Problem 5.1

5.1(b)

```
mean.std.CI<-function(thsample){
  G <- dim(thsample)[1]
  thmean <- apply(thsample, 2, mean)
  thstd <- sqrt(apply(thsample, 2, var))
  thl <- thmean - 1.96 * thstd
  thu <- thmean + 1.96 * thstd
  thsort <- apply(thsample, 2, sort)
  indexx <- floor(c(0.025 * G, 0.975 * G))
  thL <- (thsort[indexx[1], ]+ thsort[indexx[1]+1, ])/2
  thU <- (thsort[indexx[2], ]+ thsort[indexx[2]+1, ])/2
  results <- c(thmean, thstd, thl, thu, thL, thU)
  return(results)
}


f1<-function(G){
# Call: c(thmean, thstd, thl, thu, thL, thU)
#       <- mean.std.CI(thsample)
# G is the bootstrap sample size
  x <- c(32., 46.4, 48.1, 27.7, 35.5, 52.6, 66.0, 41.3,
         49.9, 36.1, 50.0, 44.7, 48.2, 36.9, 40.8, 35.1,
         63.3, 42.5, 52.4, 40.9, 38.6, 43.2, 41.7, 35.6)
  n <- length(x)
  muMLE <- mean(x)
  si2MLE <- sum((x - muMLE) * (x - muMLE))/n
  siMLE <- sqrt(si2MLE)
  CVMLE <- siMLE/muMLE
  CV.star.sample <- matrix(0, G, 1)
  for(g in 1:G) {
     xstar <- rnorm(n, mean = muMLE, sd = siMLE)
     mustar <- mean(xstar)
     CVstar <- sqrt(sum((xstar - mustar) * (xstar -
               mustar))/n)/mustar
     CV.star.sample[g, 1] <- CVstar
  }
  M <- mean.std.CI(CV.star.sample)
  CVmean <- M[[1]]
  CVstd <- M[[2]]
  CVL <- M[[5]]
  CVU <- M[[6]]
  return(c(CVMLE, CVmean, CVstd, CVL, CVU))
}
set.seed(110)
f1(10000)
```

```
## [1] 0.2045256 0.1986571 0.0306177 0.1408569 0.2614154
```

5.1(c) 1

```
f2<-function(G)
{
     # Name: CS.Assignment5.1.parametric.median(G=20000)
     # Call: c(thmean, thstd, thl, thu, thL, thU)
     #       <- mean.std.CI(thsample)
     # G is the bootstrap sample size
     x <- c(32., 46.4, 48.1, 27.7, 35.5, 52.6, 66., 41.3,
          49.9, 36.1, 50., 44.7, 48.2, 36.9, 40.8, 35.1,
          63.3, 42.5, 52.4, 40.9, 38.6, 43.2, 41.7, 35.6)
     n <- length(x)
     muMLE <- mean(x)
     si2MLE <- sum((x - muMLE) * (x - muMLE))/n
     siMLE <- sqrt(si2MLE)
     thMLE <- median(x)
     th.star.sample <- matrix(0, G, 1)
     for(g in 1:G) {
     xstar <- rnorm(n, mean = muMLE, sd = siMLE)
          thstar <- median(xstar)
          th.star.sample[g, 1] <- thstar
     }
     M <- mean.std.CI(th.star.sample)
     thmean <- M[[1]]
     thstd <- M[[2]]
     thL <- M[[5]]
thU <- M[[6]]
     return(c(thMLE, thmean, thstd, thL, thU))
}
set.seed(111)
f2(10000)
```

```
## [1] 42.100000 43.701528  2.217601 39.370458 48.077605
```

5.1(c) 2

```
f3<- function(G){
# Name: CS.Assignment5.1.nonparametric.median(G=20000)
# Call: c(thmean, thstd, thl, thu, thL, thU)
#       <- mean.std.CI(thsample)
# G is the bootstrap sample size
  x <- c(32., 46.4, 48.1, 27.7, 35.5, 52.6, 66., 41.3,
     49.9, 36.1, 50., 44.7, 48.2, 36.9, 40.8, 35.1,
     63.3, 42.5, 52.4, 40.9, 38.6, 43.2, 41.7, 35.6)
  n <- length(x)
  p <- rep(1/n, n)
  th.star.sample <- matrix(0, G, 1)
  for(g in 1:G) {
     xstar <- sample(x, n, prob = p, replace = T)
     thstar <- median(xstar)
     th.star.sample[g, 1] <- thstar
}
  M <- mean.std.CI(th.star.sample)
  thmean <- M[[1]]
```

```
  thstd <- M[[2]]
  thL <- M[[5]]
  thU <- M[[6]]
  return(c(thmean, thstd, thL, thU))
}
set.seed(113)
f3(10000)
```

```
## [1] 42.547045  2.107853 38.600000 48.100000
```

```
mean.std.CI<-function(thsample){
  G <- dim(thsample)[1]
  thmean <- apply(thsample, 2, mean)
  thstd <- sqrt(apply(thsample, 2, var))
  thl <- thmean - 1.96 * thstd
  thu <- thmean + 1.96 * thstd
  thsort <- apply(thsample, 2, sort)
  indexx <- floor(c(0.025 * G, 0.975 * G))
  thL <- (thsort[indexx[1], ]+ thsort[indexx[1]+1, ])/2
  thU <- (thsort[indexx[2], ]+ thsort[indexx[2]+1, ])/2
  results <- c(thmean, thstd, thl, thu, thL, thU)
  return(results)
}

set.seed(12345)
Ass<- function(G) {
  x<-c(0,0,1,1,1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0)
  n <- length(x)
  thhat <- mean(x)
  th.star.sample <- matrix(0, G, 1)
  for(g in 1:G) {
    xstar <- rbinom(n, 1, thhat)
    thstar <- mean(xstar)
    th.star.sample[g] <- thstar
}
  M <- mean.std.CI(th.star.sample)
  results <- c(thhat, M)
  return(results)
}
Ass(10000)
```

```
## [1] 0.63333333 0.63241000 0.08879591 0.45837002 0.80644998 0.46666667 0.80000000
```