2021bcs0175

Hazarathayya

LAB – 3

DATA WAREHOUSING AND DATA MINING

Q1. Read the pupae data. Convert 'CO, _treatment' to a factor. Inspect the levels of this factor variable.

Code:

```
print(getwd())

setwd("College/SEM VI/Data warehousing and data mining/Lab3/")


data <- read.csv("pupae.csv")

data

data$CO2_treatment <- as.factor(data$CO2_treatment)

levels(data$CO2_treatment)
```

Output:

```
64    elevated          400       0       0.331 1.962
> data$CO2_treatment <- as.factor(data$CO2_treatment)
> levels(data$CO2_treatment)
[1] "280" "400"
```

Q2. Make a scatter plot of Frass vs. 'PupalWeight', with blue solid circles for a CO, concentration of, 280ppm and red for 400ppm. Also add a legend.

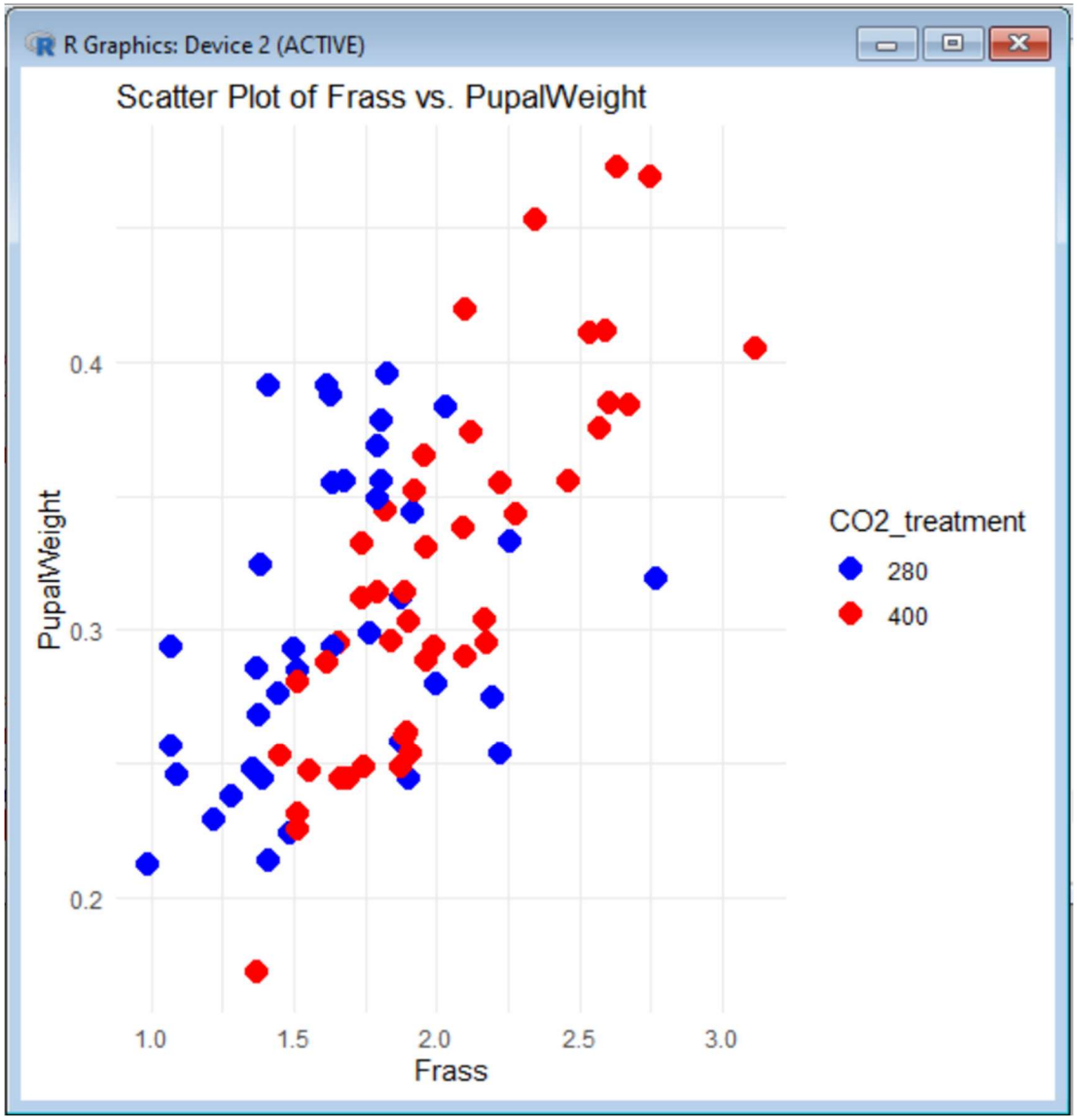
Code:

```
data <- read.csv("pupae.csv")

data

grps <- as.factor(df$CO2_treatment)

colors = c("#FF0000", "#0000FF")


plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color = CO2_treatment)) +
  geom_point(shape = 16, size = 4) +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
  theme_minimal()
```

Display the plot with legend

print(plot)

Output:

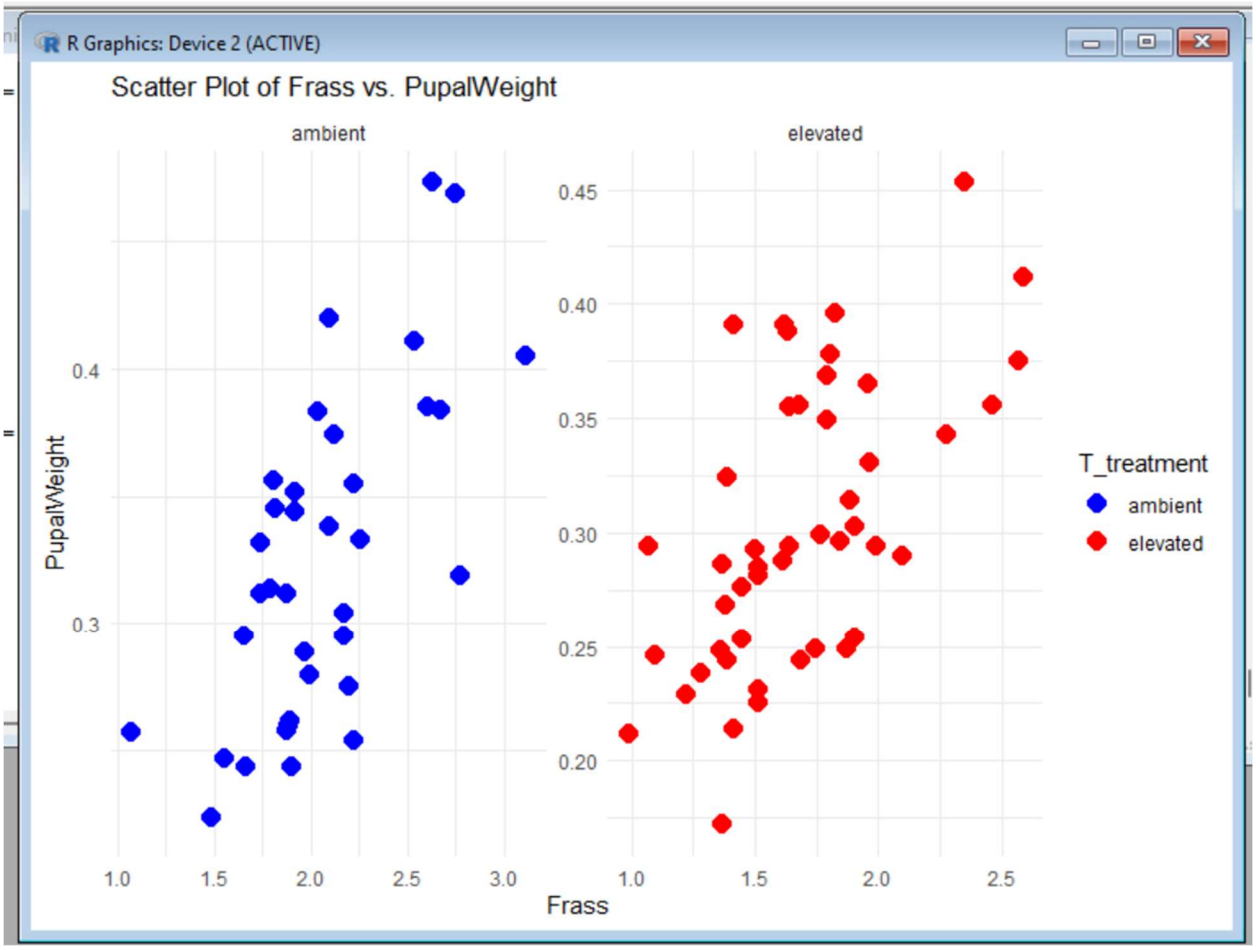


3. Make two plots (two plots side by side), one with the 'ambient' temperature treatment, one with 'elevated'.

Code:

```
plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color = T_treatment)) +
  geom_point(shape = 16, size = 4) +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
  theme_minimal() +
  facet_wrap(~T_treatment, scales = "free")

# Display the plot with two facets
print(plot)
```

Output:



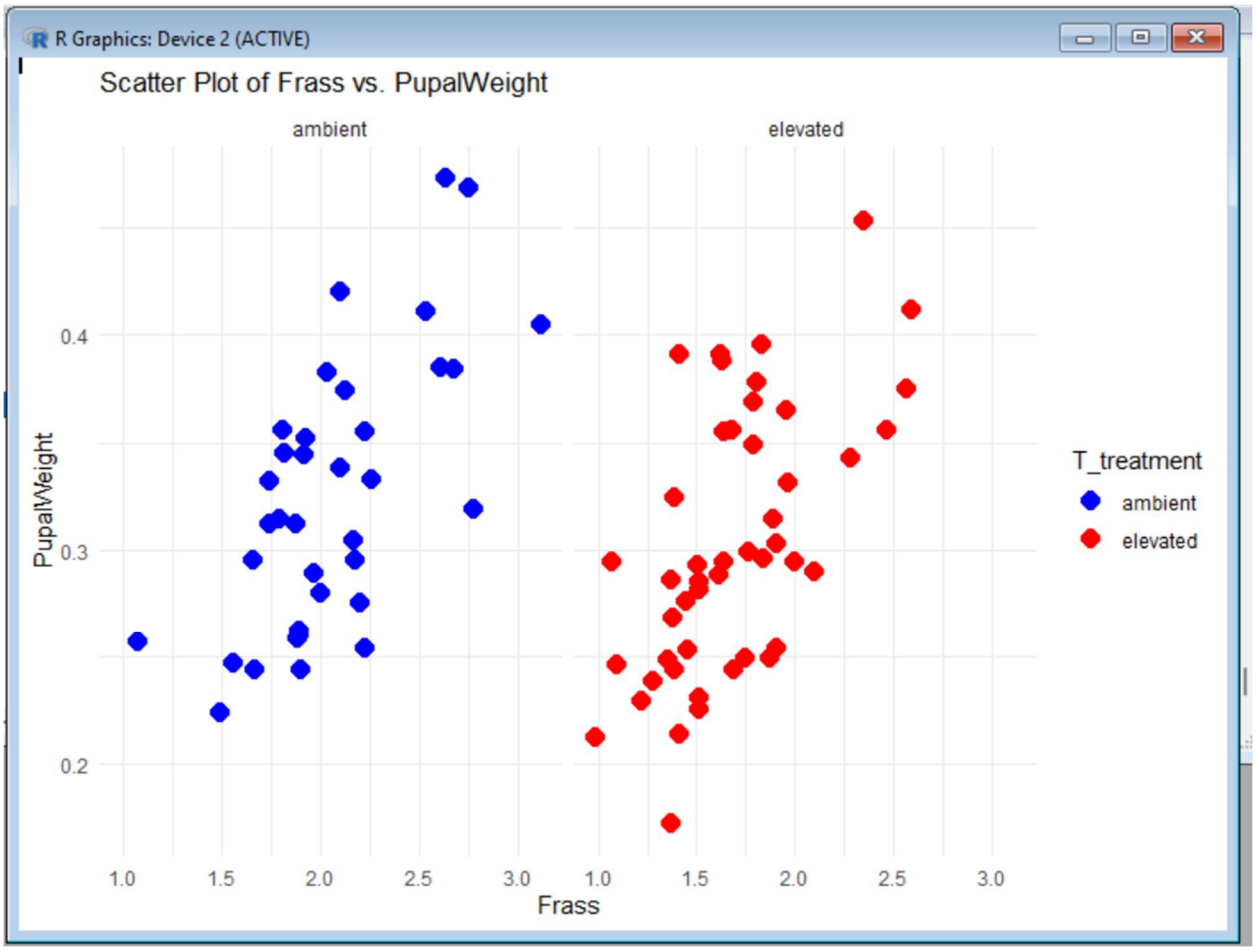
4. In the above plot, make sure that the X and Y axis ranges are the same for both plots.

Code:

```
plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color = T_treatment)) +
  geom_point(shape = 16, size = 4) +
  scale_color_manual(values = c("blue", "red")) +
```

```
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
  theme_minimal() +
  facet_wrap(~T_treatment, scales = "fixed")

# Display the plot with two facets
print(plot)
```

Output:



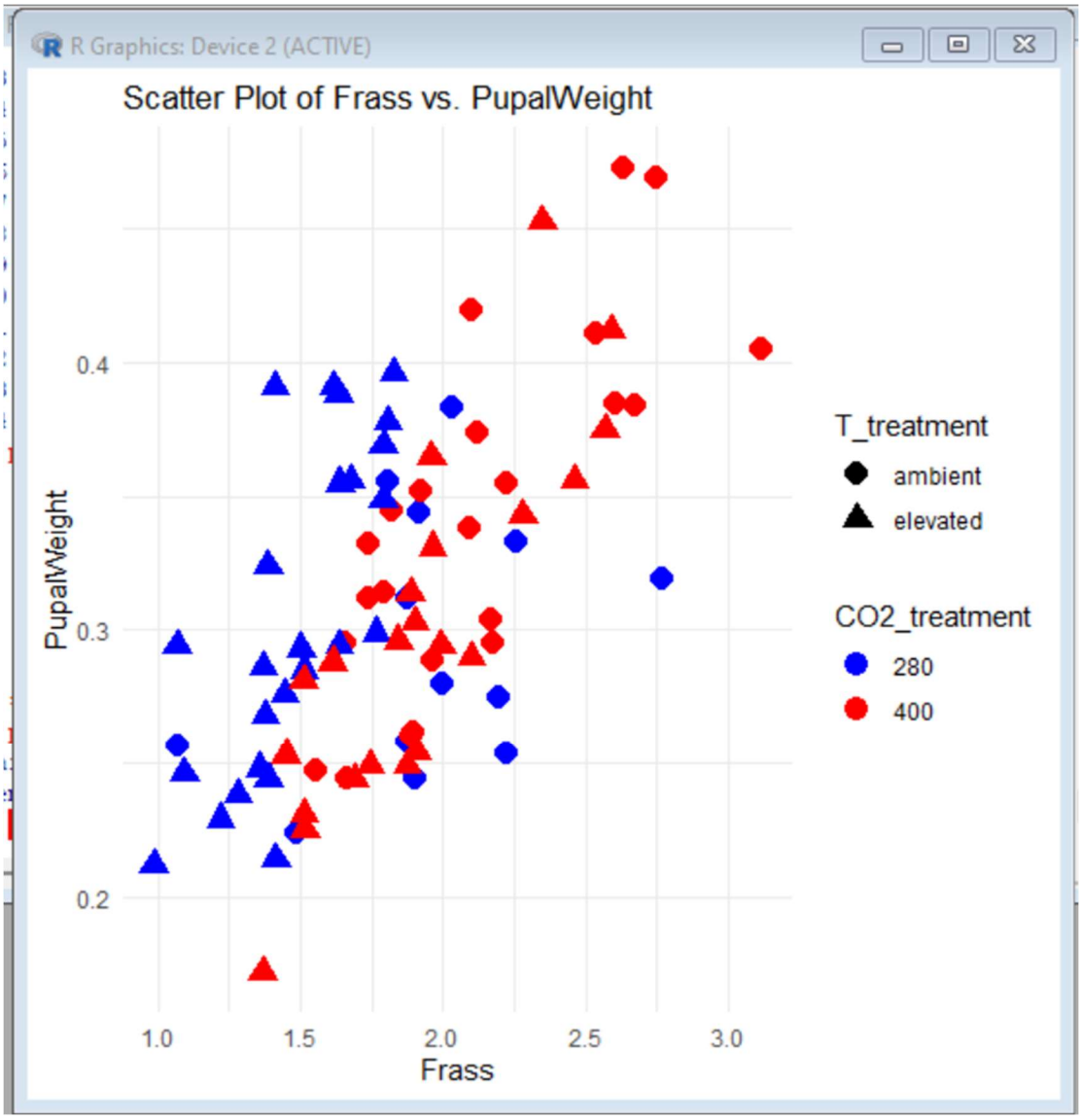
5. Instead of making two separate plots, make one plot that uses different colors for the CO(2) treatments and different symbols for the 'ambient' and 'elevated' temperature treatments.

Code:

```
plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color = CO2_treatment, shape = T_treatment)) +
```

```
  geom_point(size = 4) +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
  theme_minimal()

# Display the plot
print(plot)
```

Output:

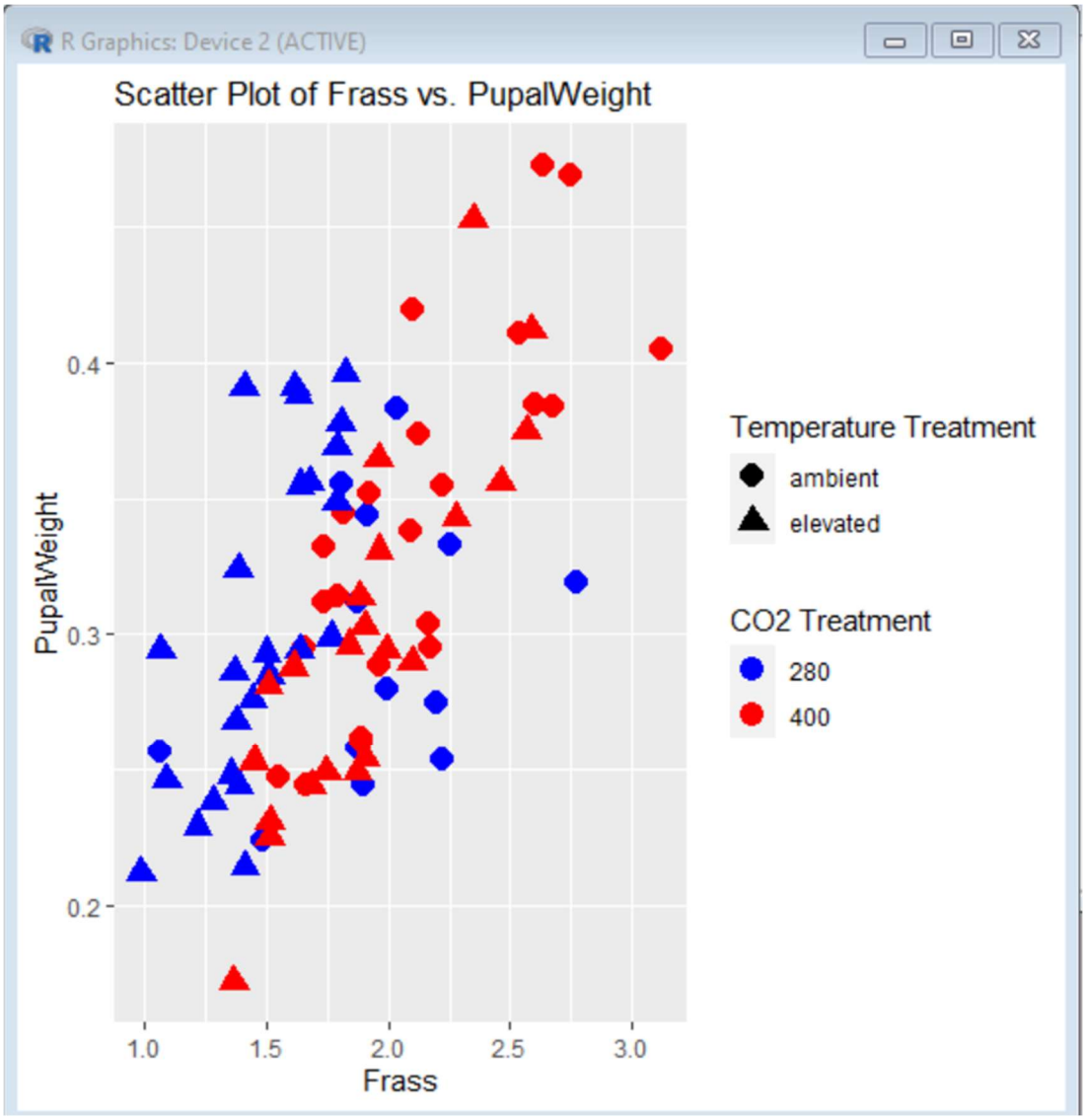


6. Add two legends to the above plot, one for the temperature treatment (showing different plotting symbols), and one for the CO2 treatments (showing different colours).

Code:

```
plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color =
CO2_treatment, shape = T_treatment)) +
  geom_point(size = 4) +
  scale_color_manual(values = c("blue", "red")) +
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
       guides(
    shape = guide_legend(title = "Temperature Treatment"),
    color = guide_legend(title = "CO2 Treatment")
  )
  theme_minimal()
```

Output:

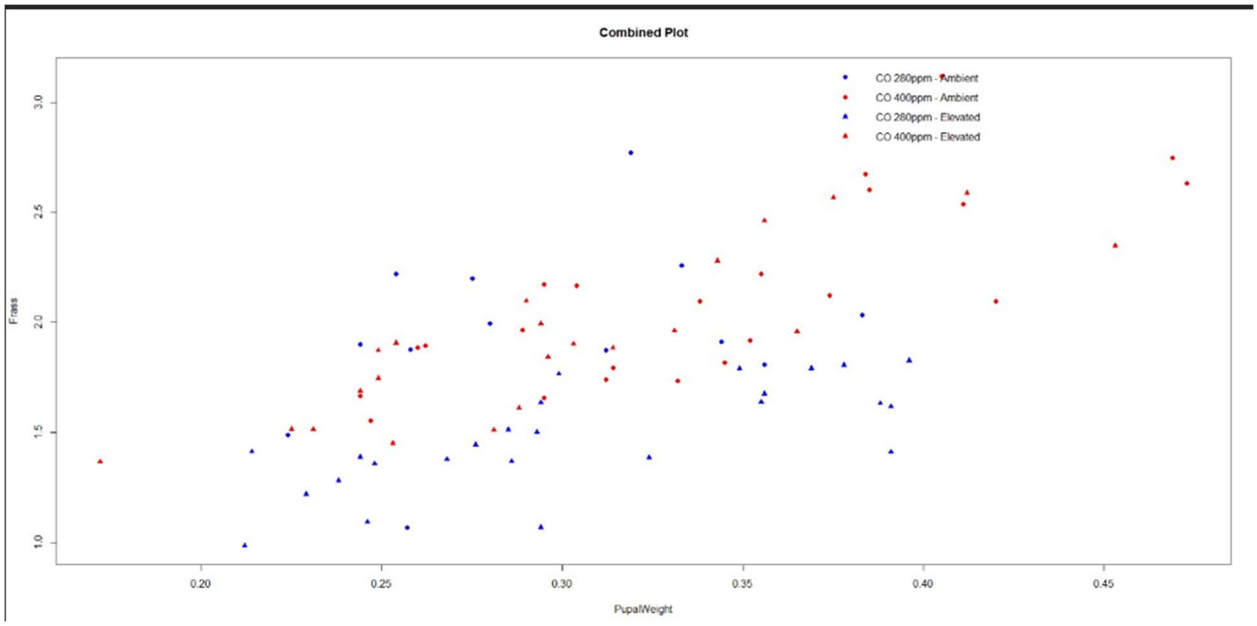


7. Generate the same plot as Q.(6) but this time add a single legend that contains symbols and colours for each treatment combination (CO2 : T).

Code:

```
plot <- ggplot(data, aes(x = Frass, y = PupalWeight, color =
interaction(CO2_treatment, T_treatment), shape =
interaction(CO2_treatment, T_treatment))) +
  geom_point(size = 4) +
  scale_color_manual(values = c("blue", "red"), name =
"CO2:Temperature") +
  labs(title = "Scatter Plot of Frass vs. PupalWeight",
       x = "Frass",
       y = "PupalWeight") +
  theme_minimal()

# Display the plot
print(plot)
```

Output:



8. Create a sample of random data points from a normal distribution with mean •€€ and standard deviation, and store the result in a vector. Plot a histogram and a boxplot of the vector you just created.

Code:

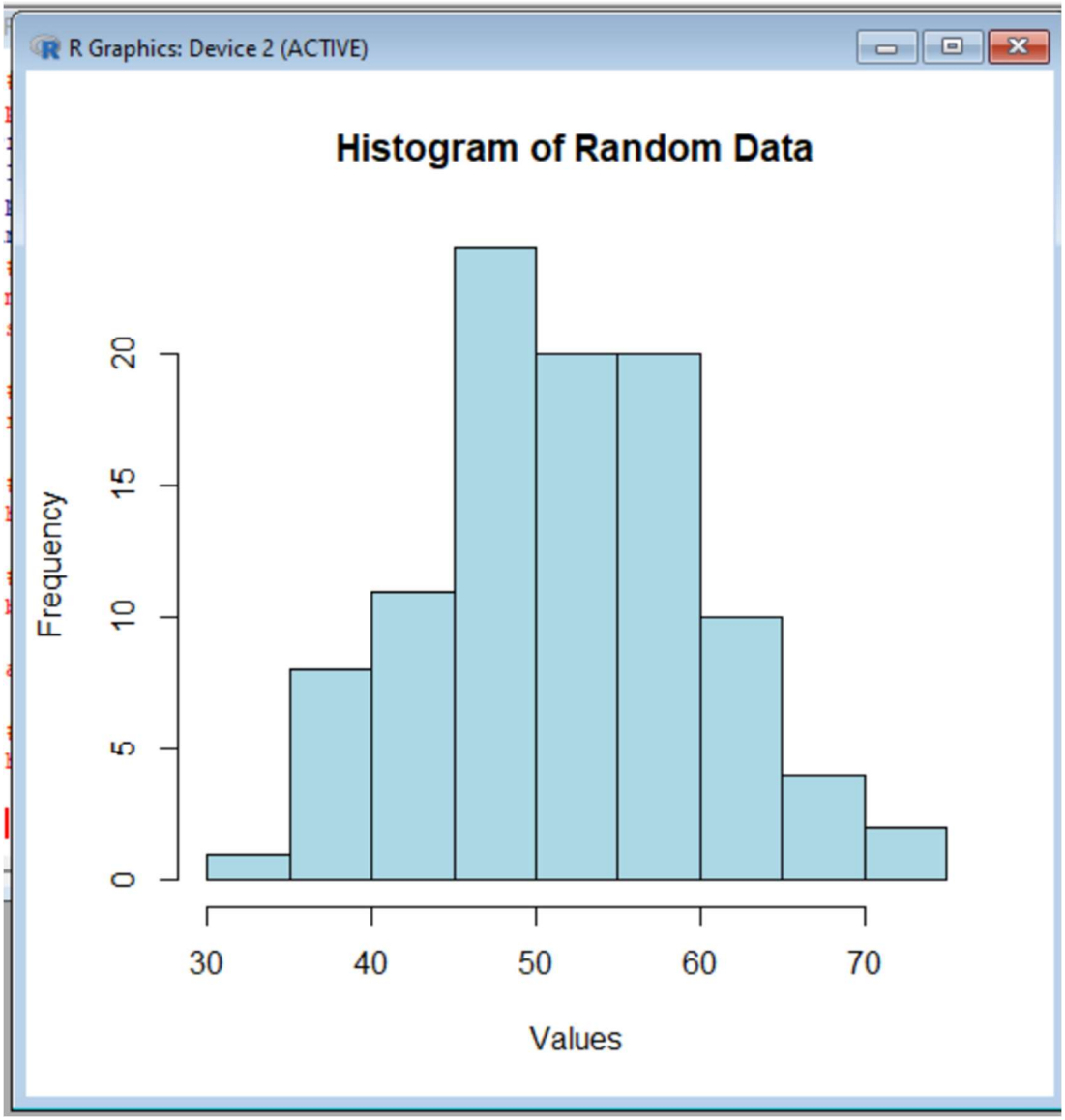
Set the mean and standard deviation

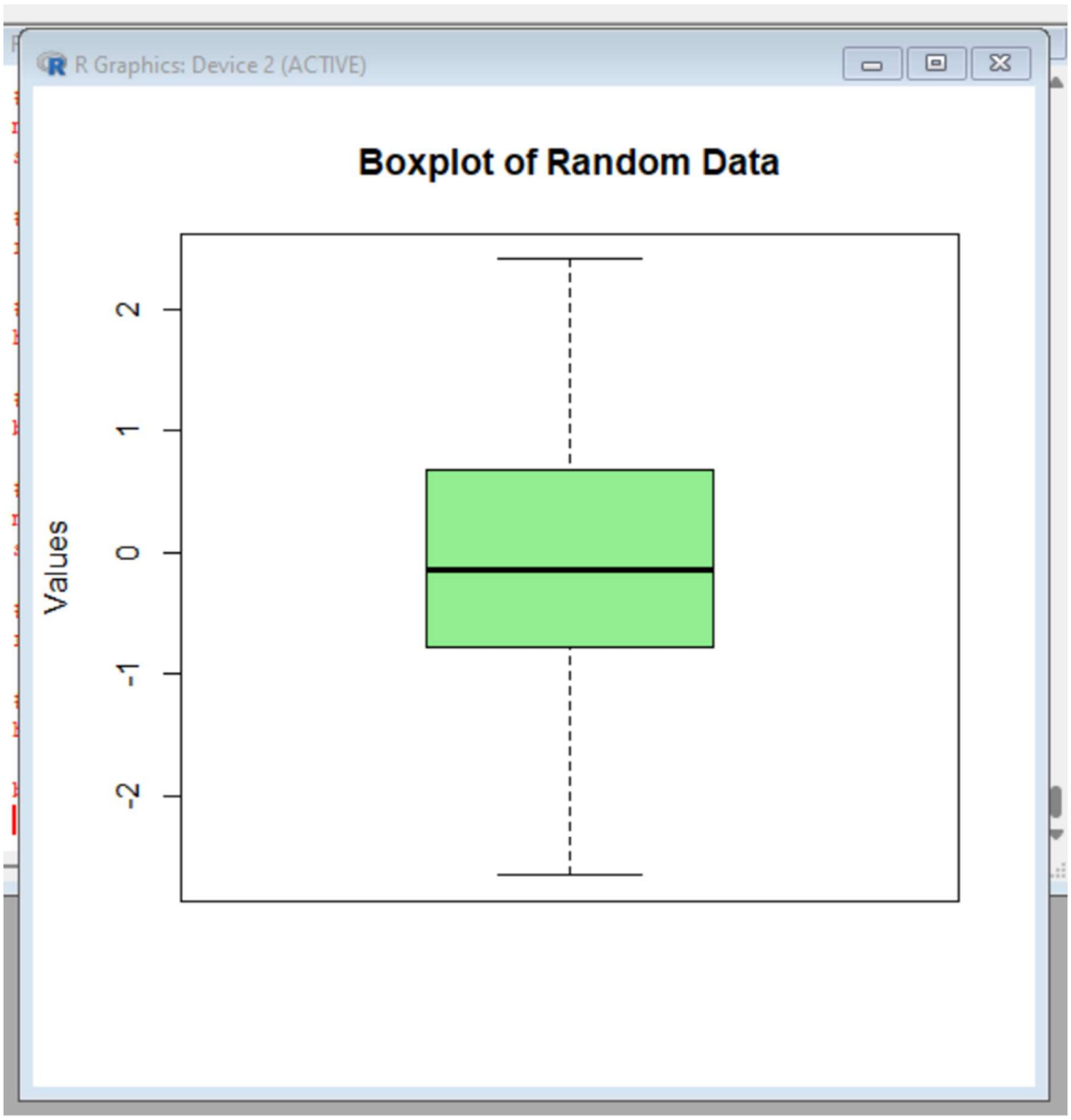
mean_value <- 0

sd_value <- 1

```
# Generate a random sample from a normal distribution
random_data <- rnorm(100, mean = mean_value, sd = sd_value)

# Plot a histogram
hist(random_data, main = "Histogram of Random Data", xlab =
"Values", col = "lightblue", border = "black")

# Plot a boxplot
boxplot(random_data, main = "Boxplot of Random Data", ylab =
"Values", col = "lightgreen", border = "black")
```

Output:


R Graphics: Device 2 (ACTIVE)
Histogram of Random Data
Frequency
0
5
10
15
20
30
40
50
60
70
Values



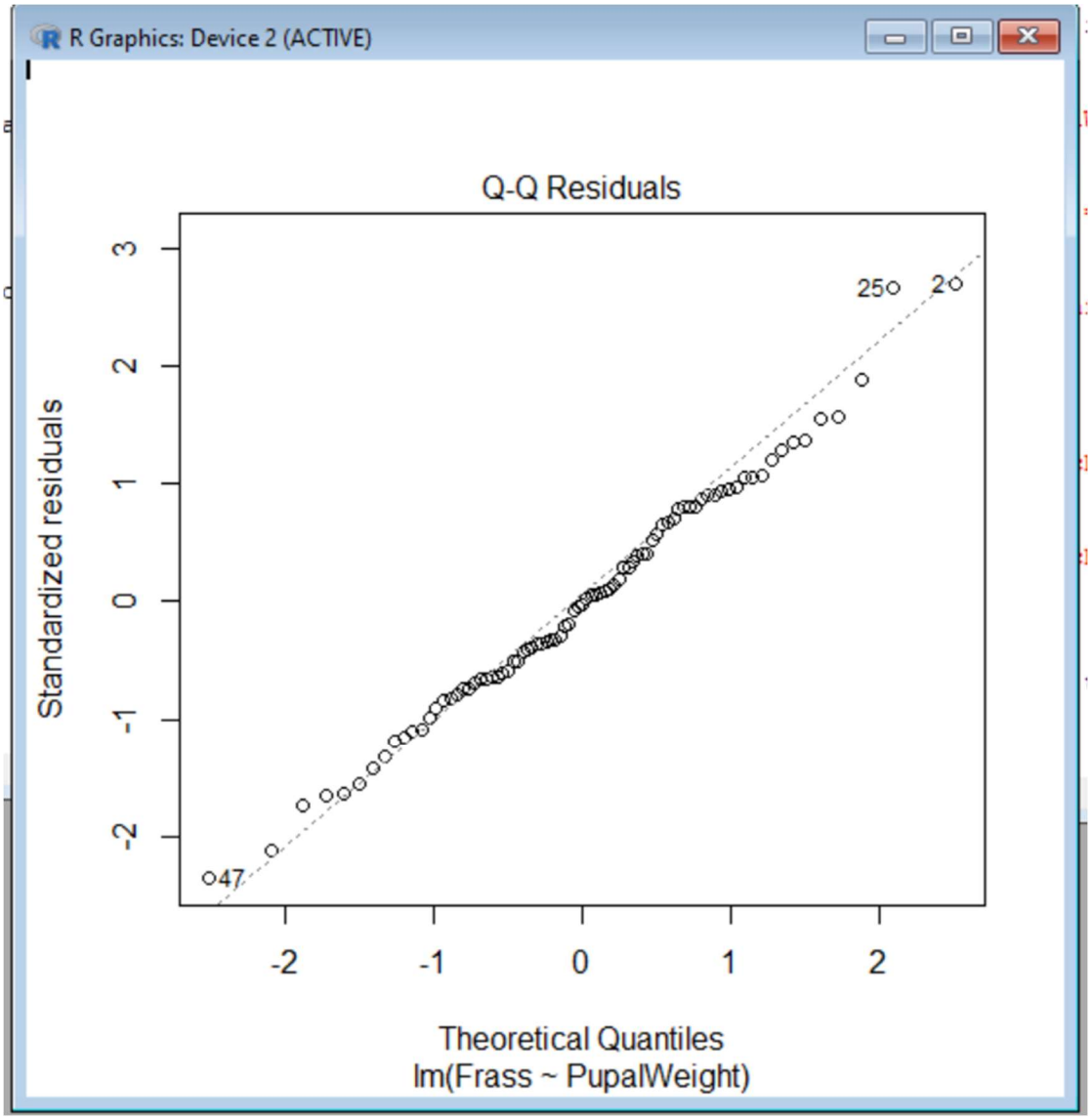
9. Use the pupae data. Perform a simple linear regression of Frass on PupalWeight. Produce and inspect the following: i. Plots of the data. ii. Display Residual and qq plot

Code:

```
library(ggplot2)
library(car)

# Simple linear regression
lm_model <- lm(Frass ~ PupalWeight, data = data)

# Plots of the data
plot1 <- ggplot(data, aes(x = PupalWeight, y = Frass)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Simple Linear Regression",
       x = "PupalWeight",
       y = "Frass")

# Display Residual plot
plot2 <- plot(lm_model, which = 1)

# Display qq plot
```

```
plot3 <- plot(lm_model, which = 2)


# Combine plots
library(gridExtra)
grid.arrange(plot1, plot2, plot3, nrow = 2)
```

Output:


R Graphics: Device 2 (ACTIVE)
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(Frass ~ PupalWeight)
25
2
47
-2
-1
0
1
2
3