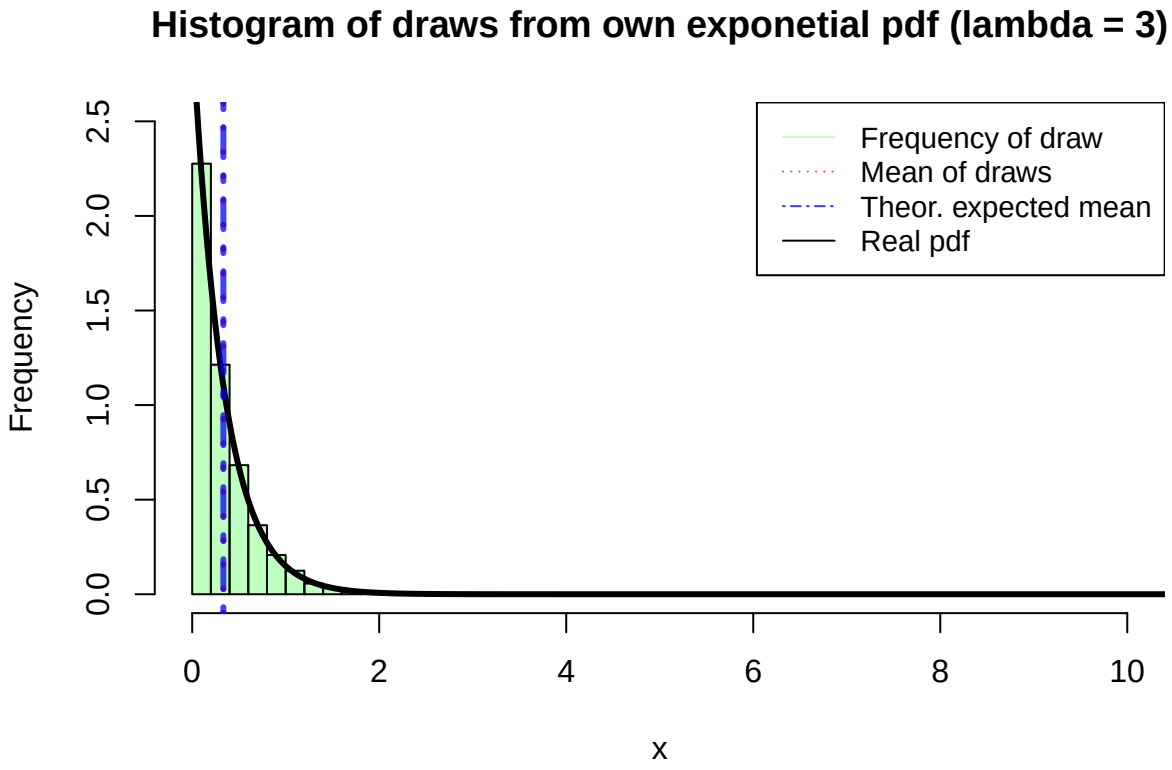

Figure 2: Draw from exponential with lambda = 3, n=10000.

Problem A2

Problem A.2.a

Wan to find the cumulative distribution function and the inverse of the cumulative distribution function of.

$$g(x) = \begin{cases} cx^{\alpha-1}, & 0 < x < 1 \\ ce^{-x} & 1 \leq x \\ 0, & \text{otherwise} \end{cases}$$

Where $\alpha \in (0, 1)$

We find the CDF, we start by splitting into different cases:

For $0 < x < 1$:

$$G(x) = \int_{-\infty}^{x} g(t)dt = \int_{0}^{x} ct^{\alpha-1}dt = \left[\frac{c}{\alpha}t^{\alpha}\right]_{0}^{x} = \frac{c}{\alpha}x^{\alpha}$$

For $x \geq 1$:

$$G(x) = \int_{-\infty}^{x} g(t)dt = \int_{0}^{1} ct^{\alpha-1}dt + \int_{1}^{x} ce^{-t}dt = \left[\frac{c}{\alpha}t^{\alpha}\right]_{0}^{1} - [ce^{-t}]_{1}^{x} = \frac{c}{\alpha} - ce^{-x} + ce^{-1}$$

This gives:

$$G(x) = \begin{cases} \frac{c}{\alpha}x^{\alpha}, & 0 < x < 1 \\ \frac{c}{\alpha} - ce^{-x} + ce^{-1}, & 1 \leq x \end{cases}$$

We now want to find the inverse transform. We again split into different cases.

Inverse transform of the, $0 < x < 1$:

$$y = G(x) = \frac{c}{\alpha}x^{\alpha} \Rightarrow x = \left(\frac{\alpha}{c}y\right)^{1/\alpha}$$

Inverse transform, $x \geq 1$:

$$y = G(x) = c(\frac{1}{\alpha} - e^{-x} + e^{-1})$$

$$\frac{y}{c} - \frac{1}{\alpha} - e^{-1} = -e^{-x}$$

$$x = -\ln(-\frac{y}{c} + \frac{1}{\alpha} + e^{-1})$$

We note that g is not continous. This as:

$$g(1) = ce^{-1}$$

And

$$\lim_{x \to 1^-} g(x) = c$$

G(0) = 0 and:

$$\lim_{x \to 0^+} G(x) = 0$$

Further $G(1) = \frac{c}{\alpha}$ and

$$\lim_{x \to 1^-} G(x) = \frac{c}{\alpha}$$

Which means that the inverse is:

$$G^{-1}(y) = \begin{cases} \left(\frac{\alpha y}{c}\right)^{\frac{1}{\alpha}}, & 0 < y < \frac{c}{\alpha} \\ -\ln(-\frac{y}{c} + \frac{1}{\alpha} + e^{-1}), & \frac{c}{\alpha} \leq y \leq 1 \end{cases}, \quad y \in [0, 1]$$

We now need to find c: We need the integral over the whole parameter space to equal 1, since g(x) is a pdf.

$$1 = \int_{-\infty}^{\infty} g(x)dx = \int_0^1 cx^{\alpha-1}dx + \int_1^{\infty} ce^{-x}dx = \left[\frac{c}{\alpha}x^{\alpha}\right]_0^1 - \left[ce^{-x}\right]_1^{\infty} = c\left(\frac{1}{\alpha} + e^{-1}\right)$$

$$\Rightarrow c = \left(\frac{1}{\alpha} + e^{-1}\right)^{-1}$$

Problem A.2.b

We now want to use rejection sampling to sample from g

```
# Function for finding inverse of g when 0 < x < 1
g1_inv <- function(y, c, alpha) {
  return((alpha/c*y)^(1/alpha))
}

# Function for finding inverse of g when x >= 1
g2_inv <- function(y, c, alpha) {
  return(-log(-y/c + 1/alpha + exp(-1)))
}

# Inverse of g on [0,1], this is a "private" function
# so we trust that this is utilized correctly
g_inv <- function(y, c, alpha) {
  if (y <= c/alpha) { # If in part of g1
    return(g1_inv(y, c, alpha))
  }
  else { # If in part of g2
    return(g2_inv(y, c, alpha))
  }
}

# Calculates the c constant in the pdf of g
find.c.of.g <- function(alpha) {
  return((1/alpha + exp(-1))^(-1))
}

# Method for drawing from pdf of g, given alpha and given sample size n
# Use inverse sampling technique
rg <- function(alpha, n = 1) {
 y <- runif(n)
 c = find.c.of.g(alpha)
 x = unlist(lapply(y, function(y) g_inv(y, c, alpha)))
 return(x)
}

# Method defining the g pdf (for use later and for plotting)
g <- function(x, alpha) {
  c = find.c.of.g(alpha)
  if (x <= 0) {return(0)}
  if (x < 1) {
      return(c*x^(alpha - 1))
  } else {
      return(c*exp(-x))
  }
}
```

We note that some theoretical values to compare to are:

$$E_g(x) = \int_{-\infty}^{\infty} xg(x)dx = \int_0^1 xg(x)dx + \int_1^{\infty} xg(x)dx = \int_0^1 cx^{\alpha}dx + \int_1^{\infty} cxe^{-x}dx$$
$$= c\left(\left[\frac{x^{\alpha+1}}{\alpha+1}\right]_0^1 + \left[-e^{-x}(1+x)\right]_1^{\infty}\right) = c\left(\frac{1}{\alpha+1} + 2e^{-1}\right)$$

$$E_g(x^2) = \int_{-\infty}^{\infty} x^2g(x)dx = \int_0^1 x^2g(x)dx + \int_1^{\infty} x^2g(x)dx = \int_0^1 cx^{\alpha+1}dx + \int_1^{\infty} cx^2e^{-x}dx$$
$$= c\left(\left[\frac{x^{\alpha+2}}{\alpha+2}\right]_0^1 + \left[-e^{-x}(2+2x+x^2)\right]_1^{\infty}\right) = c\left(\frac{1}{\alpha+2} + 5e^{-1}\right)$$

$$Var_g(x) = c\left(\frac{1}{\alpha+2} + 5e^{-1}\right) - c^2\left(\frac{1}{\alpha+2} + 5e^{-1}\right)^2$$

```
# Functions to calculate theoretical values seen in formulas above

# Theoretical expectation
theoretical_expectation <- function(alpha){
  c <- find.c.of.g(alpha)
  return(c*(1/(alpha + 1)+2*exp(-1)))
}

# Theoretical variance
theoretical_variance <- function(alpha){
  c <- find.c.of.g(alpha)
  expectation_value <- theoretical_expectation(alpha)
  return(c*(1/(alpha+2)+5*exp(-1))-expectation_value^2)

}
```

We use our implementation to draw from g with different values for α.

We first start with $\alpha = 0.8$.

```
# Draw to validate results
n = 100000 # Number of points to draw
alpha = 0.8
draws = rg(alpha, n = n) # Drawing using inverse transform technique

# Compare variance to real variance
obs_var <- round(var(draws),4)
expected_var <- round(theoretical_variance(alpha), 4)
theo.expected <- round(theoretical_expectation(alpha), 4)
print(paste0("Observed variance is: ",
             obs_var,
             " Theoretical variance is: ",
             expected_var, " With alpha: ",
             alpha))
```

```
## [1] "Observed variance is: 0.7166 Theoretical variance is: 0.7206 With alpha: 0.8"
```

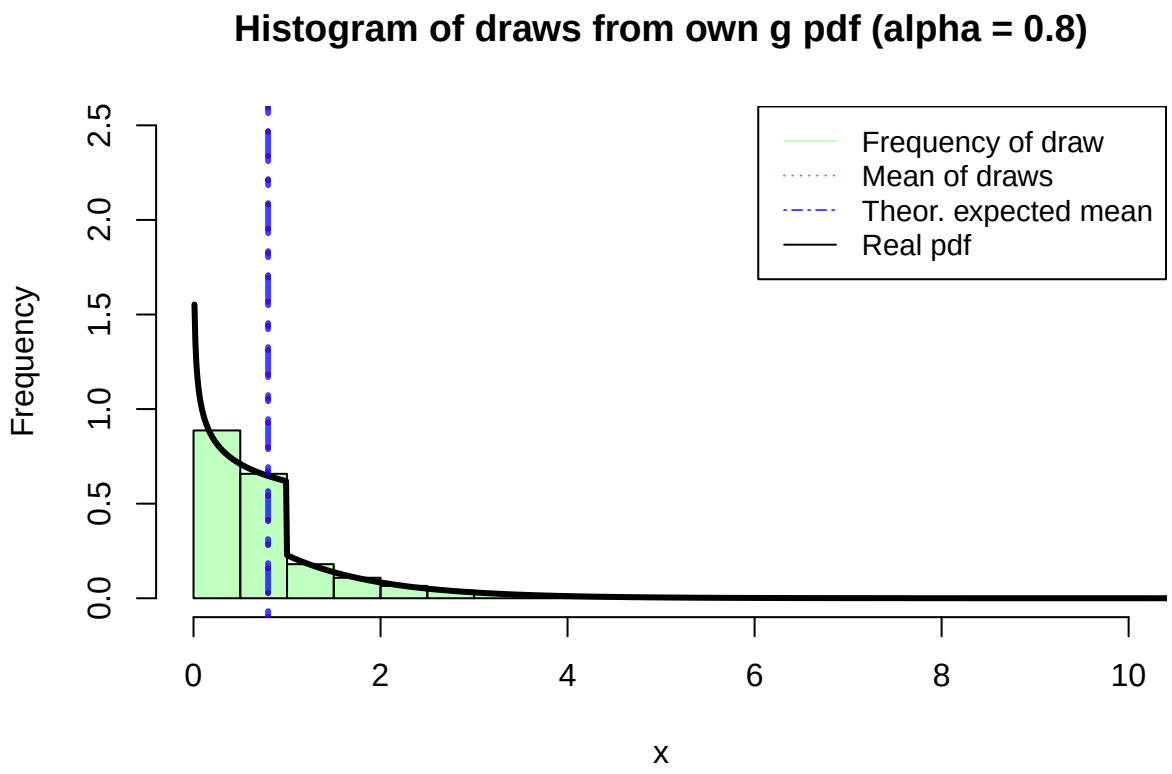

Figure 3: Draw from g with alpha = 0.8, n=100000

Results are illustrated in Figure 3. From the figure: The theoretical expectation seems to match observed mean. And the Frequency distribution seems to be matching theoretical pdf

Secondly we try $\alpha = 0.1$.

```
# Draw to validate results
n = 100000 # Number of points to draw
alpha = 0.1
draws = rg(alpha, n = n) # Drawing using inverse transform technique

# Compare variance to real variance
obs_var <- round(var(draws),4)
expected_var <- round(theoretical_variance(alpha), 4)
theo.expected <- round(theoretical_expectation(alpha), 4)
print(paste0("Observed variance is: ",
             obs_var,
             " Theoretical variance is: ",
             expected_var,
             " With alpha: ",
             alpha))
```

```
## [1] "Observed variance is: 0.202 Theoretical variance is: 0.1982 With alpha: 0.1"
```

Results are illustrated in Figure 4. From the figure: The theoretical expectation seems to match observed mean. And the Frequency distribution seems to be matching theoretical pdf

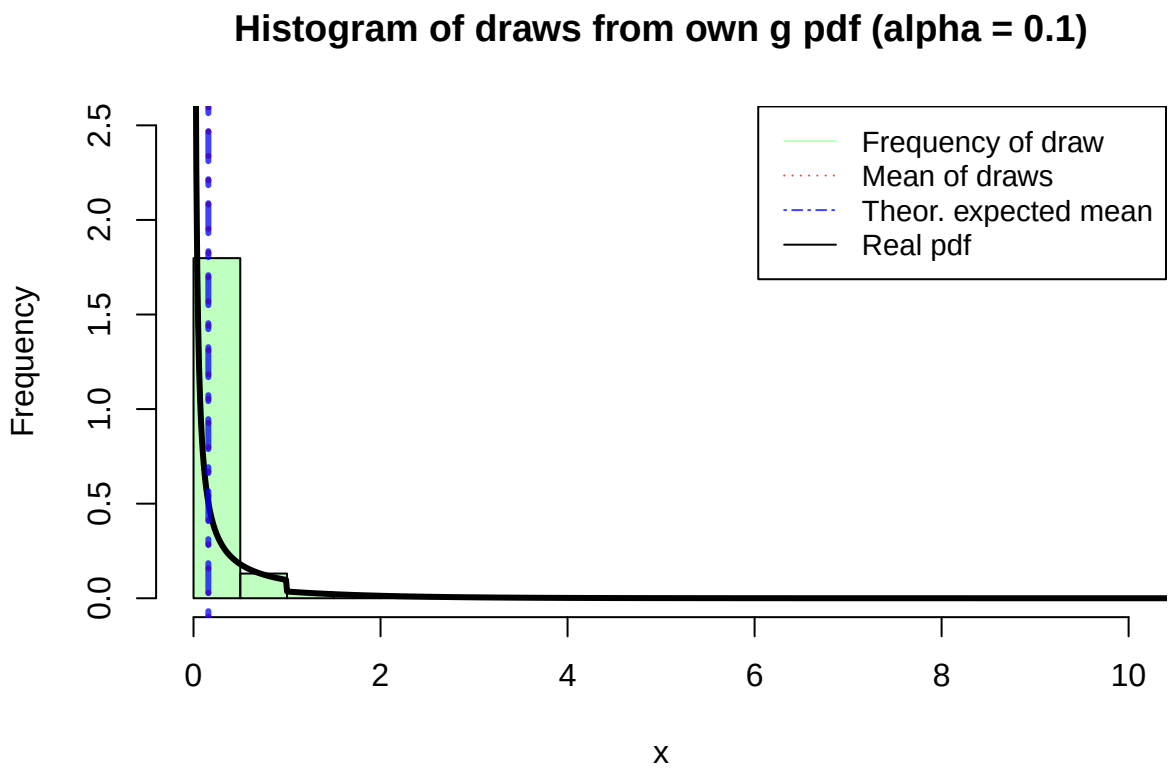

Figure 4: Draw from g with alpha = 0.1, n=100000

Problem A.3

Want to create method to draw from normal distribution. Use Box-Muller.

```
rnormStandard <- function(n) {
  x1 = runif(ceiling(n/2)) * 2*pi
  x2 = my.rexp(lambda = 0.5, n = ceiling(n/2))

  y1 = sqrt(x2)*cos(x1)
  y2 = sqrt(x2)*sin(x1)
  y = c(y1, y2)[1:n]
  return(y)
}

# For plotting
normal.standard.pdf <- function(x) {
  return(1/sqrt(2*pi)*exp(-0.5*x^2))
}
```

To validate our results we draw 10000 samples.

```
# Draw to validate results
n = 100000 # Number of points to draw
draws = rnormStandard(n) # Drawing using inverse transform technique
```

We compare the sample mean and variance to theoretical mean and variance.

```
# Compare variance to real variance
obs_var <- round(var(draws),4)
```

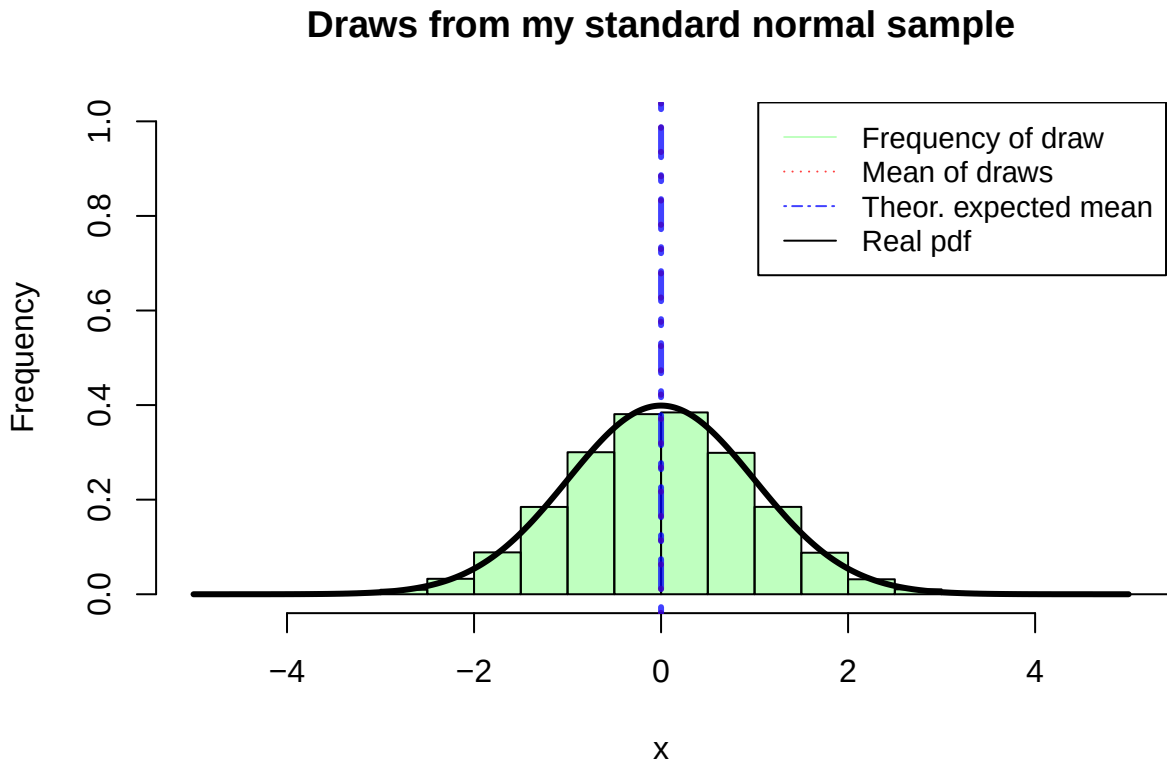

Figure 5: Draw from our standard normal function.

```
expected_var <- 1
theo.expected <- 0
print(paste0("Observed variance is: ",
             obs_var,
             " Theoretical variance is: ",
             expected_var))
```

```
## [1] "Observed variance is: 0.9991 Theoretical variance is: 1"
```

From output above variance seems to be matching.

The sample is illustrated in figure 5. The observed sample mean seems to match theoretical mean. The observed sample also seems to match the standard normal pdf.

Problem A.4

Want to make multivariate normal sample function. Want to draw:

$$\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Say $\mathbf{X}$ has dimension (dx1)

Can use facotrization $\mathbf{A}$ of $\boldsymbol{\Sigma}$: s.t. $\mathbf{A}^T\mathbf{A} = \boldsymbol{\Sigma}$

If we first sample d iid standard normal samples and organize them in $\mathbf{Z}$. Then:

$$\mathbf{X} = \mathbf{A}^T\mathbf{Z} + \boldsymbol{\mu} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Write code to draw:

```
my.multivariate.normal <- function(d, n, mu = NULL, sigma = NULL) {
  # If no argument given for mu and sigma the standard multivariate normal.
  # d is dimension
  # n is number of samples

  # Some argument checking, and creation of default arguments
  if (is.null(mu)) {
    mu = rep(0L, d)
  }

  if (is.null(sigma)) {
    sigma = diag(1, nrow=d)
  }
  # Assume sigma is valid cov matrix, that will say symmetric and pos. def
  l = rep(NA, n)

  C = chol(sigma) # Transposed cholesky of sigma
  l = sapply(l, function(x) t(C) %*% rnormStandard(d) + mu) # Do the transformation

  return(t(l))
}
```

Want to evaluate our draw:

Empirical coviarance:

$$\mathbf{Q} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

Can use matrix norms to compare difference, so use the frobenius norm.

$$||A||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}$$

Use 2-norm to evaluate error in estimated mean.

Run 1000 instances with random instanced of covariance matrix and mean vector to to check our sampling. Then calculate mean percentage error.

```
# Function to generate random covariance matrix
random.covariance.matrix <- function(d){
  A <- matrix(runif(d^2)*2-1, ncol=d)
```

```r
  return(t(A) %*% A)
}

# Function to calculate frobenius error difference between a
# samples genereated from randomly selected covariance and a mu, versus estimated
multiple_cov_checks <- function(n, d){
  # Crate random covariance
  sigma.test <- random.covariance.matrix(d)
    mu.test = runif(d) # Select random mu
  mnorm.sample <- my.multivariate.normal(d, n, mu = mu.test, sigma = sigma.test) # Sample
  mnorm.sample.list <- lapply(1:nrow(mnorm.sample), function(i) mnorm.sample[i,]) # Each row to list el
  mu.est <- apply(mnorm.sample, 2, mean)

  empirical.cov = 1/(n-1)*Reduce("+",
                                 lapply(mnorm.sample.list,
                                        # Calculate empirical covariance matrix
                                        function(x) (x-mu.est) %*% t(x-mu.est)))

  # Percentage error
  perc_error_cov = sum((sigma.test - empirical.cov)^2)^0.5  / sum((sigma.test)^2)^0.5
  perc_error_mean = sum((mu.test - mu.est)^2)^.5 / sum(mu.test^2)^.5
  return(c(perc_error_cov, perc_error_mean))
}

# Run many 1000 instances of multiple_cov_checks. Select 1000 samples each time, in dimension 3

# Uncomment if you want to run
#library(future.apply)
#plan(multiprocess)
#percentage_errors <- future_lapply(rep(NA, 1000), function(x){
#  return(multiple_cov_checks(10000, 3))
#})
#save(percentage_errors, file="percentage_errors_mnorm.RData")
load("percentage_errors_mnorm.RData")

percentage_mean_errors <- lapply(percentage_errors, function(x) x[2])
percentage_cov_errors <- lapply(percentage_errors, function(x) x[1])

percentage_mean_errors <- unlist(percentage_mean_errors)
percentage_cov_errors <- unlist(percentage_cov_errors)

# Calculate mean percentage error
mean_percentage_mean_errors <- mean(percentage_mean_errors)*100
mean_percentage_cov_errors <- mean(percentage_cov_errors)*100
```

Giving us a mean percentage of error (in 2-norm) on mean vector of: 1.6514138% (mean_percentage_mean_errors)

And giving us a mean percentage error (in frobenius norm) in estimation of covariance vector of: 1.2607942%(mean_percentage_cov_errors)

This is a relatively low error.

Problem B

The goal of problem B is to create a sampling function from the gamma distribution. We split this into three cases: $0 < \alpha < 1$, $\alpha = 1$, $\alpha > 1$.

Problem B.1

First look at $0 < \alpha < 1$:

Problem B.1.a

In rejection sampling we need to find a constant c such that $\frac{f(x)}{g(x)} \leq c$ for all c, where we use $g(x)$ is as in Problem A.

$$f(x) = \begin{cases} \frac{1}{\Gamma(\alpha)} \cdot x^{\alpha-1}e^{-x}, & 0 < x \\ 0, \text{otherwise} \end{cases}$$

Where $\alpha \in (0, 1)$

We prefer having c as small as possible, as that gives us a greater chance of accepting proposed values.

For $0 < x < 1$:

$$\frac{f(x)}{g(x)} = \frac{\frac{1}{\Gamma(\alpha)} \cdot x^{\alpha-1}e^{-x}}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot x^{\alpha-1}} = \frac{e^{-x}}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot \Gamma(\alpha)} \leq \frac{1}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot \Gamma(\alpha)} = \frac{(\frac{1}{\alpha} + e^{-1})}{\Gamma(\alpha)}$$

For $x \geq 1$

$$\frac{f(x)}{g(x)} = \frac{\frac{1}{\Gamma(\alpha)} \cdot x^{\alpha-1}e^{-x}}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot e^{-x}} = \frac{x^{\alpha-1}}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot \Gamma(\alpha)} \leq \frac{1}{(\frac{1}{\alpha} + e^{-1})^{-1} \cdot \Gamma(\alpha)} = \frac{(\frac{1}{\alpha} + e^{-1})}{\Gamma(\alpha)}$$

where the inequality follows from the fact that $\alpha \in (0, 1)$.

This means that we can choose $c = \frac{(\frac{1}{\alpha}+e^{-1})}{\Gamma(\alpha)}$, where α is a known constant.

We know that the acceptance probability in the rejection sampling algorithm is c^{-1} which means that the acceptance probability is $\frac{\Gamma(\alpha)}{\frac{1}{\alpha}+e^{-1}}$.

Problem B.1.b

```
# PDF of gamma with beta = 1
f <- function(x, alpha) {
  if (x > 0) {
    return(1/gamma(alpha) * x^(alpha - 1) * exp(-x))
  }
  return(0)
}

# Calculates the c mentioned above
calcKonst <- function(alpha) {
  # Calculate the constant s.a. f(x)/g(x) < constant, in the rejections sampling
  return((find.c.of.g(alpha)*gamma(alpha))^(-1))
  # find.c.of.g is the nominator in the expressions for c in the theory above
}
```

```r
my.rgamma_alpha_smaller1 <- function(alpha, n) {
  # Sample from a gamma distribution with beta = 1 and some alpha between zero and one.
  threshold <- function(x) f(x, alpha)/(calcKonst(alpha)*g(x, alpha)) # Find the threshold for the
  #rejection sampling, it's called alpha in the lecture notes, but since alpha is already
  #used we called it threshold instead

  if (alpha <= 0 | alpha >= 1) stop("Invalid alpha!!!")

  result <- c() # Vector to store our results in

  nLeft = n  # Number of values we still need to sample
  while(nLeft > 0) {

    u <- runif(ceiling(1/calcKonst(alpha)*nLeft))  # We try to approximate the number of tries
    #we will need to get n samples, we do that by using the acceptance probability
    gsample <- rg(alpha, length(u))

    thresholdValues <- sapply(gsample, threshold) # Calculate the threshold values

    result <- c(result, gsample[u < thresholdValues]) # Check if the sampled u's are below the
    #threshold, if TRUE we keep them

    nLeft = n - length(result)  # Correct the number of values we still need to sample
  }

  return(result)
}

alpha <- 0.5
draws <- my.rgamma_alpha_smaller1(alpha, 10000)

if (abs(mean(draws) - 0.5) > 10^-1) { # Know that the mean of a gamma
  #distribution with beta = 1 is alpha
  print("Sampling gives wrong mean")
}

if (abs(var(draws) - 0.5) > 10^-1) { # Know that the variance of a gamma
  #distribution with beta = 1 is alpha
  print("Sampling gives wrong variance")
}
```

As wee can see from Figure 6 the sampled values correspond well to the theoretical pdf.

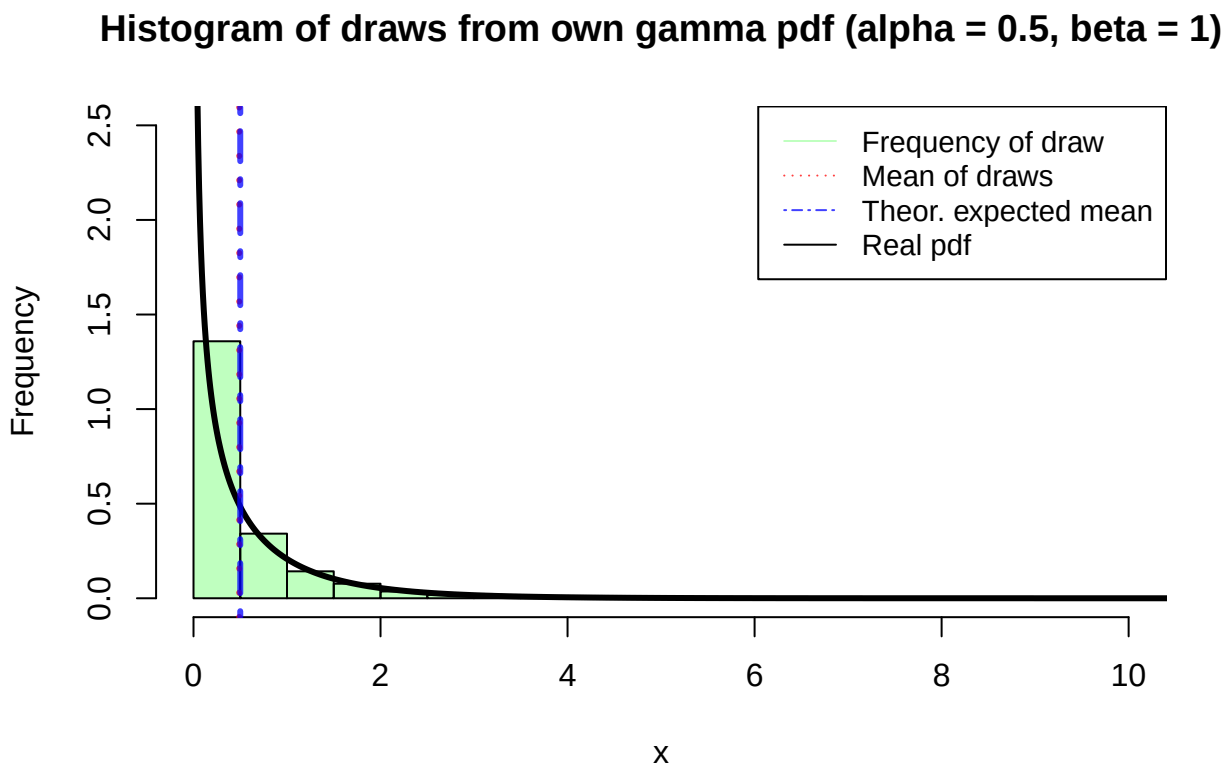

Figure 6: Draw from gamma with alpha = 0.5, beta = 1, n=10000. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf

Problem B.2

We now look at: $\alpha > 1$

Problem B.2.a

To use rejection sampling we need to find constants a, b_- and b_+. We start with a defined as:

$$a = \sqrt{\sup_x f^*(x)}$$

To find the supremum of f^* we will find where the derivative of f^* is 0, or equivalently where the derivative of $log(f^*)$ is 0.

$$\frac{\partial}{\partial x} log(f^*)(x) = \frac{\partial}{\partial x}(\alpha - 1)log(x) - x = \frac{\alpha - 1}{x} - 1 = 0$$

gives a zero point at $x = \alpha - 1$.

For $\alpha = 4$ we get $f^*(\alpha - 2) = f^*(2) = 1.08$, $f^*(\alpha - 1) = f^*(3) = 1.34$, $f^*(\alpha) = f^*(4) = 1.17$. This relationship will be the same for all α so we see that $x = \alpha - 1$ is the supremum.

This gives:

$$a = \sqrt{\sup_x f^*(x)} = \sqrt{f^*(\alpha - 1)} = \sqrt{(\alpha - 1)^{\alpha - 1} e^{\alpha - 1}}$$

Next is b_-:

$$b_- = \sqrt{\sup_{x \leq 0}(x^2 f^*(x))} = 0$$

since f^* is undefined for $x < 0$.

Lastly we have b_+:

$$b_+ = \sqrt{\sup_{x \geq 0} x^2 f^*(x)}$$

We will again use the method of finding the zero points of the derivative of $log(x^2 f^*(x))$:

$$\frac{\partial}{\partial x} log(x^2 f^*)(x) = \frac{\partial}{\partial x}[2log(x) + (\alpha - 1)log(x) - x] = \frac{2}{x} + \frac{\alpha - 1}{x} - 1 = 0$$

which gives us $x = \alpha + 1$.

We again set $\alpha = 4$ and find: $(\alpha)^2 f^*(\alpha) = 4^2 f^*(4) = 18.75$, $(\alpha + 1)^2 f^*(\alpha + 1) = 5^2 f^*(5) = 21.05$, $(\alpha + 2)^2 f^*(\alpha + 2) = 6^2 f^*(6) = 19.27$, so we see that $x = \alpha + 1$ is the supremum.

This gives:

$$b_+ = \sqrt{\sup_{x \geq 0} x^2 f^*(x)} = \sqrt{(\alpha + 1)^2 f^*(\alpha + 1)} = \sqrt{(\alpha + 1)^{\alpha+1} e^{\alpha+1}}$$

We define f^* as a function in R

```
fstar.log <- function(x, alpha) {
  # f* in log scale, it accepts variables x that is not in log scale
  if (x > 0) {
    return(log(x)*(alpha - 1) - x)
  } else {
    return(1)
  }
}
```

Problem B.2.b

Since, for α bigger than ~ 150, a and b_+ are bigger than our computers can handle we want to run the algorithm in log-scale instead. In the normal algorithm we want to sample $x_1 \sim Unif(0, a)$ and $x_2 \sim Unif(b_-, b_+)$, but since we can't handle a and b_+ we want to sample $log(x_1)$ and $log(x_2)$ instead. To be able to sample from these variables we use the formula to find the pdf of a function of another stochastic variable.

We have $u = g(x_1) = log(x_1)$ and $v = g(x_2) = log(x_2)$, which means that equivalently we have $x_1 = g^{-1}(u) = e^u$ and $x_2 = g^{-1}(v) = e^v$.

We have the formula

$$f_u(u) = f_{x_1}(g^{-1}(u)) * \frac{\partial}{\partial u} g^{-1}(u)$$

Since $x_1 \sim Unif(0, a)$ we have $f_{x_1}(x_1) = \frac{1}{a}$.

$$\frac{\partial}{\partial u} g^{-1}(u) = \frac{\partial}{\partial u} e^u = e^u$$

This gives $f_u(u) = \frac{1}{a} * e^u$.

To sample from u we use the inversion method, so we find the cdf:

$$F_u(u) = \int_{-\infty}^{u} f_u(u) = \frac{1}{a} e^u = f_u(u)$$

Which gives us $u = log(a) + log(y)$. This means that if we want to sample from u we sample $y \sim Unif(0, 1)$ and let $u = log(a) + log(y)$.

Similarly for v we get $v = log(b_+ - b_-) + log(w)$, where $w \sim Unif(0, 1)$.

We also need to translate the region C_f to work with u and v:

$$\{0 \leq x_1 \leq \sqrt{f^*(\frac{x_2}{x_1})}\}$$

$$\Leftrightarrow \{-\infty \leq u \leq 0.5 * log(f^*(\frac{x_2}{x_1})\}$$

$$\Leftrightarrow \{u \leq 0.5((\alpha - 1)(log(x_2) - log(x_1)) - \frac{x_2}{x_1})\}$$

$$\Leftrightarrow \{u \leq 0.5((\alpha - 1)(v - u) - e^{u-v})\}$$

Write R code to sample:

```r
inArea <- function(u, v, alpha) { # Checks if the points u and v (log(x_1) and log(x_2))
  # are in Cf
  return(u <= 0.5*(alpha - 1)*(v - u) - 0.5*exp(v - u))
}

my.rgamma_alpha_bigger1 <- function(alpha, n, seed = NULL) {
  # Samples n values from a gamma distribution with alpha > 1.
  # We acceps a seed parameter for reproducibility
  if(!is.null(seed)){ # Set seed
    set.seed(seed)
  }

  log.a = 0.5*fstar.log(alpha - 1, alpha) # Calculate log(a)
  log.b = log(alpha + 1) + 0.5*fstar.log(alpha + 1, alpha) # Calculate log(b)

  # We first try n values
  y = runif(n) # Random number to sample u
  w = runif(n) # Random number to sample v

  u = log.a + log(y) #u = log(x_1)
  v = log.b + log(w) #v = log(x_2)

  in.Cf <- sapply(1:n, function(i) inArea(u[i], v[i], alpha)) # Checks if each of the n values
  # we tried is inside of Cf
  result <- exp(v[in.Cf] - u[in.Cf]) # If the values are inside Cf
  # we include them and calculate x_2/x_1
  numberTrials = n # Number of trials we have used until now
  nLeft = n - sum(in.Cf) # Number of values we still have to sample

  approximateAcceptanceRate <- length(result)/n # We try to approximate how many tries
  # we need for each successful sample

  while(nLeft > 0) {
     if(approximateAcceptanceRate == 0){
      approximateAcceptanceRate = 1 # Just in case it is zero we set it to something
     }

    sample.interval <- ceiling(nLeft/approximateAcceptanceRate) # We calculate
    # approximately how many tries we need for the rest of the samples

    y = runif(sample.interval) # Random number to sample u
    w = runif(sample.interval) # Random number to sample v

    u = log.a + log(y) #u = log(x_1)
    v = log.b + log(w) #v = log(x_2)

    in.Cf <- sapply(1:sample.interval, function(i) inArea(u[i], v[i], alpha)) # Checks if
    # each of the n values we tried is inside of Cf
    result <- c(result, exp(v[in.Cf] - u[in.Cf])) # If the values are inside Cf we
    # include them and calculate x_2/x_1
    numberTrials = numberTrials + sample.interval # Number of trials we have used until now
```

```
    approximateAcceptanceRate <- length(result)/numberTrials # update our approximation
    # of number of trials for each success

    nLeft = nLeft - sum(in.Cf)
  }

  numOverflow <- length(result) - n # Since we might have gotten too many sample values
  # we need to know how many extra we got

  # Here we find out how many tries we used on the extra samples we got
  i <- length(in.Cf)
  sumTrues <- 0
  while(sumTrues < numOverflow) {
    sumTrues = sumTrues + in.Cf[i]
    i <- i - 1
  }

  numberTrials <- numberTrials - length(in.Cf) + i


 return(list(sample = result, NumberOfTrials = numberTrials) )
}
```

We test that we get correct mean and variance:

```
alpha = 10
draws = my.rgamma_alpha_bigger1(alpha, 100000)

if (abs(mean(draws$sample) - 10) > 10^-1) {  # Know that the mean of a gamma distribution
  # with beta = 1 is alpha
  print("Wrong mean")
}
if (abs(var(draws$sample) - 10) > 10^(-1)) { # Know that the variance of a gamma
  # distribution with beta = 1 is alpha
  print("Wrong variance")
}
```

We sample from our method. To check the number of trials needed to get 1000 samples for alpha values up to 2000.

```
alphas <- seq(from=1, to=2000, by=2)[-1]
n = 1000

# Uncomment if you want to run
#library(future.apply)
#plan(multiprocess) # Takes forever on one core, this allows R to run on multiple cores
#results <- future_lapply(alphas, function(alpha)
#    my.rgamma_alpha_bigger1(alpha, n, seed = alpha)) # future_lapply runs on several cores
#save(results, file="B2B_results.RData") # We save the results
## since the code takes a little while to run
load("B2B_results.RData") # Load the values we saved earlier

nTrials <- lapply(results, function(x) x$NumberOfTrials)
nTrials <- unlist(nTrials)
```

We plot the result in Figure 7. We see from Figure 7 that the number of iterations we need to get a sample

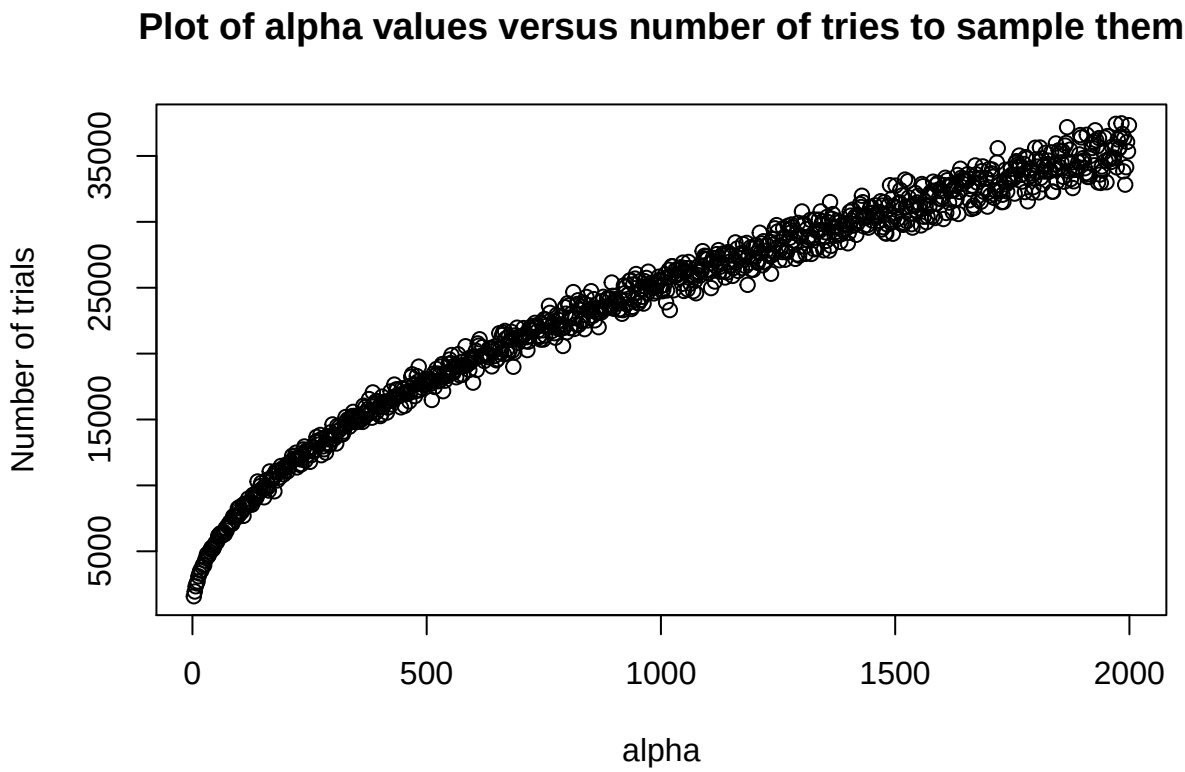

Figure 7: Number of random numbers sampled versus alpha value when sampling n = 100000 numbers from a gamma distribution with alpha according to the x axix and beta = 1

of 1000 values increases as α increases. It seems as if the slope flattens as α increases, so the biggest increase is in the lower values of α.

Figure 8 shows us that the sampled gamma distribution with $\alpha = 10$ and $\beta = 1$ corresponds well to the theoretical pdf.

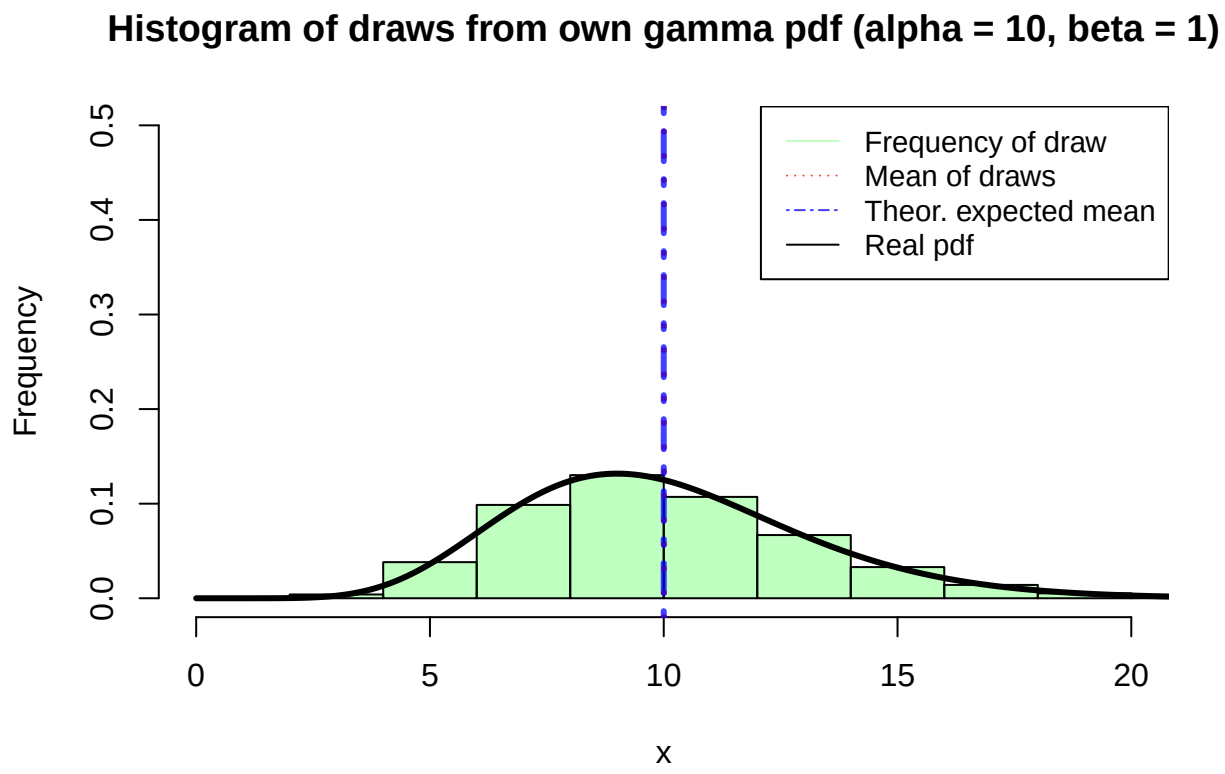

Figure 8: Draw from gamma with alpha = 10, beta = 1, n=100000. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf.

Problem B.3

We now create a method to draw from all possible alpha and beta values.

First use inversion sampling to draw from gamma with $\alpha = \beta = 1$ (we note that this is the same as what is done in Problem A)

```
# Sample from gamma distribution when alpha = 1, beta = 1
my.rgamma.alpha1 <- function(n){ # A gamma distribution with beta=1 is an
  #exponential distribution. We use the inverse sampling we did in
  #lectures to sample from exponential with alpha = 1
  # Sample from a gamme distribution with alpha = beta = 1
  u <- runif(n)
  x <- - log(u)
  return(x)
}
```

We note: As we can draw from:

$$X \sim gamma(\alpha, 1)$$

We can also draw from:

$$Y \sim gamma(\alpha, \beta)$$

This as:

$$f_{X/\beta}(x) = f_X(\beta x)\beta = \frac{1}{\Gamma(\alpha)}\beta^\alpha x^{\alpha-1}e^{-x} \sim gamma(\alpha, \beta)$$

So drawing X and calculating $Y = \frac{X}{\beta}$ would give $Y \sim gamma(\alpha, \beta)$

We are now able to create a method to draw from the pdf.

```
my.rgamma <- function(n, alpha = 1, beta = 1){
  # Sample from a gamma distribution with all valid alpha and beta values
  if(alpha < 1){
    x <- my.rgamma_alpha_smaller1(alpha, n)
  }else if(alpha > 1){
    x <- my.rgamma_alpha_bigger1(alpha, n)$sample
  }else{
    x <- my.rgamma.alpha1(n)
  }

  # Beta inverse transform
  x <- x/beta

  # Return result
  return(x)
}
```

We now test for some parameters.

```
alpha = 1
beta = 10
draws = my.rgamma(10000, alpha = alpha, beta = beta)

if (abs(mean(draws) - 0.1) > 0.1) { # We know that the mean of a gamma distribution is alpha/beta
  print("Wrong mean")
}
if (abs(var(draws) - 0.01) > 0.1) { # We know that the variance of a gamma distribution is alpha/beta^2
  print("Wrong variance")
}
```

Seems to work fine.

The results from the above sampling is illustrated in Figure 9. Our method seems to work fine.

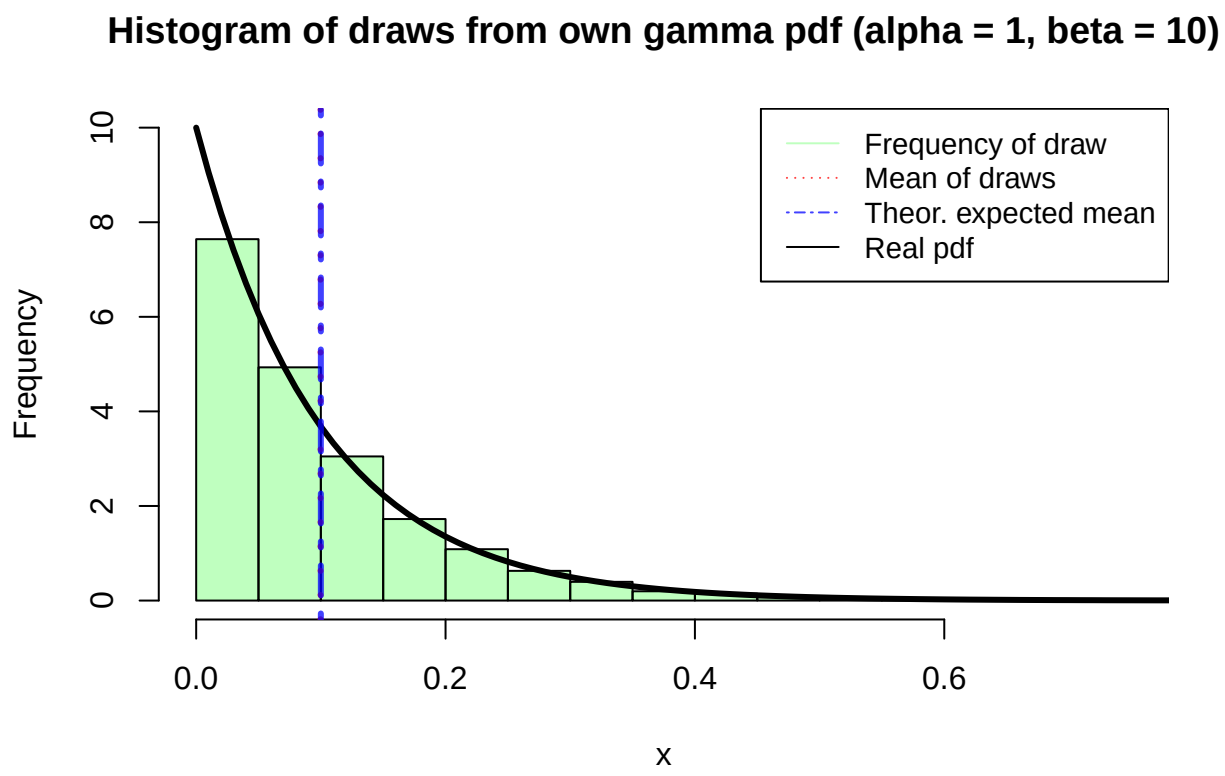

Figure 9: Draw from gamma with alpha = 1, beta = 10, n=10000. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf. Note scale is different from Figure 8

Problem C

Problem C.1.

We have that:

$$z_k \sim gamma(\alpha_k, 1), \qquad k = 1, ..., K$$

Thus:

$$f_{\mathbf{z}}(\mathbf{z}) = \prod_{i=1}^{K} \frac{z_i^{\alpha_i - 1} e^{-z_i}}{\Gamma(\alpha_i)}$$

We define $x_k = \frac{z_k}{v}$ for $k = 1, ..., K$ and $v = \sum_{i=1}^{K} z_k$. Want to find:

$$f_{\mathbf{x}}(x_1, .., x_{k-1}, v) \tag{1}$$

Use the following transform:

$$g(\mathbf{z}) = \begin{bmatrix} \frac{z_1}{v} \\ \frac{z_2}{v} \\ \vdots \\ \frac{z_{K-1}}{v} \\ \sum_{i=1}^{K} z_i \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K-1} \\ v \end{bmatrix} \tag{2}$$

We define $w = (x_1, x_2, ..., x_{K-1}, v)$. Then

$$g^{-1}(w) = g^{-1}(x_1, ..., x_{k-1}, v) = \begin{bmatrix} x_1 v \\ x_2 v \\ \vdots \\ x_{K-1} v \\ v(1 - \sum_{i=1}^{K-1} x_i) \end{bmatrix} \tag{3}$$

$$\begin{aligned} \mathbf{J}(g^{-1}) &= \begin{bmatrix} v & 0 & \dots & 0 & x_1 \\ 0 & v & \dots & 0 & x_2 \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & v & x_{K-1} \\ -v & -v & \dots & -v & 1 - \sum_{i=1}^{K-1} x_i \end{bmatrix} \\ &= \begin{bmatrix} v\mathbf{I}_{K-1} & \mathbf{x}_{K-1} \\ -v\mathbf{1}_{K-1}^T & 1 - \sum_{i=1}^{K-1} x_i \end{bmatrix} \end{aligned} \tag{4}$$

Using known determinant formulas:

$$det(\mathbf{J}(g^{-1})) = (1 - \sum_{i=1}^{K-1} x_i) - ((-v\mathbf{1}_{K-1}^T)(v\mathbf{I}_{K-1})^{-1}(\mathbf{x}_{K-1}))det(v\mathbf{I}_{K-1}) = v^{K-1} \tag{5}$$

This gives:

$$\begin{aligned} f_{\mathbf{z}}(g^{-1}(\mathbf{w})) &= \prod_{i=1}^{K} \left(\frac{(w_i)^{\alpha_i - 1} e^{-w_i}}{\Gamma(\alpha_i)} \right) \\ &= \prod_{i=1}^{K-1} \left(\frac{(x_i v)^{\alpha_i - 1} e^{-x_i v}}{\Gamma(\alpha_i)} \right) \left(\frac{\left(v(1 - \sum_{i=1}^{K-1} x_i)\right)^{\alpha_K - 1} e^{v(1 - \sum_{i=1}^{K-1} x_i)}}{\Gamma(\alpha_K)} \right) \end{aligned} \tag{6}$$

$$
\begin{aligned}
f_{\mathbf{w}}(\mathbf{w}) &= |\mathbf{J}_{\mathbf{w}}| f_{\mathbf{z}}(g^{-1}(\mathbf{w})) \\
&= v^{K-1} \cdot \prod_{i=1}^{K-1} \left(\frac{(x_i v)^{\alpha_i - 1} e^{-x_i v}}{\Gamma(\alpha_i)} \right) \left(\frac{\left(v(1 - \sum_{i=1}^{K-1} x_i) \right)^{\alpha_K - 1} e^{-v(1-\sum_{i=1}^{K-1} x_i)}}{\Gamma(\alpha_K)} \right) \\
&= v^{\Sigma_{i=1}^{K} \alpha_i - 1} e^{-v} \left(\frac{\prod_{i=1}^{K-1} x_i^{\alpha_i - 1}}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \right) \left(1 - \sum_{i=1}^{K-1} x_i \right)^{\alpha_K - 1}
\end{aligned} \tag{7}
$$

We want to integrate out v to get $f(x_1, ..., x_{K-1})$.

$$
\begin{aligned}
f(x_1, ..., x_{K-1}) &= \int_0^\infty f_{\mathbf{w}}(x_1, \ldots, x_{K-1}, v) dv \\
&= \int_0^\infty v^{\Sigma_{i=1}^{K} \alpha_i - 1} e^{-v} \left(\frac{\prod_{i=1}^{K-1} x_i^{\alpha_i - 1}}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \right) \left(1 - \sum_{i=1}^{K-1} x_i \right)^{\alpha_K - 1} dv \\
&= \left(\frac{\prod_{i=1}^{K} x_i^{\alpha_i - 1}}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \right) \left(1 - \sum_{i=1}^{K-1} x_i \right)^{\alpha_K - 1} \int_0^\infty v^{\Sigma_{i=1}^{K} \alpha_i - 1} e^{-v} dv
\end{aligned} \tag{8}
$$

Using the definition of the gamma function we have:

$$
\int_0^\infty x^{z-1} e^x dx = \Gamma(z). \tag{9}
$$

Thus equation (8) can be written:

$$
\begin{aligned}
&\left(\frac{\prod_{i=1}^{K-1} x_i^{\alpha_i - 1}}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \right) \int_0^\infty v^{\left(\sum_{i=1}^{K} \alpha_i - 1 \right)} e^{-v} dv \\
&= \left(\frac{\prod_{i=1}^{K-1} x_i^{\alpha_i - 1}}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \right) \left(1 - \sum_{i=1}^{K-1} x_i \right)^{\alpha_K - 1} \Gamma\left(\sum_{i=1}^{K} \alpha_i \right)
\end{aligned} \tag{10}
$$

Which is the pdf of the Dirichlet distribution.

Problem C.2.

Sample from Dirichlet, use transformation on gamma as in C.1.

```
my.rdirchelet <- function(alphas){
  K = length(alphas) # Find K
  z = sapply(alphas, function(a){
    a <- as.numeric(a) # Ensure alpha is numeric
    my.rgamma(n = 1, alpha = a, beta = 1) # Draw gammas given alpha
  })

  v = sum(z) # Find v
  return(z/v) # Do transformation as in C.1.
}
```

We also know (from https://en.wikipedia.org/wiki/Dirichlet_distribution) that when

$$(X_1, \dots X_k) \sim \text{Dirchlett}(\alpha_1, \dots, \alpha_k)$$

Then:

$$\tilde{\alpha}_i = \frac{\alpha_i}{\sum_{i=1}^{K} \alpha_i}$$
$$E(X_i) = \tilde{\alpha}_i$$
$$Var(X_i) = \frac{\tilde{\alpha}_i(1-\tilde{\alpha}_i)}{\Sigma_{j=1}^{K} a_j}$$

Use this to verify our method.

Do one trial uniformly selected alphas on interval (0.01, 10.01) using K = 100 and drawing 1000 samples from distribution. Then calculate mean distance from in samples from expected values (using 2-norm). And mean distance from variance (again with 2-norm).

```
# Set values
K = 100
n = 1000

# Select alpha
alphas = runif(n)*10 + 0.01 # To low causes rounding problems



# Takes a long time, so uncomment if you want to run
# Draw dirchlette
#xx <- lapply(1:n, function(x) my.rdirchelet(alphas))
#xx <- Reduce(rbind, xx)
#save(xx, file="DirchValidation.Rdata") # We save the results
## since the code takes a little while to run
load("DirchValidation.Rdata") # Load the values we saved earlier

awave = alphas/sum(alphas)

# Finding theoretical values using known alphas
theoretical_expected = awave
theretical_variance = awave*(1-awave)/sum(alphas)

# Calculating estimated mean and variance
```

```r
means <- apply(xx, 2, mean)
vars <- apply(xx, 2, var)

# Calculating percentage error in estimation from theoretical
perc_mean_error <- ((sum(theoretical_expected - means)^2)^0.5)/((sum(theoretical_expected)^2)^0.5)
perc_vars_error <- ((sum(theretical_variance - vars)^2)^0.5)/((sum(theretical_variance)^2)^0.5)
```

Giving a percentage error in mean estimation of 3.804872×10^{-17} (perc_mean_error). And giving a percentage error in var estimation of 0.0239093(perc_vars_error) which is quite low.

Problem D

D1

We want to sample from $f(\theta|y) \propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}$ by using rejection sampling with Unif(0, 1) as the proposal density, i.e. $g(\theta) = 1$.

Let's call $f^*(\theta) = (2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}$ and let $N = \int_{-\infty}^{\infty} f^*(\theta)d\theta$ be the normalising constant, which is independent of θ. Then $f(\theta|y) = \frac{f^*(\theta)}{N}$

To perform the rejection sampling we need to find a c such that $\frac{f(\theta|y)}{g(\theta)} \leq c$ for all $\theta \in (0, 1)$.

$$\frac{f(\theta|y)}{g(\theta)} = \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}{N}$$

This means that to find c we need to find the maximum of $f^*(\theta|y)$, as N does not depend on θ.

To find the maximum we will solve the equation $\frac{\partial f^*}{\partial \theta}(\theta) = 0$. By the product rule we get:

$$\frac{\partial f^*}{\partial \theta}(\theta) = \frac{\partial}{\partial \theta}(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}$$

$$= y_1(2+\theta)^{y_1-1}(1-\theta)^{y_2+y_3}\theta^{y_4} - (y_2+y_3)(2+\theta)^{y_1}(1-\theta)^{y_2+y_3-1}\theta^{y_4} + y_4(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4-1}$$

$$= (2+\theta)^{y_1-1}(1-\theta)^{y_2+y_3-1}\theta^{y_4-1}(y_1(1-\theta)\theta - (y_2+y_3)(2+\theta)\theta + y_4(2+\theta)(1-\theta))$$

Using $y_1 = 125, y_2 = 18, y_3 = 20, y_4 = 34$ we get the solutions:

$$\theta_1 = -2, \theta_2 = 1, \theta_3 = 0, \theta_4 = -0.55, \theta_5 = 0.626$$

Of these only θ_2, θ_3 and θ_5 are inside the definition interval. $f^*(\theta_2) = f^*(\theta_3) = 0$ and $f^*(\theta_5) = 1.84 * 10^{29}$, which gives us the maximum value of f^* and therefore the θ which gives the maximum value of $\frac{f(\theta|y)}{g(\theta)} = \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}{N}$.

We approximate N numerically by using R's `integrate` function, giving $N \approx 2.36 * 10^{28}$.

This gives us $c \approx \frac{1.84*10^{29}}{2.36*10^{28}} \approx 7.8$. (This is rounded up to ensure that inequality is held)

```
y <- c(125, 18, 20, 34)
f_theta_normalising_constant <- 2.36*10^28 # Function of y

f_theta <- function(theta, y) {
  return((2 + theta)^y[1] * (1 - theta)^(y[2] + y[3]) * theta^y[4] / f_theta_normalising_constant)
}

c = 7.8

sample1.f_theta <- function(y) {
  # Sample's one value from f(theta | y)
  # It also returns the number of tries we needed to get the sample

  num_tries <- 0

  x <- runif(1)
  alpha <- f_theta(x, y) / c  # Threshold value
  u <- runif(1)
  num_tries <- num_tries + 1
  while (u > alpha) { # Keep trying until we get a value that we accept as a sample of f
```

```
    x <- runif(1)
    alpha <- f_theta(x, y) / c
    u <- runif(1)
    num_tries <- num_tries + 1
  }

  result <- c(x, num_tries)

  return(result)
}

sample.f_theta <- function(n, y) {
  # Sample n values from f(theta | y)

  sample <- rep(NA, n) # Make a vector to store the sample

  sample <- sapply(sample, function(x) sample1.f_theta(y)) # Fill the vector with samples
  # and the number of tries for each sample

  sample <- as.data.frame(t(sample))
  colnames(sample) <- c("sample", "Num_tries")

  return(sample)
}
```

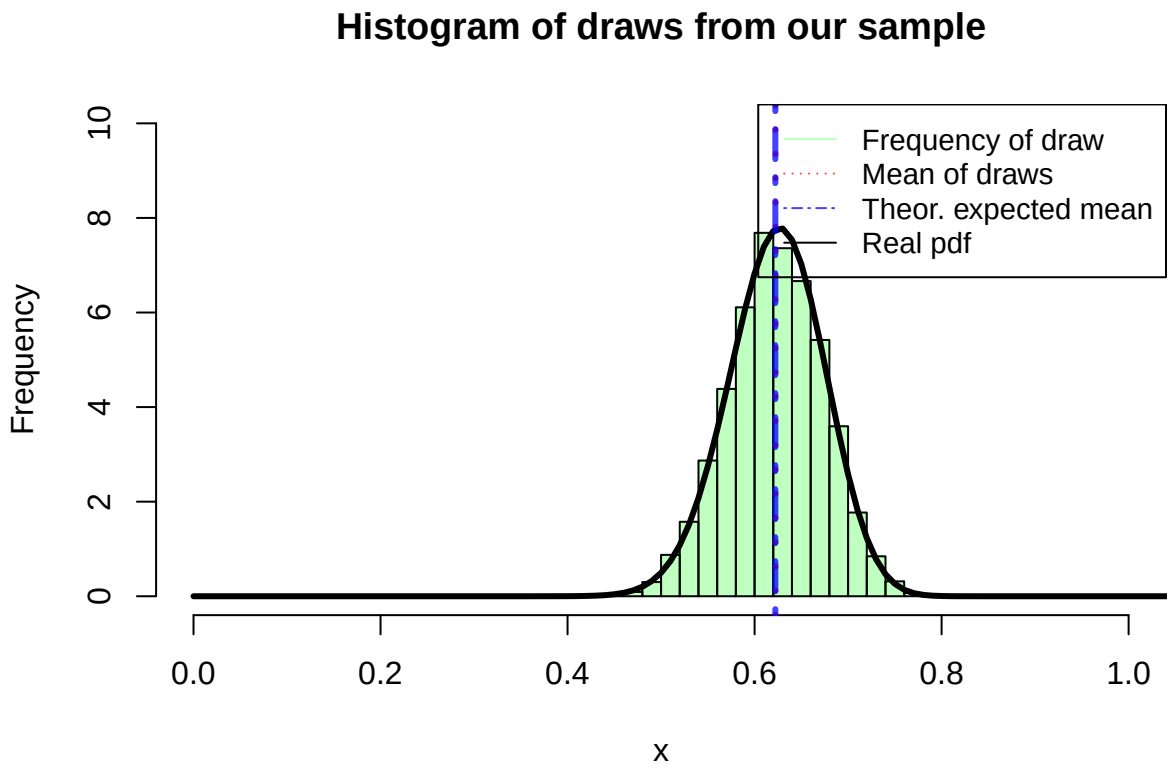

Figure 10: Draw from posterior, n=10000. Theoretical expectation seems to match observed mean. Frequency distribution seems to be matching theoretical pdf

D2

In this problem we want to estimate the mean of $f(\theta|y) = \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}}{N}$ i.e. $\int_{-\infty}^{\infty} \theta f(\theta|y)d\theta = E_f(\theta|y) \approx \frac{1}{M}\sum_{i=1}^{M}\theta_i$, where $\theta_1, \theta_2, ..., \theta_M$ are iid from $f(\theta|y)$.

```
draws <- sample.f_theta(10000, y) # Sample 10000 values from f(theta | y)
mu <- mean(draws$sample) # Estimate the expected value
```

The estimated mean is 0.6224472.

```
theoretical_mu <- integrate(function(t) t*f_theta(t, y), 0, 1)
```

The theoretical mean, found by numerical integration is 0.6221979, which is very close to the estimated one by sampling.

Figure 10 shows us that the sampled values corresponds very well with the pdf.

D3

```
mean_num_tries <- mean(draws$Num_tries) # We return the number of tries for each sample element, so to
```

The mean number of random numbers our algorithm needs to get one sample is 7.882 which is very close to the theoretical one, 7.8, i.e. c.

D4

Here we will use a different prior compared to the earlier parts of the problem. Instead of $Beta(1,1)$ i.e. $Unif(0,1)$ we will use $Beta(1,5)$.

$$f(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}}$$

Letting $\alpha = 1$ and $\beta = 5$ we get:

$$f(\theta) = \frac{\theta^0(1-\theta)^4}{\frac{\Gamma(1)\Gamma(5)}{\Gamma(6)}} = 5(1-\theta)^4$$

which means that the posterior becomes:

$$f(\theta|y) = \frac{f(y|\theta)f(\theta)}{f(y)} \propto \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4} * 5(1-\theta)^4}{\int_{-\infty}^{\infty} f(y|\theta)f(\theta)d\theta}$$

$$\propto (2+\theta)^{y_1}(1-\theta)^{y_2+y_3+4}\theta^{y_4}$$

.

We are now interested in estimating the mean of this new posterior with the samples we made of the old posterior in D2. To do that we will use importance sampling where the weights become:

$$w_i = \frac{f_{new}(\theta|y)}{f_{old}(\theta|y)} \propto \frac{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3+4}\theta^{y_4}}{(2+\theta)^{y_1}(1-\theta)^{y_2+y_3}\theta^{y_4}} = (1-\theta)^4$$

.

This gives the estimator

$$\mu_{IS} = \frac{\sum_i \theta_i(1-\theta_i)^4}{\sum_i (1-\theta_i)^4}$$

.

```
mu_IS <- function(sample) {
  nominator <- sum(sapply(sample, function(theta) theta*(1 - theta)^4))
  denominator <- sum(sapply(sample, function(theta) (1 - theta)^4))

  return(nominator/denominator)
}
importance_sampling_mean <- mu_IS(draws$sample)
```

which gives the estimate 0.5955834.

We can calculate the theoretical mean by numerical integration.

```
f_theta_new <- function(theta, y) {
  # The new posterior pdf, without the normalising constant
  return((2 + theta)^y[1] * (1 - theta)^(y[2] + y[3] + 4) * theta^y[4])
}

# Calculate the new normalising constant
normalising_constant_new <- integrate(function(t) f_theta_new(t, y), 0, 1)

# Calculate theoretical mean
theoretical_mu_new <- integrate(function(t)
    t*f_theta_new(t, y) / normalising_constant_new$value, 0, 1)
```

We see that the theoretical mean, 0.5959316 is very similar to the one estimated with importance sampling.

This is somewhat smaller than the estimation from the posterior with the uniform prior. This mainly comes from the $+4$ in the exponent of the $(1-\theta)$ factor in the newest posterior.