Project 2
STAT 355

Sabbir Ahmed

April 10, 2017

1 Part 1

1000 random samples of size 40 were generated from normal distribution with mean $\mu = 3$ and standard deviation $\sigma = 2$.

```
# initialize parameters for normal distribution
N <- 40 # size
mu <- 3 # mean
sigma <- 2 # standard deviation
sampMeans <- rep(0, times=NUMSAMPS) # initialize empty array
# generate 1000 samples
for (i in 1:NUMSAMPS){
    generatedData <- rnorm(N, mu, sigma)
    # store the sample means in vector
    sampMeans[i] = mean(generatedData)

    if (i == 1) {
        firstMean = mean(generatedData)
        firstStd = sd(generatedData)
    }

}
```

1.1 Output

The first sample mean and standard deviation were computed:

$$E(\overline{X}) = 2.833, \ \sigma_{\overline{X}} = 0.316$$

All the samples were then used to find the sample mean and standard deviation. The theoretical values were also computed based on the relationships:

$$\mu = \mu$$
$$E(\overline{X}) = \mu$$
$$\sigma = \sigma$$
$$\sigma_{\overline{X}} = \frac{\sigma}{\sqrt{n}}$$

	Actual	Theoretical
μ	3.000	3.000
$E(\overline{X})$	2.990	3.000
σ	2.000	2.000
$\sigma_{\overline{X}}$	0.311	0.316

1.2 Distribution

Distribution of the data was plotted with a histogram using ggplot2 in Figure 1.

```
ggplot() + aes(sampMeans) +
    geom_histogram(binwidth=0.1, color="black", fill="white") +
    labs(y="Count", x="Sample Means")
```

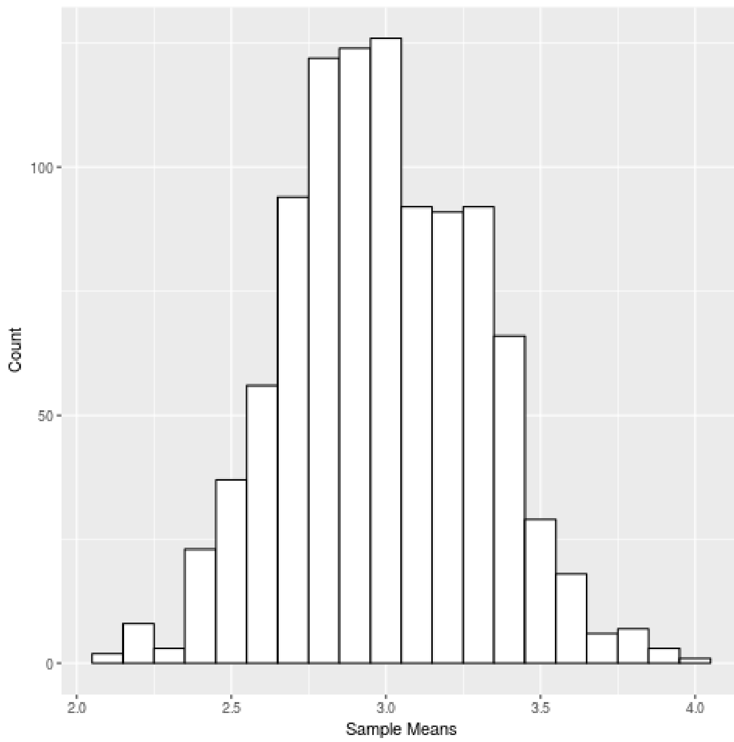

Figure 1: Histogram of the Generated Data

2 Part 2

1000 random samples of size 15 were generated from a binomial distribution with n = 10 and standard deviation p = 0.15.

```
# initialize parameters for binomial distribution
N <- 15
n <- 10
p <- 0.15
sampMeans <- rep(0, times=NUMSAMPS) # initialize empty array
for (i in 1:NUMSAMPS){
```

```
        generatedData <- rbinom(N, n, p)
        sampMeans[i] = mean(generatedData)

        if (i == 1) {
            firstMean = mean(generatedData)
            firstStd = sd(generatedData)
        }

    }
```

2.1 Output

The first sample mean and standard deviation were computed:

$$E(\overline{X}) = 1.000,\ \ \sigma_{\overline{X}} = 0.926$$

All the samples were then used to find the sample mean and standard deviation. The theoretical values were also computed based on the relationships:

$$\mu = np$$
$$E(\overline{X}) = np$$
$$\sigma = np(1-p)$$
$$\sigma_{\overline{X}} = \frac{np(1-p)}{\sqrt{n}}$$

	Actual	**Theoretical**
μ	1.500	1.500
E($\overline{X}$)	1.518	1.500
σ	1.275	1.275
$\sigma_{\overline{X}}$	0.292	0.329

2.2 Distribution

Distribution of the data was plotted with a histogram using ggplot2 in Figure 2.

```
    # plot a histogram of the data
    ggplot() + aes(sampMeans) +
        geom_histogram(binwidth=0.2, color="black", fill="white") +
        labs(y="Count", x="Sample Means")
```

3 Part 3

1000 random samples of size 120 were generated from a binomial distribution with n = 10 and standard deviation p = 0.15.

```
    # initialize parameters for binomial distribution
    N <- 120
    n <- 10
    p <- 0.15
    sampMeans <- rep(0, times=NUMSAMPS) # initialize empty array
    for (i in 1:NUMSAMPS){
```

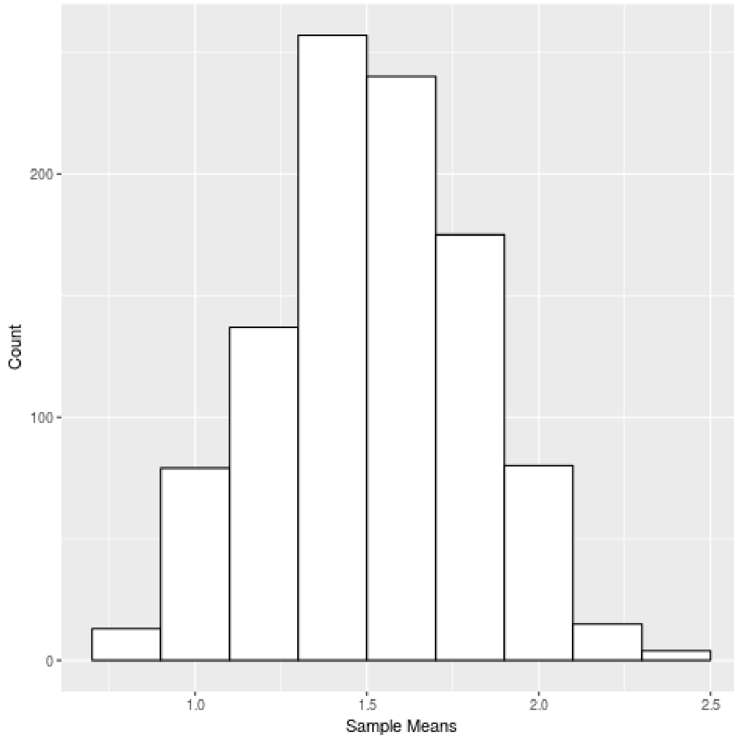

Figure 2: Histogram of the Generated Data

```
    generatedData <- rbinom(N, n, p)
    sampMeans[i] = mean(generatedData)

    if (i == 1) {
        firstMean = mean(generatedData)
        firstStd = sd(generatedData)
    }

}
```

3.1 Output

The first sample mean and standard deviation were computed:

$$E(\overline{X}) = 1.492, \ \sigma_{\overline{X}} = 0.011$$

All the samples were then used to find the sample mean and standard deviation. The theoretical values were also computed based on the relationships:

$$\mu = np$$
$$E(\overline{X}) = np$$
$$\sigma = np(1-p)$$
$$\sigma_{\overline{X}} = \frac{np(1-p)}{\sqrt{n}}$$

	Actual	Theoretical
μ	1.500	1.500
$\mathrm{E}(\overline{X})$	1.502	1.500
σ	1.275	1.275
$\sigma_{\overline{X}}$	0.101	0.116

3.2 Distribution

Distribution of the data was plotted with a histogram using ggplot2 in Figure 3.

```
# plot a histogram of the data
ggplot() + aes(sampMeans) +
    geom_histogram(binwidth=0.1, color="black", fill="white") +
    labs(y="Count", x="Sample Means")
```

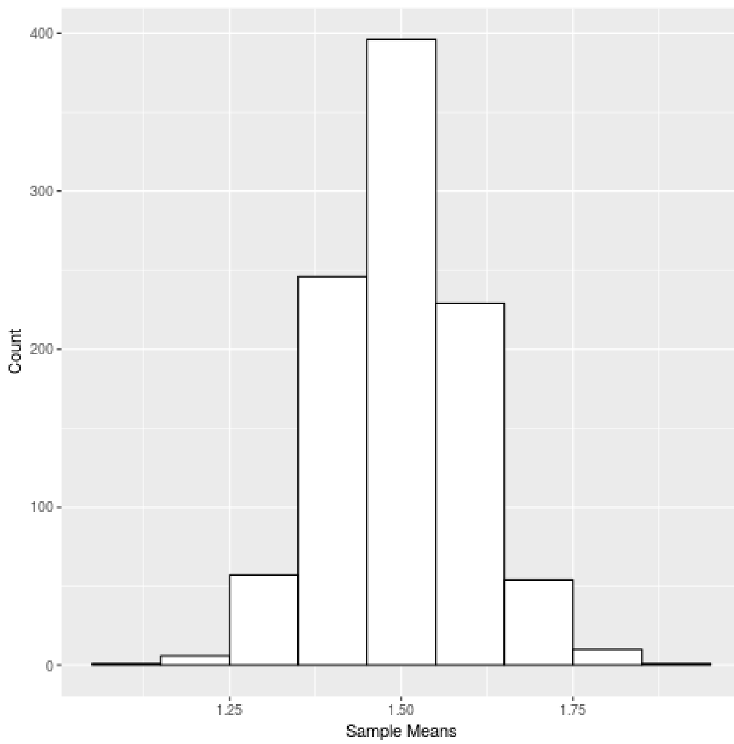

Figure 3: Histogram of the Generated Data

4 Conclusion

main.R

```r
# main.R
# This file contains the implementation of the functions in the Project 2
# NOTE: THIS SCRIPT WAS COMPILED ON A LINUX MACHINE - SOME STATEMENTS MAY THROW
# WARNINGS OR ERRORS IN OTHER SYSTEMS

library(ggplot2)  # for generating high quality plots
set.seed(124)  # seed the random generators

# LaTex template for the output
outputTemplate <- "\\subsection{Output}

    The first sample mean and standard deviation were computed:

    \\[ E(\\overline{X}) = %.3f, \\ \\sigma_{\\overline{X}} = %.3f \\]

    All the samples were then used to find the sample mean and standard
    deviation. The theoretical values were also computed based on the
    relationships:

    \\[ \\mu = %s \\]
    \\[ E(\\overline{X}) = %s \\]
    \\[ \\sigma = %s \\]
    \\[ \\sigma_{\\overline{X}} = %s \\]


    \\begin{table}[h]
        \\centering
        \\begin{tabular*}{200pt}{@{\\extracolsep{\\fill}} c c c}

        & \\textbf{Actual} & \\textbf{Theoretical} \\\\
        \\hline
        $\\mu$ & %.3f  & %.3f \\\\
        E($\\overline{X}$) & %.3f & %.3f \\\\
        $\\sigma$ & %.3f & %.3f \\\\
        $\\sigma$\\textsubscript{$\\overline{X}$} & %.3f & %.3f \\\\

        \\end{tabular*}
    \\end{table}
"

# global variables
NUMSAMPS <- 1000
firstMean <- 0
firstStd <- 0

# ------------------------------ Part 1 ------------------------------

# initialize parameters for normal distribution
N <- 40  # size
mu <- 3  # mean
sigma <- 2  # standard deviation

sampMeans <- rep(0, times=NUMSAMPS)  # initialize empty array
firstMean <- 0
firstStd <- 0

# generate 1000 samples
for (i in 1:NUMSAMPS){
    generatedData <- rnorm(N, mu, sigma)
    # store the sample means in vector
    sampMeans[i] = mean(generatedData)

    if (i == 1) {
        firstMean = mean(generatedData)
        firstStd = sigma/sqrt(N)
```

```
    }

}

# save output
sink("part1.tex", append=FALSE, split=FALSE)
cat(
    sprintf(
        outputTemplate,
        firstMean, firstStd,
        "\\mu", "\\mu", "\\sigma", "\\frac{\\sigma}{\\sqrt{n}}",
        mu, mu,
        mean(sampMeans), mu,
        sigma, sigma,
        sd(sampMeans), sigma/sqrt(N)
    )
)
sink()

png(filename="figures/hist1.png")
# plot a histogram of the data
ggplot() + aes(sampMeans) +
    geom_histogram(binwidth=0.1, color="black", fill="white") +
    labs(y="Count", x="Sample Means")

dev.off()


# ------------------------------ Part 2 ------------------------------

# initialize parameters for binomial distribution
N <- 15
n <- 10
p <- 0.15

sampMeans <- rep(0, times=NUMSAMPS)  # initialize empty array
for (i in 1:NUMSAMPS){
    generatedData <- rbinom(N, n, p)
    sampMeans[i] = mean(generatedData)

    if (i == 1) {
        firstMean = mean(generatedData)
        firstStd = sd(generatedData)
    }

}

# save output
sink("part2.tex", append=FALSE, split=FALSE)
cat(
    sprintf(
        outputTemplate,
        firstMean, firstStd,
        "np", "np", "np(1-p)", "\\frac{np(1-p)}{\\sqrt{n}}",
        n*p, n*p,
        mean(sampMeans), n*p,
        n*p*(1-p), n*p*(1-p),
        sd(sampMeans), n*p*(1-p)/sqrt(N)
    )
)
sink()

png(filename="figures/hist2.png")

# plot a histogram of the data
ggplot() + aes(sampMeans) +
    geom_histogram(binwidth=0.2, color="black", fill="white") +
    labs(y="Count", x="Sample Means")
```

```
dev.off()


# ------------------------------ Part 3 ------------------------------

# initialize parameters for binomial distribution
N <- 120
n <- 10
p <- 0.15

sampMeans <- rep(0, times=NUMSAMPS)  # initialize empty array
for (i in 1:NUMSAMPS){
    generatedData <- rbinom(N, n, p)
    sampMeans[i] = mean(generatedData)

    if (i == 1) {
        firstMean = mean(generatedData)
        firstStd = n*p*(1-p)/N
    }

}

# save output
sink("part3.tex", append=FALSE, split=FALSE)
cat(
    sprintf(
        outputTemplate,
        firstMean, firstStd,
        "np", "np", "np(1-p)", "\\frac{np(1-p)}{\\sqrt{n}}",
        n*p, n*p,
        mean(sampMeans), n*p,
        n*p*(1-p), n*p*(1-p),
        sd(sampMeans), n*p*(1-p)/sqrt(N)
    )
)
sink()

png(filename="figures/hist3.png")

# plot a histogram of the data
ggplot() + aes(sampMeans) +
    geom_histogram(binwidth=0.1, color="black", fill="white") +
    labs(y="Count", x="Sample Means")

dev.off()
```
