STM4PSD – Workshop 8 Solutions

1. The code for each part is shown below. You should run it yourself to verify the output.

```
q4.data <- c(14.3, 20.2, 13.5, 17.4)
# Mean:
mean(q4.data)
# Variance:
var(q4.data)
# Standard error:
sd(q4.data)/sqrt(length(q4.data))
# Lower bound of the confidence interval:
mean(q4.data) - 1.96 * sd(q4.data)/sqrt(length(q4.data))
# Upper bound of the confidence interval:
mean(q4.data) + 1.96 * sd(q4.data)/sqrt(length(q4.data))
```

There is a slight shortcut to calculate the confidence interval, using the vector techniques that R provides. See below.

```
q5.data <- c(14.3, 20.2, 13.5, 17.4, 20.0, 15.2)
# Mean:
mean(q5.data)
# Variance:
var(q5.data)
# Standard error:
sd(q5.data)/sqrt(length(q5.data))
# The confidence interval as a vector containing the lower and upper bound
mean(q5.data) + c(-1,1) * 1.96 * sd(q5.data)/sqrt(length(q5.data))
```

In the above code, multiplying by `c(-1,1)` is encoding the “plus or minus” part of the definition.

```
q6.data <- c(14.3, 20.2, 13.5, 17.4, 12.2, 23.3)
# Mean:
mean(q6.data)
# Variance:
var(q6.data)
# Standard error:
sd(q6.data)/sqrt(length(q6.data))
# The confidence interval as a vector containing the lower and upper bound
mean(q6.data) + c(-1,1) * 1.96 * sd(q6.data)/sqrt(length(q6.data))
```

2. Two possible answers are shown.

```
interval <- function(data) {
    mean <- mean(data)
    sd <- sd(data)
    se <- sd(data)/sqrt(length(data))
    lower <- mean - 1.96*se
    upper <- mean + 1.96*se
    c(lower, upper)
}
```

Or, a more compact version:

```
interval <- function(data) {
    mean(data) + c(-1,1) * 1.96 * sd(data)/sqrt(length(data))
}
```

3. Using `qnorm((0.95+1)/2)` results in 1.959964 which is, to 2 decimal places, 1.96.

4. The first approach shown here is the "manual" approach.

   Using qnorm((0.8+1)/2) gives a critical value of 1.28, so an 80% confidence interval is given by

```
q5.data <- c(14.3, 20.2, 13.5, 17.4, 20.0, 15.2)
mean(q5.data) + c(-1,1) * 1.28 * sd(q5.data)/sqrt(length(q5.data))
```

   Using qnorm((0.9+1)/2) gives a critical value of 1.64, so an 80% confidence interval is given by

```
q5.data <- c(14.3, 20.2, 13.5, 17.4, 20.0, 15.2)
mean(q5.data) + c(-1,1) * 1.64 * sd(q5.data)/sqrt(length(q5.data))
```

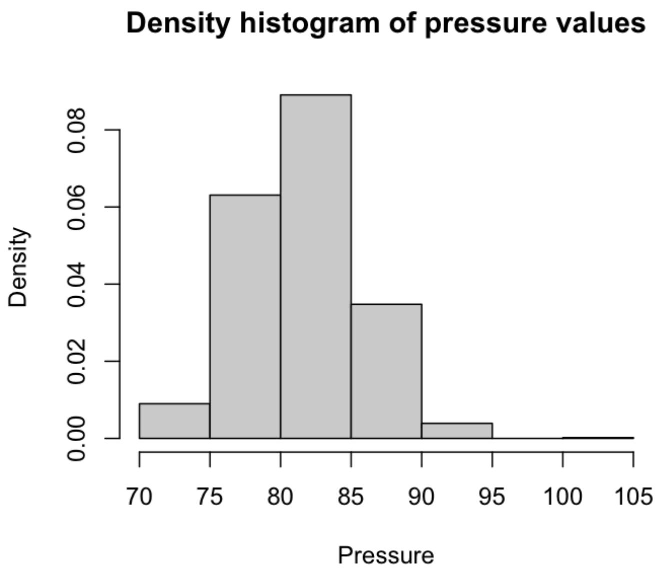
   A more robust method would be to define the interval function to also take an optional "confidence level" argument.

```
interval2 <- function(data, confidence=0.95) {
    critical.value <- qnorm((1+confidence)/2)
    mean(data) + c(-1,1) * critical.value * sd(data)/sqrt(length(data))
}
q5.data <- c(14.3, 20.2, 13.5, 17.4, 20.0, 15.2)
interval2(q5.data, 0.8) # 80% confidence interval
interval2(q5.data, 0.9) # 90% confidence interval
```

5.

```
hist(pressure1$Pressure, freq=FALSE, xlab="Pressure",
                         main="Density histogram of pressure values")
```

   This will result in the following diagram:

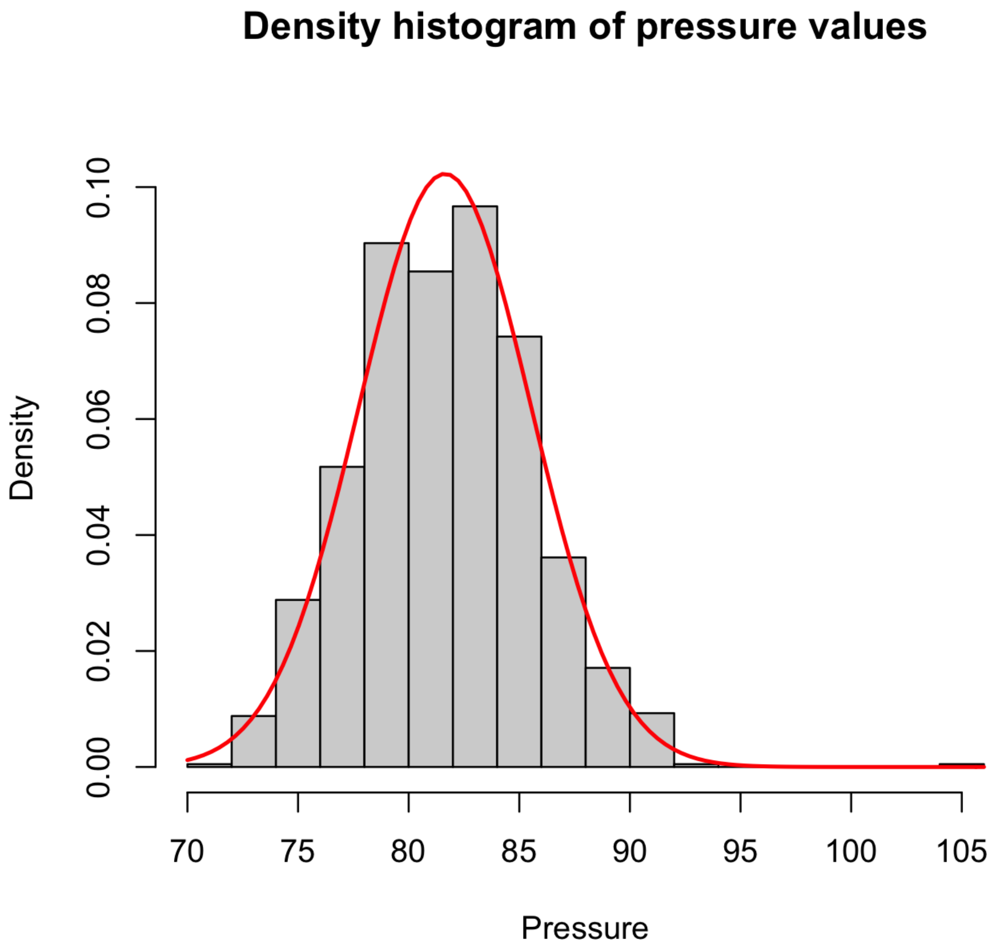


6.

```
hist(pressure1$Pressure, freq=FALSE, xlab="Pressure",
                         main="Density histogram of pressure values",
                         breaks=20)
```

7. Running mean(pressure1$Pressure) and sd(pressure1$Pressure) gives, to two decimal places, a mean of 81.64 and a standard deviation of 3.90. The code below will give the desired diagram.

```
hist(pressure1$Pressure, freq=FALSE, xlab="Pressure",
                         main="Density histogram of pressure values",
                         breaks=20,
                         ylim=c(0,0.11))
curve(dnorm(x, mean=81.64, sd=3.90), add=TRUE, lwd=2, col="red")
```



8. These calculations are performed similar to the above.
9. For example, to create the 95% confidence interval for the `pressure1.csv` file:

```
mean <- mean(pressure1$Pressure)
sd <- sd(pressure1$Pressure)
se <- sd/sqrt(length(pressure1$Pressure))
interval <- mean + c(-1,1)*1.96*se
```

This will store the interval in the `interval` variable, and inspecting its contents gives the interval $(81.39903, 81.87703)$. Or, using the interval function from Q2, you could write this to get the same result:

```
interval(pressure1$Pressure)
```