Project 3
STAT 355

Sabbir Ahmed

May 12, 2017

1 Part 1

1.1 Question

An oceanographer wants to test, on the basis of a random sample of size 35, whether the average depth of the ocean in a certain area is 72.4 fathoms. At the 0.05 level of significance, what will the oceanographer decide if she gets a sample mean of 73.2? Assume the population standard deviation is 2.1.

1.2 Answer

The null hypothesis, H_0, claims the mean depth of the ocean in a certain area is 72.4, while the alternative hypothesis, H_a, says otherwise.

$$H_0 : \mu = 72.4 \ vs \ H_a : \mu \neq 72.4$$

Since the population mean and standard deviation was known with a sample size of $n > 30$, the Z-score was calculated as follows:

$$Z = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} = \frac{73.2 - 72.4}{2.1/\sqrt{35}} = 2.2537$$

The following snippet was used to generate the Z-value and its probability:

```
X <- 73.2
mu <- 72.4
sigma <- 2.1
n <- 35

z <- (X - mu)/(sigma/sqrt(n))
print(z) # print the Z-score
print(pnorm(z)) # print the probability
```

The Z-score was computed to be:

$$\because Z_{0.025} = -1.9600 < 2.2537$$

2 Part 2

2.1 Question

A random sample of 12 graduates of a secretarial school averaged 73.2 words per minute with a standard deviation of 7.9 words per minute on a typing test. What can we conclude, at the 0.05 level, regarding the claim that secretaries at this school average less than 75 words per minute on the typing test?

2.2 Answer

The null hypothesis, H_0, claims the school averaged greater or equal to 75, while the alternative hypothesis, H_a, says otherwise.

$$H_0 : \mu \geq 75 \ vs \ H_a : \mu < 75$$

Since the population standard deviation is unknown, and the sample was $n < 30$, the t-score was calculated as follows:

$$t = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} = \frac{73.2 - 75.0}{7.9/\sqrt{12}} = -0.7893$$

The following snippet was used to generate the Z-value and its probability:

```r
X <- 73.2
mu <- 72.4
sigma <- 2.1
n <- 35

z <- (X - mu)/(sigma/sqrt(n))
print(z) # print the Z-score
print(pnorm(z)) # print the probability
```

The Z-score was computed to be:

$$\because t_{0.025,11} = -2.2010 < -0.7893$$

3 Part 3

3.1 Question

The weights of mature dogs of a certain breed approximately follow a normal distribution. Five dogs selected at random weighed 66, 63, 64, 62 and 65 pounds. A kennel club claims that the average weight for this breed is 60 pounds. Using the 0.05 level of significance, do we have reason to doubt this claim?

3.2 Answer

The null hypothesis, H_0, claims the mean weight of the breed is 60 pounds, while the alternative hypothesis, H_a, says otherwise.

$$H_0 : \mu = 60 \ vs \ H_a : \mu \neq 60$$

Since the population standard deviation is unknown, and the sample was $n < 30$, the t-score was calculated as follows:
The sample mean and standard deviation were calculated:

```r
weights <- c(66, 63, 64, 62, 65)
X <- mean(weights)
s <- sd(weights)
```

$$t = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} = \frac{64.0 - 60.0}{1.6/\sqrt{5}} = 5.6569$$

The following snippet was used to generate the Z-value and its probability:

```r
X <- 73.2
mu <- 72.4
sigma <- 2.1
n <- 35

z <- (X - mu)/(sigma/sqrt(n))
print(z) # print the Z-score
print(pnorm(z)) # print the probability
```

The Z-score was computed to be:

$$\because t_{0.025,4} = -2.7764 < 5.6569$$

References

```
# main.R
# This file contains the implementation of the functions in the Project 3
# 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(0)  # seed the random generators

scoreTemplate <-
    "\\begin{equation*}
    %s=\\frac{\\overline{X}-\\mu}{\\sfrac{\\sigma}{\\sqrt{n}}}
    =\\frac{%0.1f-%0.1f}{\\sfrac{%0.1f}{\\sqrt{%d}}}=%0.4f
    \\end{equation*}"

resultTemplate <- "\\begin{equation*}
    \\because %s=%0.4f %s %0.4f
    \\end{equation*}"

dumpComputation <- function(X, mu, sigma, n, alpha, distType, outputFile) {

    score <- (X - mu)/(sigma/sqrt(n))

    tableStr <- ""
    tableVal <- 0
    ineq <- ""

    if (distType == "Z") {

        tableStr <- paste0(distType, "_{", alpha/2, "}")
        tableVal <- qnorm(alpha/2)

    } else if (distType == "t") {

        tableStr <- paste0(distType, "_{", alpha/2, ",", n-1, "}")
        tableVal <- qt(alpha/2, df=n-1)

    }

    if (tableVal < score) {
        ineq <- "<"
    } else if (tableVal < score) {
        ineq <- ">"
    }

    # dump output to LaTex modules
    sink(
        paste0("latex_mods/", outputFile, "_out.tex"),
        append=FALSE, split=FALSE
    )
    cat(
        sprintf(scoreTemplate,
            distType, X, mu, sigma, n,
            score, distType)
    )
    sink(
        paste0("latex_mods/", outputFile, "_result.tex"),
        append=FALSE, split=FALSE
    )
    cat(
        sprintf(resultTemplate,
            tableStr, tableVal, ineq, score)
    )
    sink()  # return stdout to console

}
```

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

X <- 73.2
mu <- 72.4
sigma <- 2.1
n <- 35
alpha <- 0.05

dumpComputation(X, mu, sigma, n, alpha, "Z", "part1")


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

X <- 73.2
mu <- 75
s <- 7.9
n <- 12

dumpComputation(X, mu, s, n, alpha, "t", "part2")


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

weights <- c(66, 63, 64, 62, 65)
X <- mean(weights)
s <- sd(weights)
mu <- 60

dumpComputation(X, mu, s, length(weights), alpha, "t", "part3")


# # global variables
# NUMSAMPS <- 1000  # number of random samples per distribution

# randDist <- function(N, a, b, distType, outputFile) {
#     # Generates a random normal or binomial distribution.
#     #
#     # Args:
#     #   N: size of sample
#     #   a: First distribution parameter.
#     #       a = mu for normal distribution
#     #       a = n for binomial distribution
#     #   b: Second distribution parameter.
#     #       b = sigma for normal distribution
#     #       b = p for binomial distribution
#     #   distType: Type of distribution.
#     #       Options: "normal", "binomial"
#     #   outputFile: Name of LaTex output file

#     # initialize variables to hold data for the first sample
#     firstMean <- firstStd <- 0

#     # initialize distribution variables
#     mu <- sigma <- n <- p <- 0

#     # initialize empty arrays
#     sampMeans <- generatedData <- rep(0, times=NUMSAMPS)

#     # rename parameter values to distribution parameters for convenience
#     if (distType == "normal") {
#         mu <- a
#         sigma <- b
#     } else if (distType == "binomial") {
#         n <- a
#         p <- b
#     }

#     # generate 1000 samples
#     for (i in 1:NUMSAMPS) {

#         # generate distribution based on type chosen
#         if (distType == "normal") {
#             generatedData <- rnorm(N, mu, sigma)
#         } else if (distType == "binomial") {
#             generatedData <- rbinom(N, n, p)
#         }

#         # store the sample means in vector
#         sampMeans[i] = sum(generatedData)/N

#         if (i == 1) {

#             # store the first sample mean
#             firstMean = sum(generatedData)/N

#             # store the first sample standard deviation
#             if (distType == "normal") {
#                 # sigma/sqrt(N) if normal
#                 firstStd = sigma/sqrt(N)
#             } else if (distType == "binomial") {
#                 # sqrt(n*p*(1-p)/N) if binomial
#                 firstStd = sqrt(n*p*(1-p)/N)
#             }

#         }

#     }

#     # generate templates based on the distribution type and computed values
#     outputData <- ''
#     if (distType == "normal") {
#         outputData <- sprintf(
#             scoreTemplate,
#             firstMean, firstStd,
#             "\\mu", "\\mu", "\\sigma", "\\frac{\\sigma}{\\sqrt{n}}",
#             mu, mu,
#             mean(sampMeans), mu,
#             sigma, sigma,
#             sd(sampMeans), sigma/sqrt(N)
#         )
#     } else if (distType == "binomial") {
#         outputData <- sprintf(
#             scoreTemplate,
#             firstMean, firstStd,
#             "np", "np", "\\sqrt{np(1-p)}", "\\sqrt{\\frac{np(1-p)}{N}}",
#             n*p, n*p,
#             mean(sampMeans), n*p,
#             sqrt(n*p*(1-p)), sqrt(n*p*(1-p)),
#             sd(sampMeans), sqrt(n*p*(1-p)/N)
#         )
#     }

#     # dump output to LaTex modules
#     sink(outputFile, append=FALSE, split=FALSE)
#     cat(outputData)
#     sink()  # return stdout to console

#     return(sampMeans)

# }


# plotHist <- function(sampMeans, figureFile, binwidth) {
```

```
#     # Plot a histogram of the data
#     #
#     # Args:
#     #   sampMeans: the sample means generated from the random distributions
#     #   figureFile: file name of the output plot
#     #   binwidth: width of the bins of the histogram

#     histPlot <- ggplot() + aes(sampMeans) +
#         geom_histogram(binwidth=binwidth, color="black", fill="white") +
#         labs(y="Count", x="Sample Means")

#     # save plot to filename
#     ggsave(filename=paste0("figures/", figureFile), plot=histPlot)

# }


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

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

# sampMeans <- randDist(N, mu, sigma, "normal", "part1.tex")
# plotHist(sampMeans, "hist1.png", 0.1)


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

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

# sampMeans <- randDist(N, n, p, "binomial", "part2.tex")
# plotHist(sampMeans, "hist2.png", 0.1)


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

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

# sampMeans <- randDist(N, n, p, "binomial", "part3.tex")
# plotHist(sampMeans, "hist3.png", 0.025)
```