**Problem 1**

First, read the data from the provided csv file (data already clean, no need to “clean” it)

data <- read.csv("/Users/jakelille/Desktop/table3E2.csv")

Sort the data from smallest to biggest values

sorted\_data <- sort(data)

Calculate the n (length) number of data points, this will be used in parameters for functions

n <- length(sorted\_data)

Calculate the EXPECTED Quantiles (necessary for us to get the “shape” of our theoretical distributions to construct the probability plot); This will be used to compare the actual vs. expected to help us make the decision on what distribution best fits the data. I will later plot/generate a red line to visually show and test the distribution using this expected quantile value.

expected\_normal <- qnorm(ppoints(n))

I was having trouble getting the red-lines to show with the exponential, Weibull and lognormal distributions. After some research, I found I needed to adjust scaling (dependent on the distribution) to make sure the line would show.

For exponential, that involved setting the rate to 1 over the mean of the sorted data (this got the red-line in frame)

expected\_exponential <- qexp(ppoints(n), rate = 1/mean(sorted\_data))

For Weibull, I set the scale to the mean of the sorted data. In the pictures, we can start to see why originally we could not see the red line (as the x-axis was constrained to -2, -1, 0 , 1, etc., it now goes to the 200 to properly fit)

expected\_weibull <- qweibull(ppoints(n), shape = 2, scale = mean(sorted\_data))

Finally for lognormal, I included the mean AND standard deviation. With logs and going back and forth between lognormal to standard distributions you need both the mean and standard deviation. This allows us to properly frame the data to view the expected line.

expected\_lognormal <- qlnorm(ppoints(n), meanlog = mean(log(sorted\_data)), sdlog = sd(log(sorted\_data)))

This is only for the sake of output – when running ALL the code, it prints out our four distributions in a 2x2 grid so I can easily compare which to be of best fit.

par(mfrow = c(2, 2))

Calculating the NORMAL probability plot. As shown in class, we use qqnorm to generate the actual plots from the quantiles, inputting sorted data and titling our main graphic “Normal Probability Plot.

qqnorm(sorted\_data, main = "Normal Probability Plot")

This adds a line for the actual data (sorted data)

qqline(sorted\_data, col = 2)

This adds the “expected” red line (col = 3 for red) so we can compare

lines(expected\_normal, sorted\_data, col = 3)

A graph of a normal probability plot

Description automatically generated

Calculate exponential using qqplot. The qqplot function in R just tells R to plot against an expected distribution – we have to feed it the appropriate data and calculate quartiles for it to be “exponential”, “Weibull” or “Lognormal”. Thus, we feed the expected exponential for parameter one, against the sorted data. Then, draw the red line.

qqplot(expected\_exponential, sorted\_data, main = "Exponential Probability Plot", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

abline(0, 1, col = 2)

A graph with black dots

Description automatically generated

Same for the Weibull…

qqplot(expected\_weibull, sorted\_data, main = "Weibull Probability Plot", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

abline(0, 1, col = 2)

A graph of a graph

Description automatically generated

And finally same for the Lognormal…

qqplot(expected\_lognormal, sorted\_data, main = "Lognormal Probability Plot", xlab = "Theoretical Quantiles", ylab = "Sample Quantiles")

abline(0, 1, col = 2)

A graph of a graph

Description automatically generated

Upon reviewing all four plots, we can immediately rule out exponential and Weibull. Lognormal and normal are close, however there are more points on the redline of lognormal than normal, so **lognormal** **is the best distribution to model outpatient service time**.

**Problem 2**

Stem and leaf plot:

I was having trouble importing this particular csv file and utilizing it in the stem function, so I manually went into the file and copied the appropriate data values over to make a vector called “viscosity”.

viscosity <- c(13.3, 14.5, 15.3, 15.3, 14.3, 14.8, 15.2, 14.5, 14.6, 14.1, 14.3, 16.1, 13.1, 15.5, 12.6, 14.6, 14.3, 15.4, 15.2, 16.8, 14.9, 13.7, 15.2, 14.5, 15.3, 15.6, 15.8, 13.3, 14.1, 15.4, 15.2, 15.2, 15.9, 16.5, 14.8, 15.1, 17, 14.9, 14.8, 14, 15.8, 13.7, 15.1, 13.4, 14.1, 14.8, 14.3, 14.3, 16.4, 16.9, 14.2, 16.9, 14.9, 15.2, 14.4, 15.2, 14.6, 16.4, 14.2, 15.7, 16, 14.9, 13.6, 15.3, 14.3, 15.6, 16.1, 13.9, 15.2, 14.4, 14, 14.4, 13.7, 13.8, 15.6, 14.5, 12.8, 16.1, 16.6, 15.6)

Then, called the stem function to generate our “stem and leaf plot”

stem(viscosity)

R outputs the stem and leaf plot directly to the console, so I needed to screenshot this and it follows on the next page…

A white background with black text

Description automatically generated

Off first glance, we could say the data follows somewhat of a normal distribution – most values contained in the middle, with few on the polar ends (12 lower, 17 upper). It is skewed a little more to the right. The mean of the data is between 14 and 15, closer to 15 (around ~14.8).

Histogram…

Used the hist() function in R, set the title (main), then x and y labels appropriately…

hist(viscosity, main = "Viscosity Histogram", xlab = "Viscosity Values", ylab = "Frequency")

A graph of a person with histogram

Description automatically generated

A histogram is basically a simplified stem and leaf plot. We can draw the same/similar conclusions: the mean is between 14-15, probably closer to 14.8 still is my guess off the visual. It is more apparent that the data is right-skewed however, as we can see more values on the right than left.

Box plot…

Used the boxplot() function in R. Fairly straight-forward, as it has the same parameters as the histogram.

boxplot(viscosity, main = "Viscosity Boxplot", ylab = "Viscosity Values")

A diagram of a box plot

Description automatically generated

The middle line gives us our median, but there are no labels given with the default boxplot() function, so I simply ran the mean function to find it.

median(viscosity)

A white background with blue text

Description automatically generated

Our “interquartile range” lies within Q1 and Q3 with box plots, and that is illustrated by the shaded grey box. Q1 is just above 14, and Q3 around 15.5. The two lines on the polar ends represent our minimum and maximum values, being 12.6 and 17.0.

We can say most of our values lie within the interquartile range, with our median being 14.9 and spreading between 12.6 and 17.0.

**Describe the distribution of the data (given the plots)**..

Based off the following plots, we can say that the viscosity data is a right-skewed distribution with a central-tendency around 14.9 (median). Most of the values in the data set fall in the range of 14 to 15.5, with some outliers present, specifically towards the higher end. Of course, the distribution is not perfectly normal, however, it exhibits characteristics of normal distributions, specifically having some central-tendency around the mode, or most occurring values giving it that somewhat bell-shape curve.

**Problem 3**

I will be using a normal distribution for this problem. As we discussed in class, the Central Limit Theorem states that independent of the original distribution of the population, any sample mean with a significantly large enough sample size will be **approximately normal**… A significant sample size is 30 >=, and this problem states n being 100.

First, I encode the given values into variables within R so it’s easy to utilize in functions…

population\_mean <- 60

population\_sd <- 4

sample\_size <- 100

The sample mean in this instance is equal to the population mean, so the **sample mean is 60**. First, we don’t have the individual values of the sample to calculate the actual sample mean, and second, we are assuming that this sample is representative of the population, therefore it inherits the same mean of 60.

Then, using the formula for calculating the sample mean standard deviation, I store that in a variable “sample\_mean\_sd”.

sample\_mean\_sd <- population\_sd / sqrt(sample\_size)

This yields our result for the **Sample Standard Deviation**, which is 0.4A close-up of a computer screen

Description automatically generated

*What is the approximate probability that the sample mean (X Bar) exceeds 61 mph?*

This is a normal distribution, meaning we will use **Z-Scores** to help us determine and run a probability test to calculate the probability.

We set a variable x equal to 61, the value we want to see if it exceeds (in mph)…

x <- 61

Then, we calculate the appropriate Z **test statistic** from the formula given in class…

z <- (x - population\_mean) / sample\_mean\_sd

Then, we can use pnorm() as given in class to calculate the probability of the left. **However**, we need the **probability to the right**, or in context of the problem, the probability of it **exceeding** 61 (the upper portion of the distribution). Therefore, we need to do 1 – pnorm().

probability\_exceeds\_61 <- 1 - pnorm(z)

probability\_exceeds\_61

So, our probability is 0.006209665 that the speed will exceed 61 mph.A screenshot of a computer

Description automatically generated

*What is the approximate probability that the sample mean (X bar) is between 59.2 mph and 60.4 mph?*

For this, we are going to have to set our upper and lower limits to the given values: lower limit 59.2 and upper 60.4 (in mph). We will need both upper and lower z-scores, and then subtract the upper from the lower to yield our probability of our value being between these two limits.

Store our limits in variables to be used in functions…

lower\_limit <- 59.2

upper\_limit <- 60.4

Calculate our individual Z-Scores for the lower and upper (following the formula given in class of the given mean value – population/sample mean / sample sd)

z\_lower <- (lower\_limit - population\_mean) / sample\_mean\_sd

z\_upper <- (upper\_limit - population\_mean) / sample\_mean\_sd

Subtract the lower limit probability from the upper limit to yield the probability of it being between these two values, then store it in probability\_between so we can output the value…

probability\_between <- pnorm(z\_upper) - pnorm(z\_lower)

Output the value…

probability\_between

A screenshot of a computer program

Description automatically generated

Therefore, our probability of a value being between 59.2 and 60.4 mph is 0.8185946, which is high. This makes sense considering the mean is 60, and you expect most values to fall around there assuming a normal distribution.

**Problem 4**

*The probability that the lifetime exceeds 300 hours (utilizing a lognormal distribution…)*

Store the appropriate given values in R variables to be used in functions…

meanlog <- 6

sdlog <- 1.2

x <- 300

Next, we need to calculate the probability. Since we are looking for the exceeding X value, we are looking towards the right side of the distribution, ie. (1 – plnorm). The plnorm() function calculates the Z-test statistic for a lognormal distribution, this will give us the Z-Score for X = 300 or below. 1-plnorm() will yield the probability of the equipment exceeding 300 hours.

probability\_exceeds\_x <- 1 - plnorm(x, meanlog = meanlog, sdlog = sdlog)

probability\_exceeds\_x

A computer screen shot of a white board

Description automatically generated

The probability that the medical laser’s lifetime exceeds 300 hours is 0.5974…. etc. This equates to roughly a 60% probability the equipment exceeds 300 hours, which is fairly-likely.

*Given the cumulative distribution function, find the probability that a new spring survives 30 hours…*

In this case, x = 30, therefore, greater than 0. In this piecewise function we use the bottom of the two. We are already given the equation to find the probability, including the values needed. Lambda = 0.001 and K = 2. This is simply a plug-in equation.

Store the appropriate values into variables

lambda <- 0.001

K <- 2

x <- 30

The actual math here, utilizing R’s exp() function, which raises x to the power of e, storing the result in F.

F <- 1 - exp(-((lambda \* x)^K))

Print f to the console

F

A white screen with blue text

Description automatically generated

The probability that the new spring survives 30 hours is 0.0008995… etc. In other-words, it is extremely low and unlikely that the new spring will survive 30 hours.