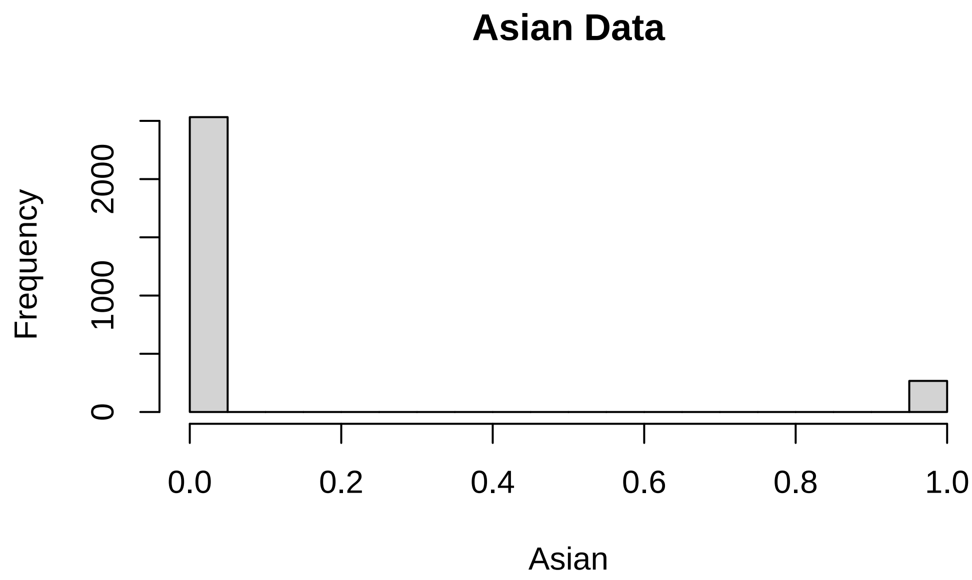
HW04_OtherProb01

```r
############################################
##HW 04 Other problems 01 code
############################################
#install.packages("tidyverse")
#install.packages("plotrix")
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
library(plotrix)
#read the data
CHIS_data <- read.csv("https://www4.stat.ncsu.edu/online/datasets/CHIS.csv")
#CHIS_data <- read_csv("data/CHIS.csv") %>% select(-1)


hist(CHIS_data$Asian, main = "Asian Data", xlab = "Asian", breaks = 20)
```




```r
mu <- mean(CHIS_data$Asian)
mu
```

```
[1] 0.09539121
```

```r
set.seed(3)


n <- 25
sample_data <- sample(CHIS_data$Asian, size = n, replace = FALSE)
sample_data
```

```
 [1] 0 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0
```

```r
mean(sample_data)
```

```
[1] 0.16
```

```r
sd(sample_data)/sqrt(n)
```

```
[1] 0.07483315
```

```r
c(mean(sample_data)-qnorm(0.975)*sd(sample_data)/sqrt(n),
  mean(sample_data)+qnorm(0.975)*sd(sample_data)/sqrt(n))
```

```
[1] 0.01332973 0.30667027
```

```r
#####################################################
#####################################################
n <- 8
N <- 5000
observed_CIs <- replicate(N, {
  sample_data <- sample(CHIS_data$Asian, size = n, replace = FALSE)
  lower <- mean(sample_data) - qnorm(0.975) * sd(sample_data) / sqrt(n)
  upper <- mean(sample_data) + qnorm(0.975) * sd(sample_data) / sqrt(n)
  c(lower, upper)
})
observed_CIs[, 1:5]
```

```
           [,1] [,2] [,3] [,4] [,5]
[1,] -0.1199955    0    0    0    0
[2,]  0.3699955    0    0    0    0
```

```r
#check how many contained the truth value
# Calculate the fraction of intervals containing the true population mean (mu)
sample_means <- mean((observed_CIs[1, ] < mu) & (observed_CIs[2, ] > mu))
sample_means
```

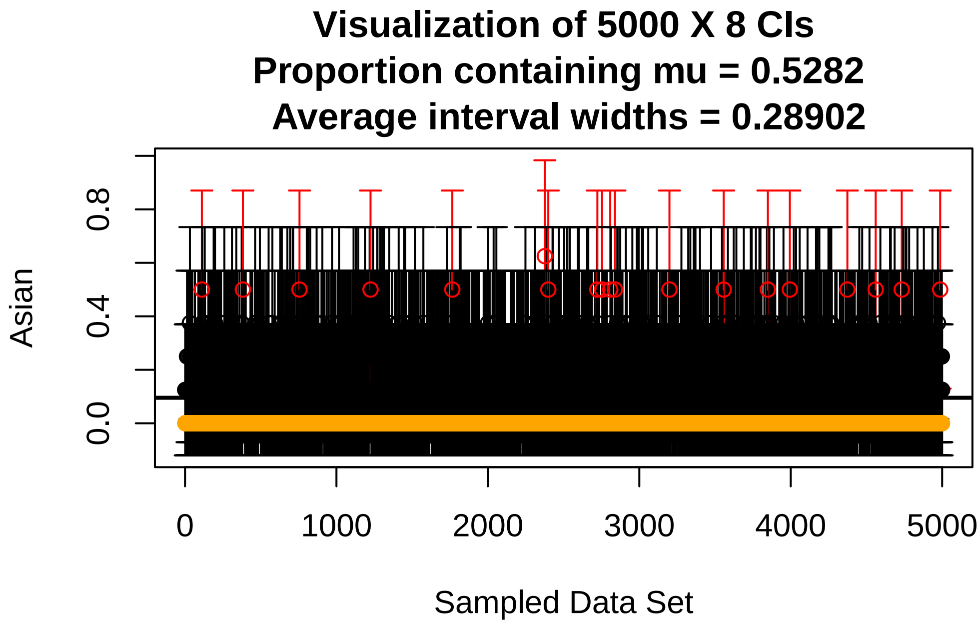
```
[1] 0.5282
```

```r
# Calculate the average width of the intervals
interval_widths <- observed_CIs[2, ] - observed_CIs[1, ]
average_width <- mean(interval_widths)
average_width
```

```
[1] 0.2890238
```

```
#quick function to color our intervals based on how they hit or miss
mycolor <- function(endpoints, par) {
  if (par < endpoints[1])
    "Red"  # if the mean is below the left endpoint of the confidence interval
  else if (par > endpoints[2])
    "Orange"  # if the mean is above the right endpoint of the confidence interval
  else "Black"  # if the mean lies between the endpoints
}

#Load the plotrix package, which contains the plotCI function.
require(plotrix)
plotCI(x = 1:N,
       y = colMeans(observed_CIs),
       li = observed_CIs[1, ],
       ui = observed_CIs[2, ],
       col = apply(FUN = mycolor, X = observed_CIs, MARGIN = 2, par = mu),
       ylab = "Asian",
       xlab = "Sampled Data Set",
       main = paste0("Visualization of 5000 X 8 CIs\nProportion containing mu = ", sample_mea
                     " \n Average interval widths = ", round(average_width, digits = 5))
)
#draw a line for true mean
abline(h = mu, lwd = 2)
```




```
#######################################################
#> n <- 8
#> N <- 5000

#> observed_CIs[, 1:5]
#[,1] [,2] [,3] [,4] [,5]
#[1,] -0.1199955    0    0    0    0
#[2,]  0.3699955    0    0    0    0

#> sample_means
#[1] 0.5282

#> average_width
#[1] 0.2890238

#######################################################

n <- 50
N <- 5000
observed_CIs <- replicate(N, {
  sample_data <- sample(CHIS_data$Asian, size = n, replace = FALSE)
  lower <- mean(sample_data) - qnorm(0.975) * sd(sample_data) / sqrt(n)
  upper <- mean(sample_data) + qnorm(0.975) * sd(sample_data) / sqrt(n)
```

```r
  c(lower, upper)
})
observed_CIs[, 1:5]
```

```
            [,1]      [,2]       [,3]         [,4]       [,5]
[1,] -0.01486756 0.0040393 0.01600154 -0.006495094 0.04284542
[2,]  0.09486756 0.1559607 0.18399846  0.126495094 0.23715458
```

```r
#check how many contained the truth value
# Calculate the fraction of intervals containing the true population mean (mu)
sample_means <- mean((observed_CIs[1, ] < mu) & (observed_CIs[2, ] > mu))
sample_means
```

```
[1] 0.8626
```

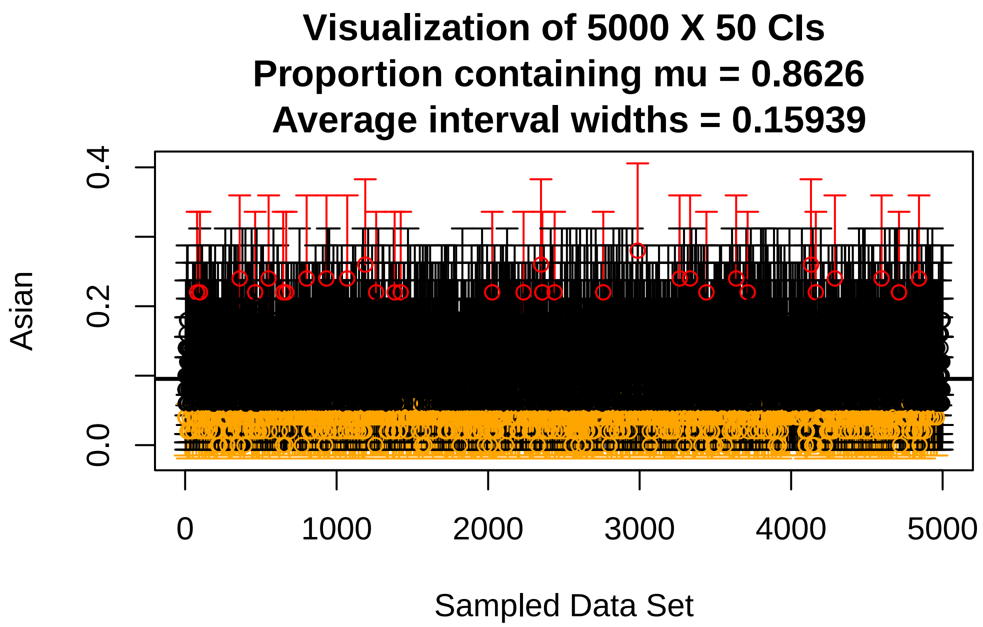
```r
# Calculate the average width of the intervals
interval_widths <- observed_CIs[2, ] - observed_CIs[1, ]
average_width <- mean(interval_widths)
average_width
```

```
[1] 0.1593889
```

```r
#quick function to color our intervals based on how they hit or miss
mycolor <- function(endpoints, par) {
  if (par < endpoints[1])
    "Red"  # if the mean is below the left endpoint of the confidence interval
  else if (par > endpoints[2])
    "Orange"  # if the mean is above the right endpoint of the confidence interval
  else "Black"  # if the mean lies between the endpoints
}

#Load the plotrix package, which contains the plotCI function.
require(plotrix)
plotCI(x = 1:N,
       y = colMeans(observed_CIs),
       li = observed_CIs[1, ],
       ui = observed_CIs[2, ],
       col = apply(FUN = mycolor, X = observed_CIs, MARGIN = 2, par = mu),
       ylab = "Asian",
       xlab = "Sampled Data Set",
```

```r
      main = paste0("Visualization of 5000 X 50 CIs\nProportion containing mu = ", sample_me
                    " \n Average interval widths = ", round(average_width, digits = 5))
)
#draw a line for true mean
abline(h = mu, lwd = 2)
```




```
######################################################
#> n <- 50
#> N <- 5000

#> observed_CIs[, 1:5]
#[,1]       [,2]       [,3]         [,4]        [,5]
#[1,] -0.01486756 0.0040393 0.01600154 -0.006495094 0.04284542
#[2,]  0.09486756 0.1559607 0.18399846  0.126495094 0.23715458

#> sample_means
#[1] 0.8626

#> average_width
#[1] 0.1593889
######################################################
```

```r
n <- 100
N <- 5000
observed_CIs <- replicate(N, {
  sample_data <- sample(CHIS_data$Asian, size = n, replace = FALSE)
  lower <- mean(sample_data) - qnorm(0.975) * sd(sample_data) / sqrt(n)
  upper <- mean(sample_data) + qnorm(0.975) * sd(sample_data) / sqrt(n)
  c(lower, upper)
})
observed_CIs[, 1:5]
```

```
            [,1]       [,2]       [,3]       [,4]       [,5]
[1,] 0.001399218 0.03362683 0.05598784 0.02655964 0.04090486
[2,] 0.078600782 0.14637317 0.18401216 0.13344036 0.15909514
```

```r
#check how many contained the truth value
# Calculate the fraction of intervals containing the true population mean (mu)
sample_means <- mean((observed_CIs[1, ] < mu) & (observed_CIs[2, ] > mu))
sample_means
```

```
[1] 0.91
```

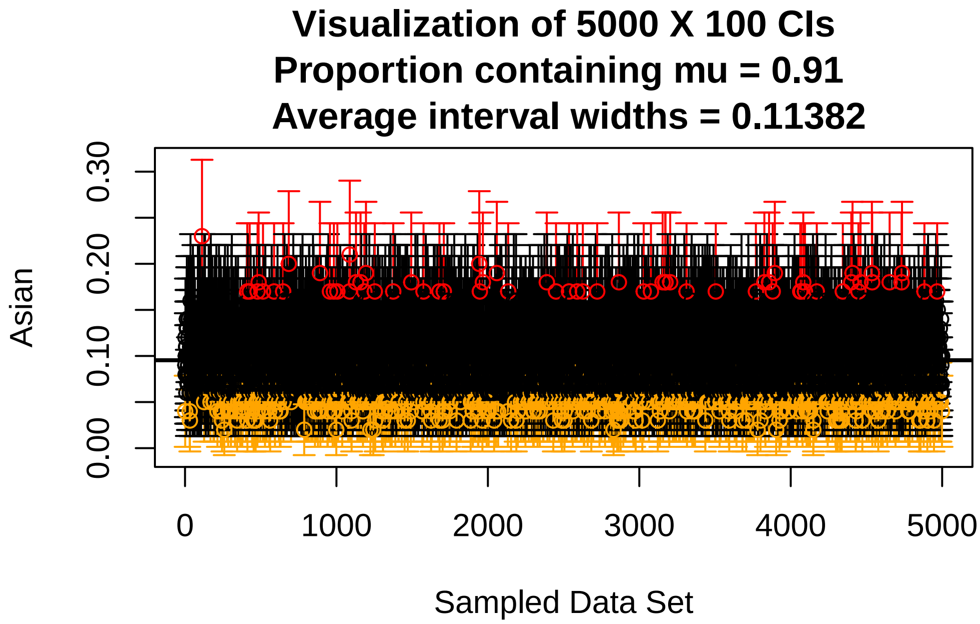
```r
# Calculate the average width of the intervals
interval_widths <- observed_CIs[2, ] - observed_CIs[1, ]
average_width <- mean(interval_widths)
average_width
```

```
[1] 0.1138199
```

```r
#quick function to color our intervals based on how they hit or miss
mycolor <- function(endpoints, par) {
  if (par < endpoints[1])
    "Red"  # if the mean is below the left endpoint of the confidence interval
  else if (par > endpoints[2])
    "Orange"  # if the mean is above the right endpoint of the confidence interval
  else "Black"  # if the mean lies between the endpoints
}

#Load the plotrix package, which contains the plotCI function.
require(plotrix)
plotCI(x = 1:N,
```

```r
        y = colMeans(observed_CIs),
        li = observed_CIs[1, ],
        ui = observed_CIs[2, ],
        col = apply(FUN = mycolor, X = observed_CIs, MARGIN = 2, par = mu),
        ylab = "Asian",
        xlab = "Sampled Data Set",
        main = paste0("Visualization of 5000 X 100 CIs\nProportion containing mu = ", sample_m
                      " \n Average interval widths = ", round(average_width, digits = 5))
)
#draw a line for true mean
abline(h = mu, lwd = 2)
```




```
#####################################################
#> n <- 100
#> N <- 5000

#> observed_CIs[, 1:5]
#[,1]        [,2]       [,3]       [,4]       [,5]
#[1,] 0.001399218 0.03362683 0.05598784 0.02655964 0.04090486
#[2,] 0.078600782 0.14637317 0.18401216 0.13344036 0.15909514

#> sample_means
```

```
#[1] 0.91

#> average_width
#[1] 0.1138199
#####################################################

n <- 1000
N <- 5000
observed_CIs <- replicate(N, {
  sample_data <- sample(CHIS_data$Asian, size = n, replace = FALSE)
  lower <- mean(sample_data) - qnorm(0.975) * sd(sample_data) / sqrt(n)
  upper <- mean(sample_data) + qnorm(0.975) * sd(sample_data) / sqrt(n)
  c(lower, upper)
})
observed_CIs[, 1:5]
```

```
           [,1]       [,2]      [,3]       [,4]       [,5]
[1,] 0.07134289 0.07681759 0.0677064 0.06679904 0.06952326
[2,] 0.10665711 0.11318241 0.1022936 0.10120096 0.10447674
```

```
#check how many contained the truth value
# Calculate the fraction of intervals containing the true population mean (mu)
sample_means <- mean((observed_CIs[1, ] < mu) & (observed_CIs[2, ] > mu))
sample_means
```

```
[1] 0.9836
```

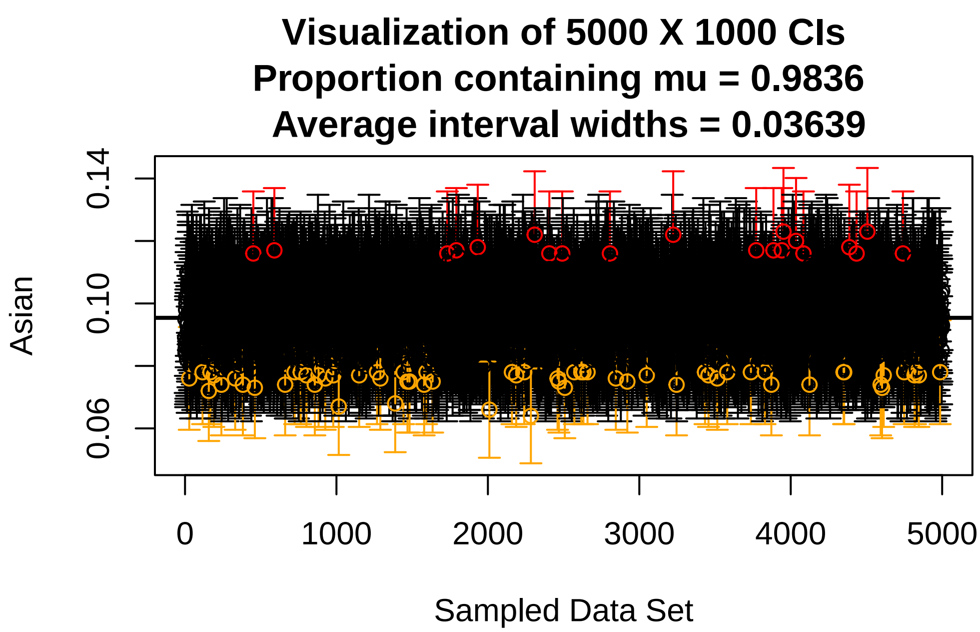
```
# Calculate the average width of the intervals
interval_widths <- observed_CIs[2, ] - observed_CIs[1, ]
average_width <- mean(interval_widths)
average_width
```

```
[1] 0.0363871
```

```
#quick function to color our intervals based on how they hit or miss
mycolor <- function(endpoints, par) {
  if (par < endpoints[1])
    "Red"  # if the mean is below the left endpoint of the confidence interval
  else if (par > endpoints[2])
    "Orange"  # if the mean is above the right endpoint of the confidence interval
```

```r
  else "Black"  # if the mean lies between the endpoints
}

#Load the plotrix package, which contains the plotCI function.
require(plotrix)
plotCI(x = 1:N,
       y = colMeans(observed_CIs),
       li = observed_CIs[1, ],
       ui = observed_CIs[2, ],
       col = apply(FUN = mycolor, X = observed_CIs, MARGIN = 2, par = mu),
       ylab = "Asian",
       xlab = "Sampled Data Set",
       main = paste0("Visualization of 5000 X 1000 CIs\nProportion containing mu = ", sample_
                     " \n Average interval widths = ", round(average_width, digits = 5))
)
#draw a line for true mean
abline(h = mu, lwd = 2)
```




```
#####################################################
#> n <- 1000
#> N <- 5000
```

```
#> observed_CIs[, 1:5]
#[,1]       [,2]      [,3]       [,4]       [,5]
#[1,] 0.07134289 0.07681759 0.0677064 0.06679904 0.06952326
#[2,] 0.10665711 0.11318241 0.1022936 0.10120096 0.10447674

#> sample_means
#[1] 0.9836

#> average_width
#[1] 0.0363871
#####################################################
#Sample Size = 8
#Sample Means: 0.5282, indicate that approximately 52.82% of the intervals contained the true
#              This is lower than the expected 95% confidence level.
#              That means small sample sizes gives less reliable intervals.
#Average Width: 0.2890, shows a large interval width due to the high variability in small sam

#Sample Size = 50
#Sample Means: 0.8626, means that approximately 86.26% of the intervals contained the true me
#              This is close to the expected 95% but still below.
#Average Width: 0.1594, indicate smaller intervals compared to n = 8. This means increased pr

#Sample Size = 100
#Sample Means: 0.91, shows that 91% of the confidence intervals captured the true mean, which
#Average Width: 0.1138, shows further narrowing of the confidence intervals, leading to more

#Sample Size = 1000
#Sample Means: 0.9836, shows that 98.36% of the intervals contained the true mean, which exce
#Average Width: 0.0364, shows very precise estimates of the true mean.

#Conclusion: Increasing the sample size results in better coverage of the true population mea
#            Larger sample sizes reduce variability, improve accuracy, and gives a higher pro
#            the intervals containing the true mean.

#####################################################
```