Statistical Methods for Data Science
CS 6313.001: Mini Project #3

Due on Tuesday March 7, 2017 at 4pm

Instructor: Pankaj Choudhary

Hanlin He / Lizhong Zhang (hxh160630 / lxz160730)

Contribution

Both team members made the same contribution in this project.

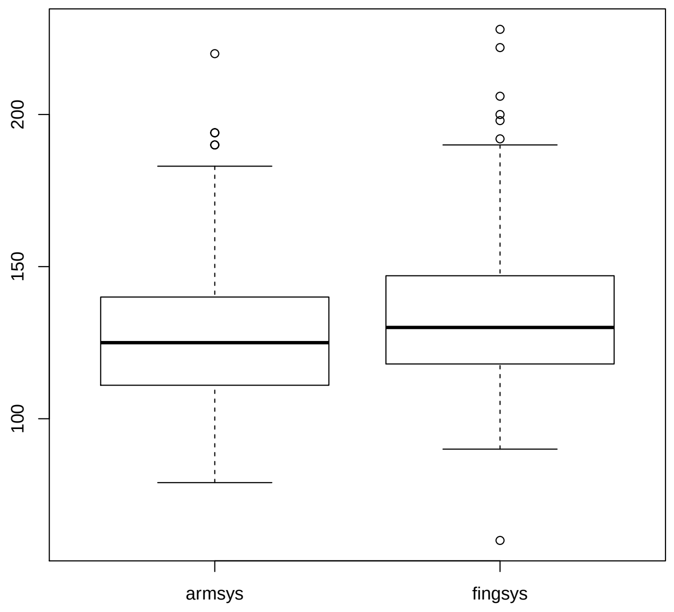
Section 1 Answers

Problem 1.1 Blood Pressure Measurement

(a) Boxplots

The boxplots are shown in fig. 1:

Figure 1: Boxplots

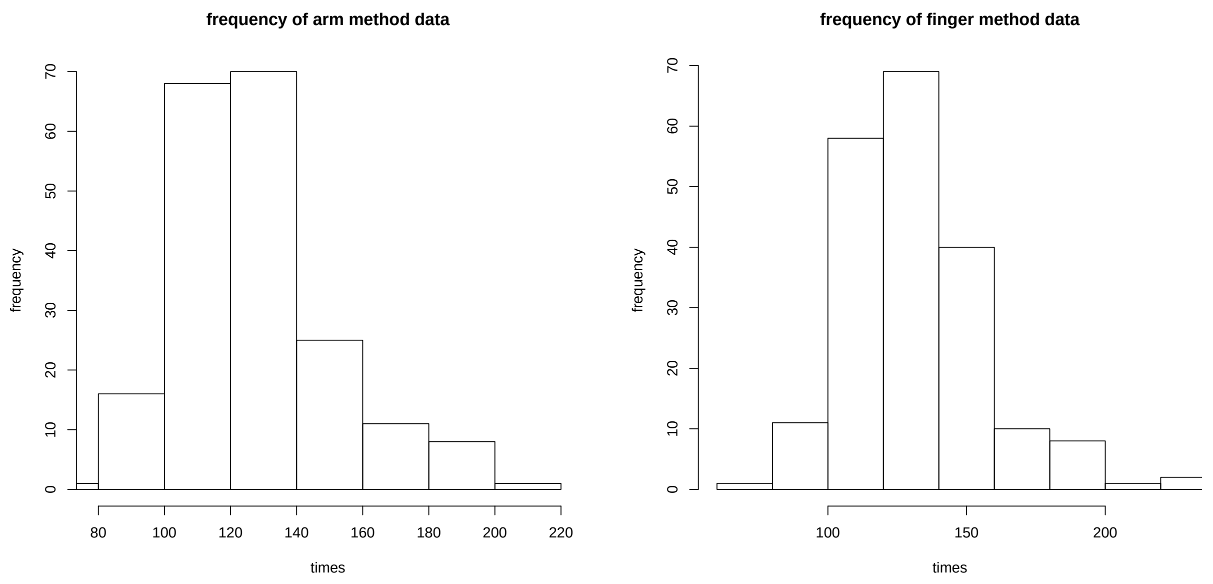


From my point of view, we could see that based on boxplots, two distributions are not similar. The data in Finger method is slightly larger than the data in Arm method, and Finger method is slightly right-skewed.

(b) Histograms & QQplot

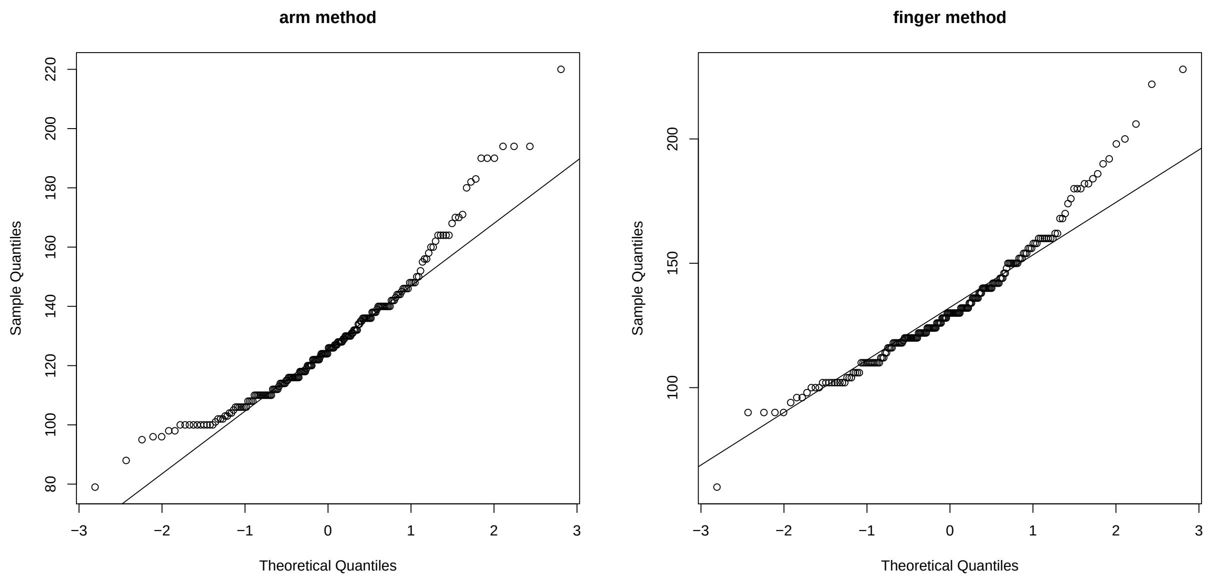
The histograms are shown in fig. 2:

Figure 2: Histogram



The QQplots are shown in fig. 3:

Figure 3: QQplot



Based on histograms and QQplots, we could see that the assumptions of normality are not reasonable.

In histograms, we could see that both distributions are slightly right-skewed. And in QQplots, both distributions have many outliers.

(c) CI

The CI is $[-0.5208386, 9.1108386]$.

According above result, we can conclude that the two methods have identical means. But this is a borderline case. We need more data for a more precise conclusion.

Problem 1.2

CI for the same sample size n of different p is shown in fig. 14

Figure 4: Confidence Intervals for $n = 50$ and Different p

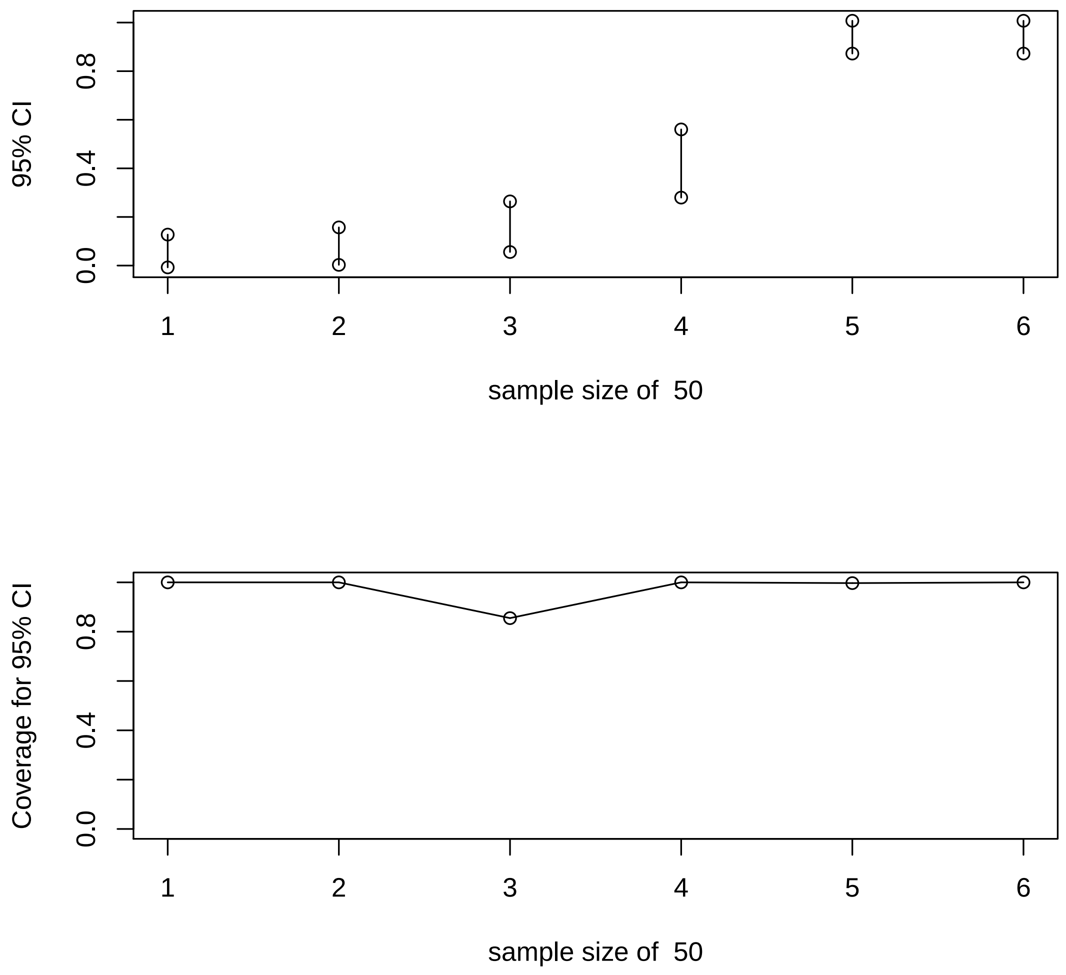
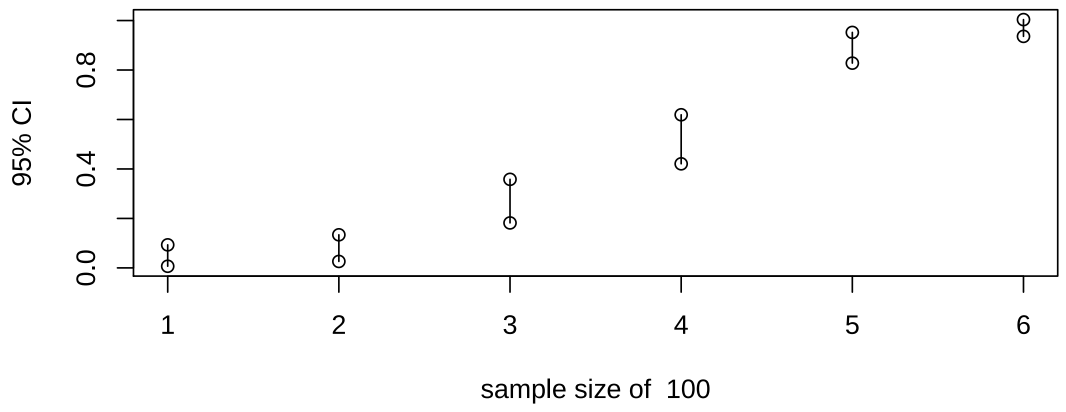


Figure 5: Confidence Intervals for Same $n = 100$ and Different p



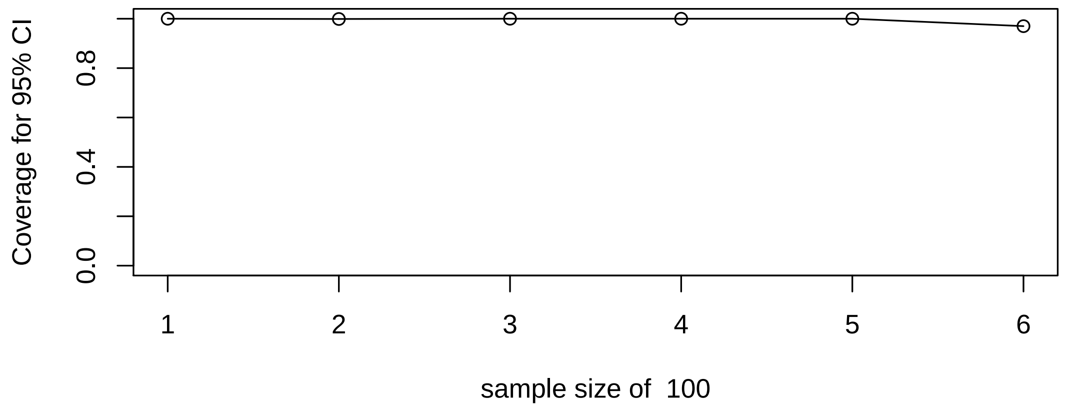


Figure 6: Confidence Intervals for Same $n = 300$ and Different p

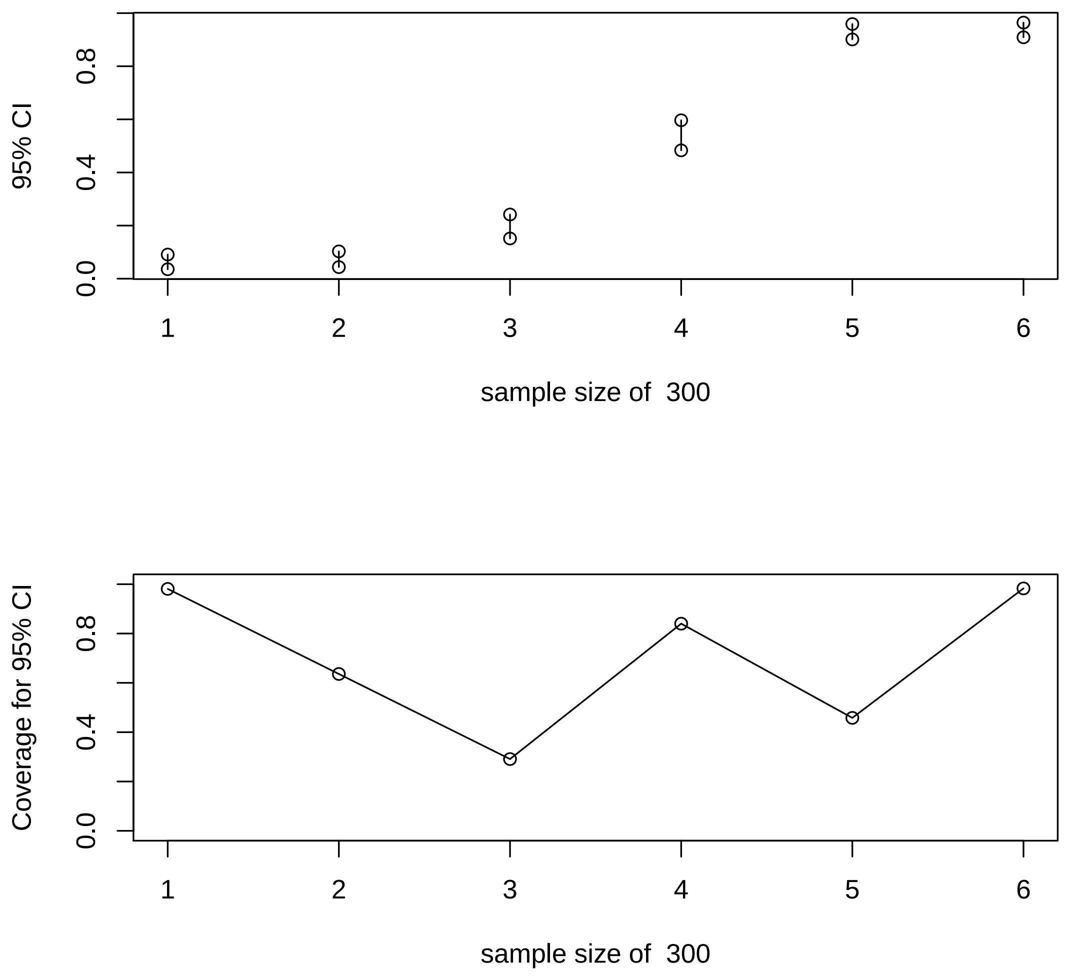


Figure 7: Confidence Intervals for Same $n = 500$ and Different p

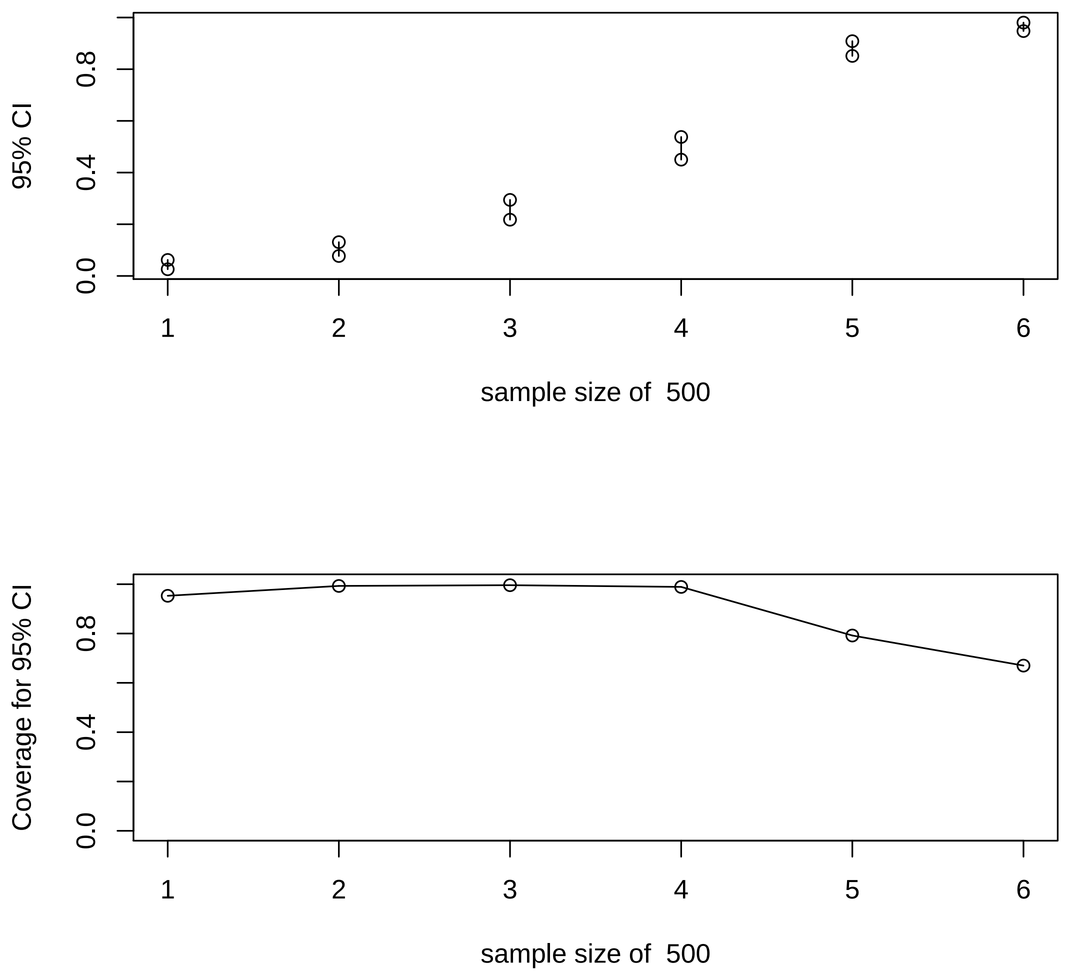
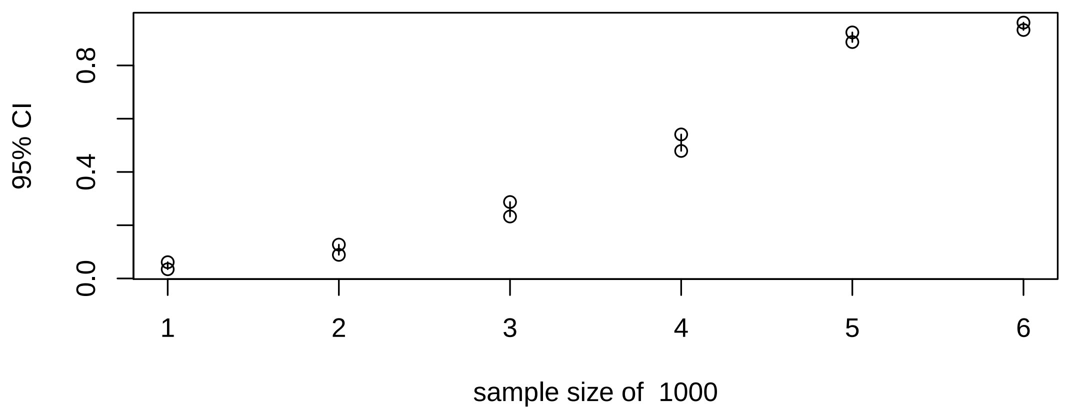
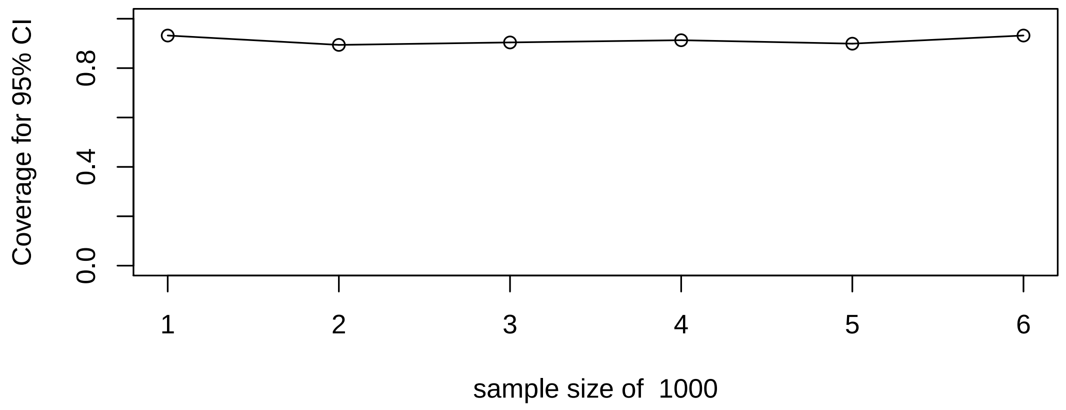


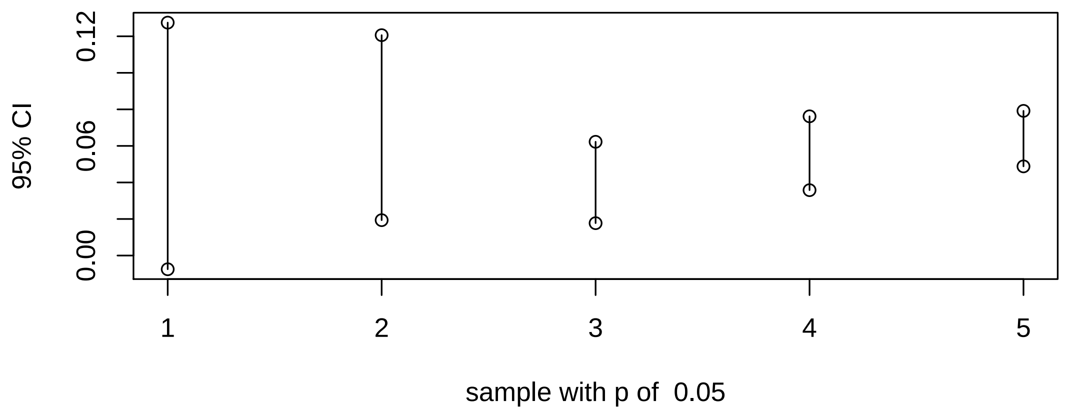
Figure 8: Confidence Intervals for Same $n = 1000$ and Different p





CI for the same p with different sample size n is shown in fig. 9

Figure 9: Confidence Intervals for $p = 0.05$ and Different n



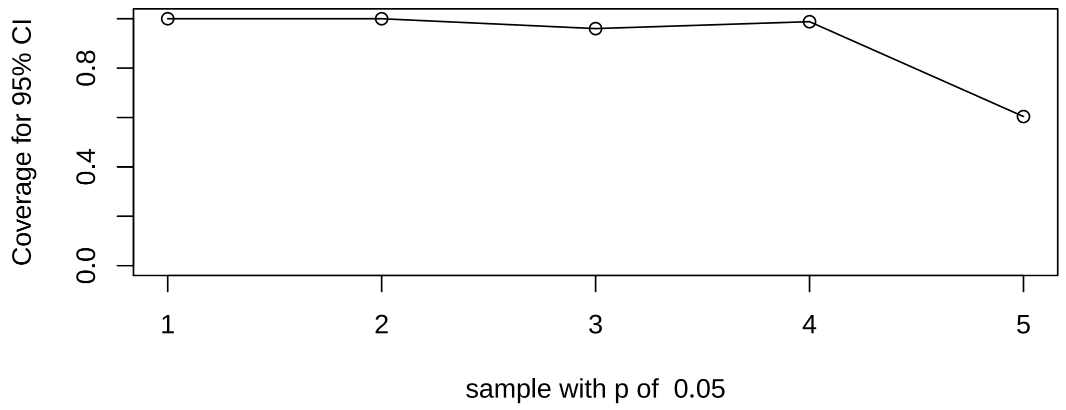


Figure 10: Confidence Intervals for $p = 0.1$ and Different n

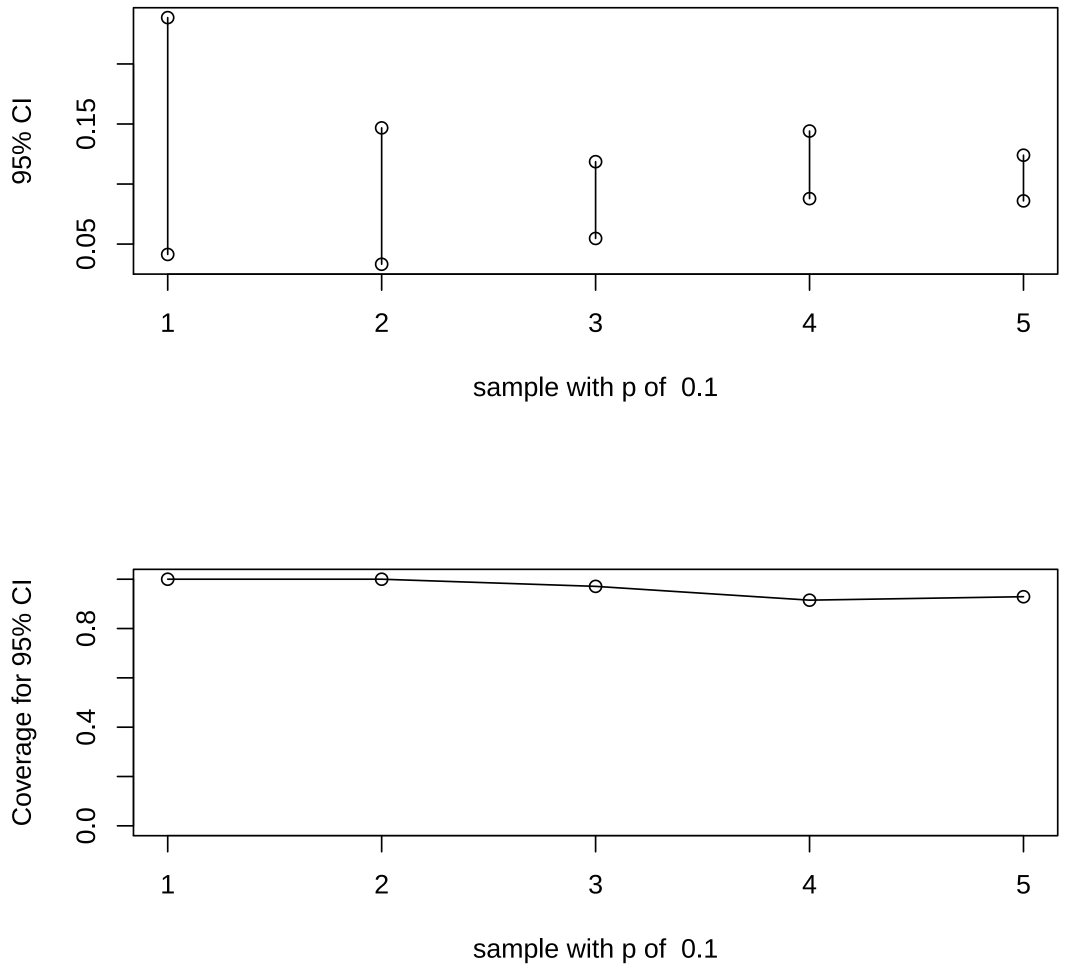


Figure 11: Confidence Intervals for $p = 0.25$ and Different n

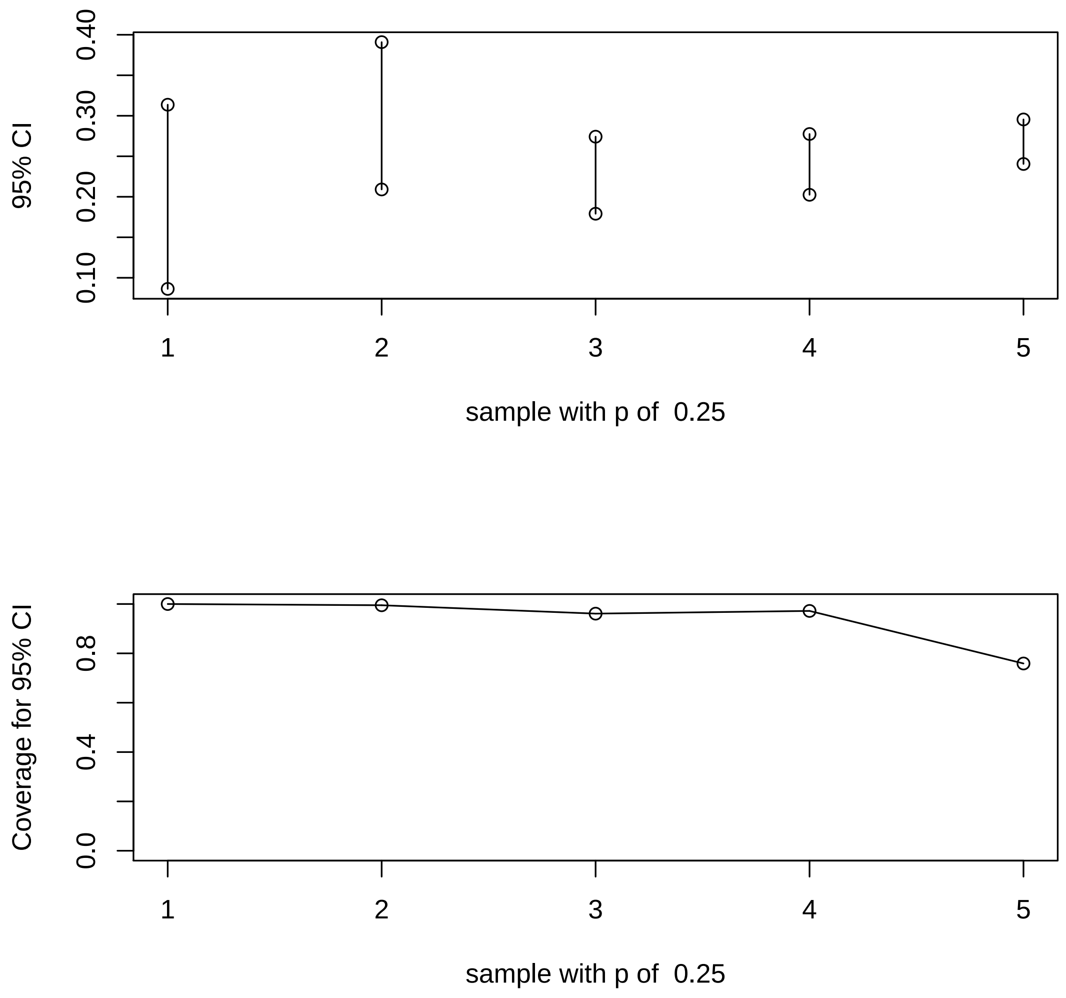
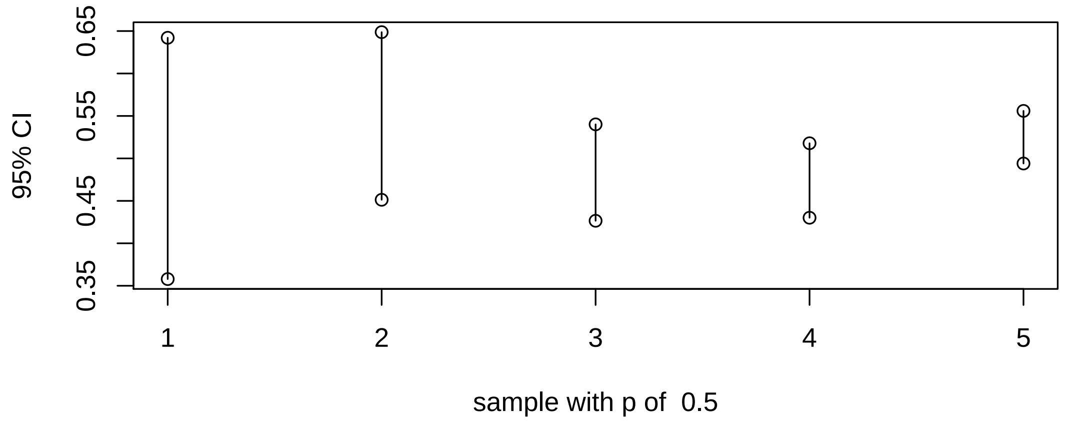


Figure 12: Confidence Intervals for $p = 0.5$ and Different n



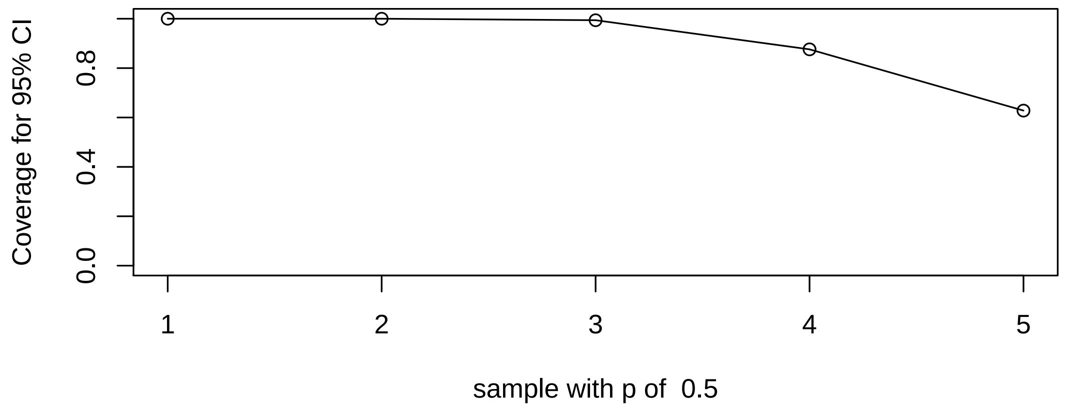
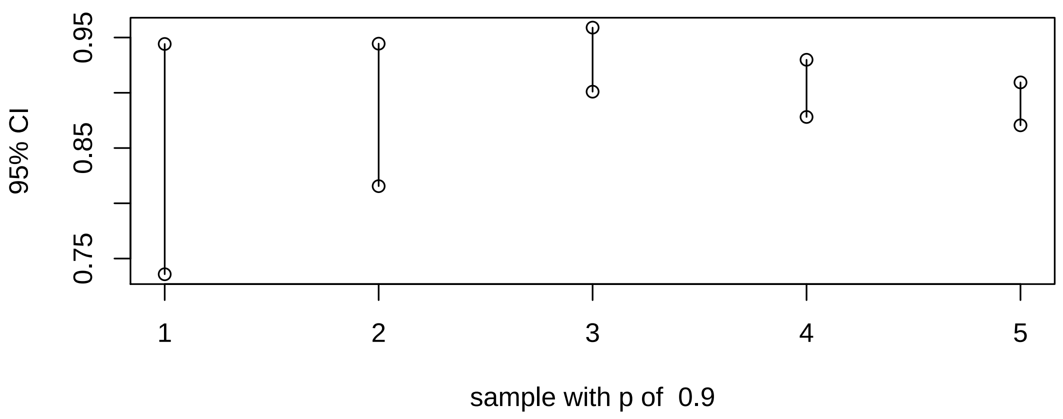


Figure 13: Confidence Intervals for $p = 0.9$ and Different n



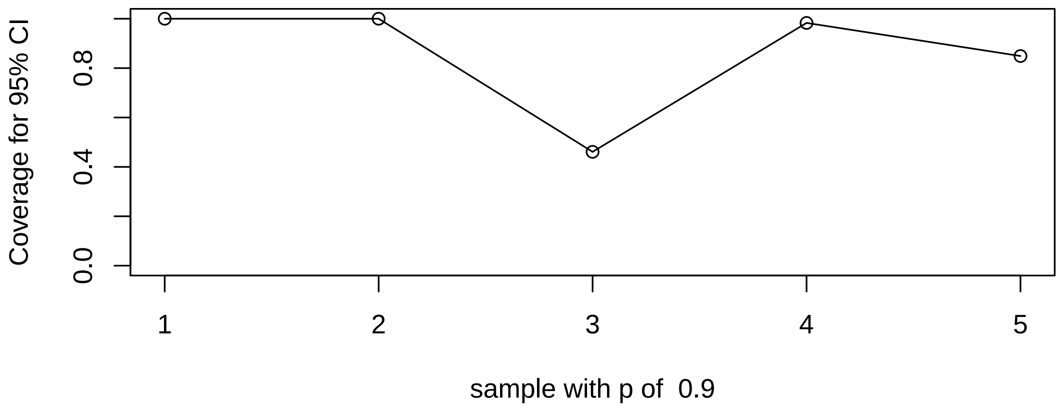
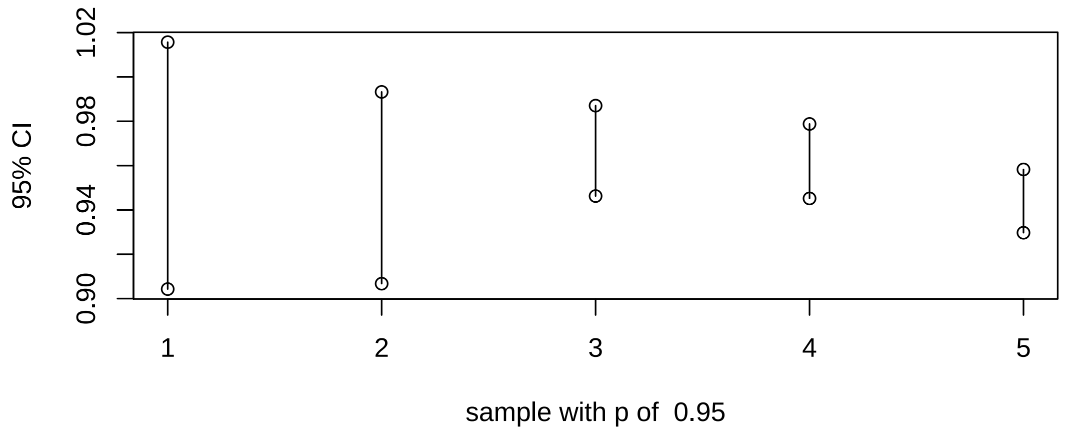
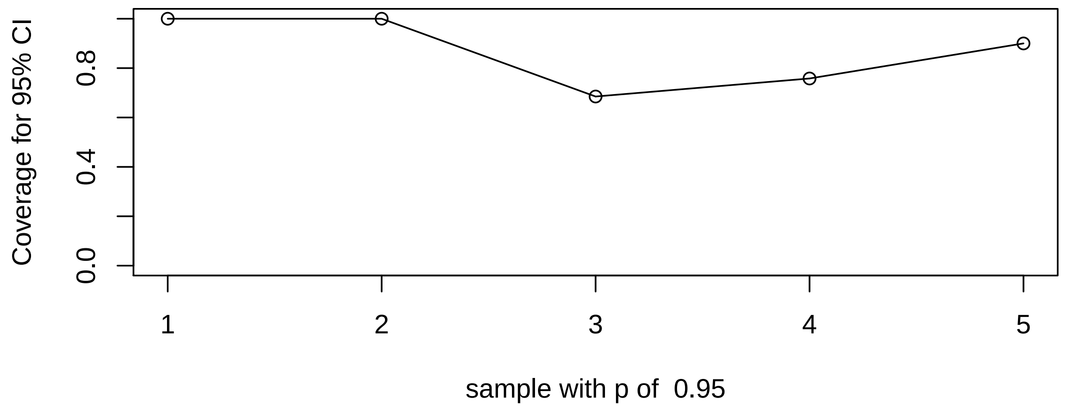


Figure 14: Confidence Intervals for $p = 0.95$ and Different n





Based on the output tables and confidence interval, we know that with the increase of n, the accuracy of confidence interval for a population proportion will be higher for same p. On the other hand, for same n, when $p = 0.5$, the CI is largest in general.

For this problem, $n = 100$ is recommended, and the answer does not rely on p.

Section 2 R Code

```r
###########################################################################
# R code for exercise 1
###########################################################################

# Read bp.txt file.
mydata <- read.table("bp.txt",
    header = TRUE,
    col.names=c("armsys", "fingsys"))

# Use 1.5 IQR rule to draw boxplots.
pdf("pic1.pdf")
boxplot(mydata, range = 1.5)
dev.off()

## pdf
##   2

# Draw a histogram for arm method.
minArm <- min(mydata[, 1])
maxArm <- max(mydata[, 1])
pdf("pic2.pdf")
hist(mydata[, 1],
    xlab = "times",
    ylab = "frequency",
    xlim = range(minArm, maxArm),
    main = "frequency of arm method data")
dev.off()

## pdf
##   2

# Draw a histogram for finger method.
minFinger <- min(mydata[, 2])
maxFinger <- max(mydata[, 2])
pdf("pic3.pdf")
hist(mydata[, 2],
    xlab = "times",
    ylab = "frequency",
    xlim = range(minFinger, maxFinger),
    main = "frequency of finger method data")
dev.off()

## pdf
##   2

# Draw a qqplot for arm method.
pdf("pic4.pdf")
qqnorm(mydata[, 1], main = "arm method")
qqline(mydata[, 1], main = "arm method")
dev.off()

## pdf
##   2

# Draw a qqlpot for finger method.
pdf("pic5.pdf")
qqnorm(mydata[, 2], main = "finger method")
qqline(mydata[, 2], main = "finger method")
dev.off()

## pdf
##   2

# CI for difference.
meanArm <- mean(mydata[, 1])
nArm <- nrow(mydata)
sdArm <- sd(mydata[, 1])
varArm <- var(mydata[, 1])

meanFinger <- mean(mydata[, 2])
nFinger <- nrow(mydata)
sdFinger <- sd(mydata[, 2])
varFinger <- var(mydata[, 2])

se <-sqrt(varFinger/nFinger + varArm/nArm)

ci.diff <- (meanFinger - meanArm) +
    c(-1,1) * qt(1 - (1 - 0.95)/2, nFinger + nArm - 2) * se
ci.diff

## [1] -0.5208386  9.1108386
```

```
############################################################################
# R code for exercise 2
############################################################################

library(functional)

pl <- cbind(c(0.05),c(0.1),c(0.25),c(0.5),c(0.9),c(0.95))
nl <- cbind(c(50),c(100),c(300),c(500),c(1000))

# Create a function to compute confidence intervals for different n and p.
# Arguments:
#       'n' is the specific number of draws in the simulation.
#       'p' is the specific probabilities for the simulation.
# Result:
#       will generate the CI for specifies p and n.
getCI <- function(n, p) {
    draw <- runif(n)
    prob <- sum(draw <= p)/sum(draw >= 0)
    ci <- prob + c(-1, 1) * qt(1-(1-0.95)/2, n-1) * sqrt(prob * (1-prob)/n)
    return(c(ci, getProb(p, ci)))
}

getProb <- function(p, ci) {
    draw <- replicate(1000, MonteCarlo(p, ci))
    return(sum(draw==TRUE)/1000)
}

MonteCarlo <- function(p, ci) {
    draw <- runif(1000)
    prob <- sum(draw <= p)/sum(draw >= 0)
    if(ci[1] <= prob && ci[2] >= prob) {
        return(TRUE)
    } else {
        return(FALSE)
    }
}

# Create a function to draw the plot for a particular n and a list of p,
#       using Curry function to specify parameter n for getCI(n,p)
# Arguments:
```

```
#       'n' is the specific number of draws in the simulation.
#       'pl' is the list of probabilities for the simulation.
# Result:
#       will generate the plot for all p in 'pl' with the same n.
#       as well as the coverage of the CI.
getCIwithAllP <- function(n, pl) {
    plotting <- function(ci, n) {
        plot(1:6,
            ci[1, 1:6],
            ylim = c(min(ci[1:2, 1:6]), max(ci[1:2, 1:6])),
            xlab = paste("sample size of ",n),
            ylab = "95% CI",
            type = "p")
        points(1:6, ci[2, 1:6])
        for(i in 1:6) {
            segments(i, ci[1, i], i, ci[2, i], lty = 1)
        }
    }
    plotting_prob <- function(ci, n) {
        plot(1:6,
            ci[3, 1:6],
            ylim = c(0,1),
            xlab = paste("sample size of ",n),
            ylab = "Coverage for 95% CI",
            type = "p")
        for(i in 1:5) {
            segments(i, ci[3, i], i+1, ci[3, i+1], lty = 1)
        }
    }
    getCIwithAllP_  <- Curry(getCI, n = n)
    ci <- apply(pl, 2, getCIwithAllP_)
    pdf(paste("sameN", n, ".pdf"))
    layout(matrix(c(1,2), 2, 1, byrow = TRUE))
    plotting(ci, n)
    plotting_prob(ci, n)
    dev.off()
}

# Curry the getCIwithAllP(n, pl) function by specifying the parameter pl.
# Function getCIforN now has one parameter: n.
```

```r
getCIforN <- Curry(getCIwithAllP, pl = pl)

# Apply getCIforN(n) to the array of n, for each n, draw the plot for all p.
apply(nl,2, getCIforN)
```

```
## [1] 2 2 2 2 2
```

```r
# Create a function to draw the plot for a particular p and a list of n,
#       using Curry function to specify parameter p for getCI(n,p)
# Arguments:
#       'nl' is the list of draws number in the simulation.
#       'p' is the specific probabilities for the simulation.
# Result:
#       will generate the plot for all n in 'nl' with the same p.
#       as well as the coverage of the CI.
getCIwithAllN <- function(nl, p) {
    plotting <- function(ci, p) {
        plot(1:5,
            ci[1, 1:5],
            ylim = c(min(ci[1:2, 1:5]), max(ci[1:2, 1:5])),
            xlab = paste("sample with p of ", p),
            ylab = "95% CI",
            type ="p")
        points(1:5, ci[2, 1:5])
        for(i in 1:5) {
            segments(i, ci[1, i], i, ci[2, i], lty = 1)
        }
    }
    plotting_prob <- function(ci, p) {
        plot(1:5,
            ci[3, 1:5],
            ylim = c(0,1),
            xlab = paste("sample with p of ", p),
            ylab = "Coverage for 95% CI",
            type ="p")
        for(i in 1:4) {
            segments(i, ci[3, i], i+1, ci[3, i+1], lty = 1)
        }
    }

    getCIwithAllN_  <- Curry(getCI, p = p)
    ci <- apply(nl, 2, getCIwithAllN_)
    pdf(paste("sameP", p, ".pdf"))
    layout(matrix(c(1,2), 2, 1, byrow = TRUE))
    plotting(ci, p)
    plotting_prob(ci, p)
    dev.off()

    plotting(apply(nl, 2, getCIwithAllN_), p)
}

# Curry the getCIwithAllN(nl, p) function by specifying the parameter nl.
# Function getCIforP now has one parameter: p.
getCIforP <- Curry(getCIwithAllN, nl = nl)

# Apply getCIforP(p) to the array of p, for each p, draw the plot for all n.
apply(pl,2, getCIforP)
```