STAT 5701: Statistical Computing

Homework 1

Jingxiang Li

September 29, 2015

Problem 1.a

Algorithm 1: Draw Sample Given Density Function f

input : Sample size n

output: A vector consists of n i.i.d. observations from density function f

begin

- sample[n] = {0}, i = 0
- **while** i ≠ n **do**
  - draw x from Unif(−1, 1)
  - draw u from Unif(0, 1)
  - **if** u < $\frac{4}{3}f$(x) **then**
    - i = i + 1
    - sample[i] = x

Problem 1.b

R Code Chunk

```
f <- function (x) {
#     Calculate density value
#     Input:
#         x: numeric value
#     Output:
#         density value
    ifelse(x > -1 && x < 1,
           3 / 4 * (1 - x^2),
           0)
}

rquad <- function (n) {
#     Draw sample from f by rejection sampling
#     Input:
#         n: sample size
#     Output:
#         x.list: n i.i.d. observations from f
#         k.list: number of iterations used to produce each observation
    x.list <- numeric(length = n)
    k.list <- numeric(length = n)
    i <- 0
    iter <- 0
    while (i != n) {
        iter <- iter + 1
        x <- runif(n = 1, min = -1, max = 1)
        u <- runif(n = 1)
        if (u < 4 / 3 * f(x)) {
            i <- i + 1
            x.list[i] <- x
            k.list[i] <- iter
            iter <- 0
```

```
        }
    }
    return (list(x.list = x.list, k.list = k.list))
}
```

Problem 1.c

R Code Chunk

```
n <- 1000
mySample <- rquad(n)
hist(mySample$x.list)
mean(mySample$k.list)

## [1] 1.493
```

On average each realization requires 1.493 iterations.

Problem 2.a

R Code Chunk

```r
box.muller.trans <- function (mu, sigma) {
#     Box-Muller's method to generate N(mu, sigma) sample
#     Input:
#         mu: mean
#         sigma: standard deviation
#     Output:
#         a vector consists 2 i.i.d. observations from N(mu, sigma)
    u <- runif(n = 2)
    z <- numeric(length = 2)
    z[1] <- sqrt(-2 * log(u[1])) * cos(2 * pi * u[2])
    z[2] <- sqrt(-2 * log(u[1])) * sin(2 * pi * u[2])
    return (z * sigma + mu)
}

myrtnorm <- function (n, mu, sigma, a, b) {
#     Draw sample from truncated normal distribution by rejection sampling
#     Z ~ (X | a < X < b) where X ~ N(mu, sigma)
#     Input:
#         n: sample size
#         mu: mean
#         sigma: standard deviation
#         a: lower bound for the truncated dist
#         b: upper bound for the truncated dist
#     Output:
#         a vector of n i.i.d. observations
    t.list <- numeric(n)
    i <- 0
    while (i < n) {
        z.vec <- box.muller.trans(mu, sigma)
        for (z in z.vec) {
            if (a < z && z < b) {
                i <- i + 1
                t.list[i] <- z
            }
        }
    }
    return (t.list)
}
```

Problem 2.b

$$\frac{1}{\Phi(\frac{b-\mu}{\sigma}) - \Phi(\frac{a-\mu}{\sigma})}$$

Problem 2.c

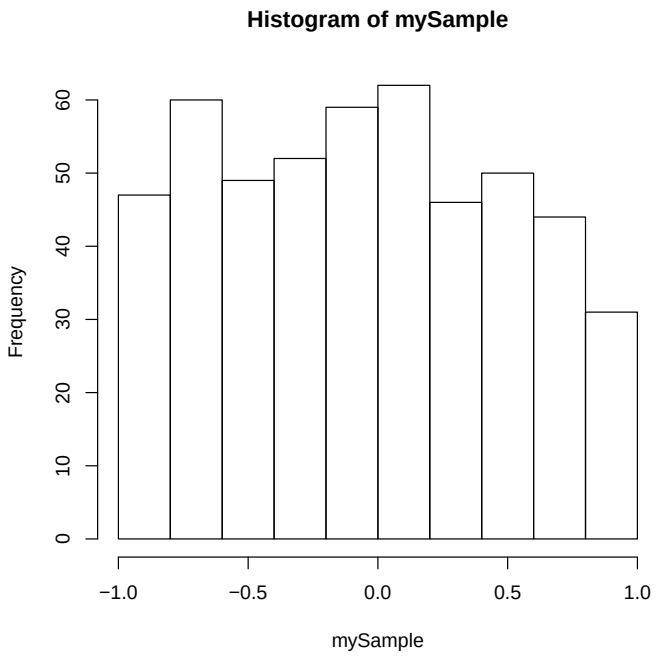
R Code Chunk

```
n <- 500
mu <- 0
sigma <- 1
a <- -1
b <- 1

iter.num <- 1 / (pnorm(q = b, mean = mu, sd = sigma) -
                     pnorm(q = a, mean = mu, sd = sigma))
iter.num

## [1] 1.464795

mySample <- myrtnorm(n = n, mu = mu, sigma = sigma,
                     a = a, b = b)
hist(mySample)
```



Problem 3.a

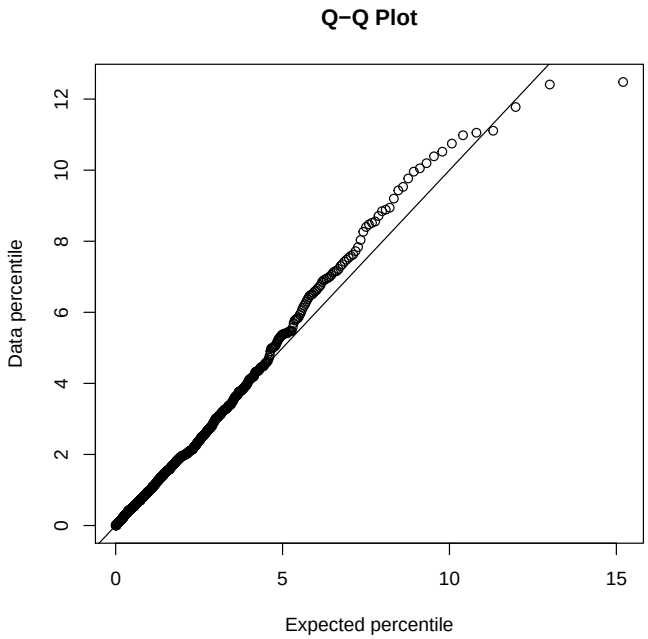
R Code Chunk

```
exp.quantile <- function (p, mu) {
#     Quantile function for Exp(mu)
#     Input:
#         p: probability
#         mu: mean
#     Output:
#         quantile value for p
    return (- log(1 - p) * mu)
}
myrexp <- function (n, mu) {
#     Draw sample from Exp(mu)
#     Input:
#         n: sample size
#         mu: mean
#     Output:
#         a vector consists of n i.i.d. observations from Exp(mu)
    return (exp.quantile(runif(n), mu))
}
```

Problem 3.b

R Code Chunk

```
adam.qqplot <- function (x.list, quant.func, ...) {
#     Draw qqplot (modified from Adam's code)
#     Input:
#         x.list: target sample
#         quant.func: quantile function for the target distribution
#         ...: parameters pass to the quant.func()
#     Output:
#         plot is generated
    n <- length(x.list)
    probs <- ppoints(n)
    plot(quant.func(probs, ...), quantile(x.list, probs),
         xlab="Expected percentile", ylab="Data percentile",
         main = "Q-Q Plot")
    abline(0, 1)
}
n <- 1000
mu <- 2
exp.list <- myrexp(n = n, mu = mu)
adam.qqplot(x.list = exp.list, quant.func = exp.quantile, mu = mu)
```



Problem 3.c

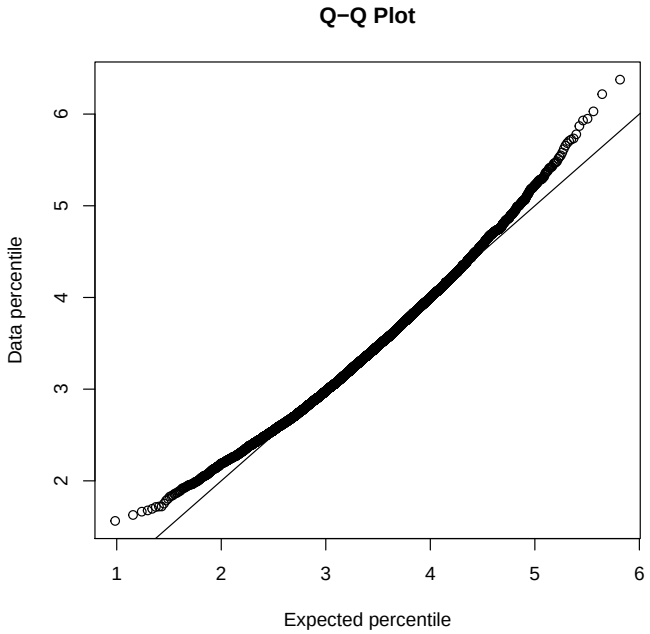
R Code Chunk

```
run.exp.sim <- function (n, mu, reps) {
#     Simulation for sample mean (Y_bar) of i.i.d. Exp(mu) random observations
#     Input:
#         n: sample size
#         mu: mean of the Exp dist
#         reps: replication times
#     Output:
#         return a vector of Y_bar realizations
#         draw qq plot to compare the dist of Y_bar and Normal dist
    ybar.list <- numeric(reps)
    for (i in 1 : reps)
        ybar.list[i] <- mean(myrexp(n, mu))
    adam.qqplot(x.list = ybar.list,
                quant.func = qnorm,
                mean = mu, sd = mu / sqrt(n))
    return (ybar.list)
}
```

Problem 3.d

R Code Chunk

```
my.sim <- run.exp.sim(n = 30, mu = 3.4, reps = 10000)
```



From the Q-Q plot we could see the sample mean shifts away from the line significantly in the tail parts, hence it's not that appropriate to apply normal approximation for inference. The sample size is not large enough.

Problem 4.a

R Code Chunk

```
mymvrnorm <- function(n, mu, Sigma) {
#     Generate n i.i.d. observations from N(mu, Sigma)
#     Input:
#         n: sample size
#         mu: mean vector
#         Sigma: variance matrix
#     Output:
#         sample.matrix (n x d): n i.i.d. observations from N(mu, Sigma)
    sigma.eig <- eigen(Sigma)
    sigma.sqrt <- sigma.eig$vectors %*%
        diag(sqrt(sigma.eig$values)) %*%
        t(sigma.eig$vectors)
    d <- dim(Sigma)[1]
    sample.matrix <- matrix(data = 0, nrow = n, ncol = d)
    for (i in 1 : d)
        sample.matrix[, i] <- myrtnorm(n = n, mu = 0, sigma = 1,
                                       a = -Inf, b = Inf)
    sample.matrix <- sample.matrix %*% sigma.sqrt
    sample.matrix <- sample.matrix + matrix(rep(mu, n), nrow = n, byrow = TRUE)
    return (sample.matrix)
}
```

Problem 4.b

i.

$$\mu(\bar{X}) = \mu \qquad \mathrm{Var}(\bar{X}) = \frac{\sigma^2}{n}$$

ii.

$$\mu(\bar{H}) = \mu \qquad \mathrm{Var}(\bar{H}) = \frac{\sigma^2}{n} + \frac{\sum_{k=1}^{n-1} \sum_{|i-j|=k} \sigma^2 \cdot 0.7^{|i-j|}}{n^2}$$

iii. $\bar{H}$ is worse than $\bar{X}$

Problem 4.c

R Code Chunk

```
n <- 50000
d <- 10
mu <- rep(68, d)
sigma <- 3
Sigma <- matrix(data = 0, nrow = d, ncol = d)
for (i in 1 : d) {
    for (j in 1 : d) {
        Sigma[i, j] = sigma^2 * 0.7^abs(i - j)
    }
}
sim.h <- mymvrnorm(n, mu, Sigma)
sim.hbar <- apply(sim.h, 1, mean)
sim.x <- mymvrnorm(n, mu, diag(rep(sigma^2, d)))
sim.xbar <- apply(sim.x, 1, mean)

mean(sim.hbar)

## [1] 68.01022

mean(sim.xbar)

## [1] 67.9996

var(sim.hbar)

## [1] 3.756895

var(sim.xbar)

## [1] 0.9074269
```

Simulation result shows that $\mu(\bar{X})$ and $\mu(\bar{H})$ are very close to μ, and $\text{Var}(\bar{H})$ is significantly larger than $\text{Var}(\bar{X})$, which supports my prevoius argument.