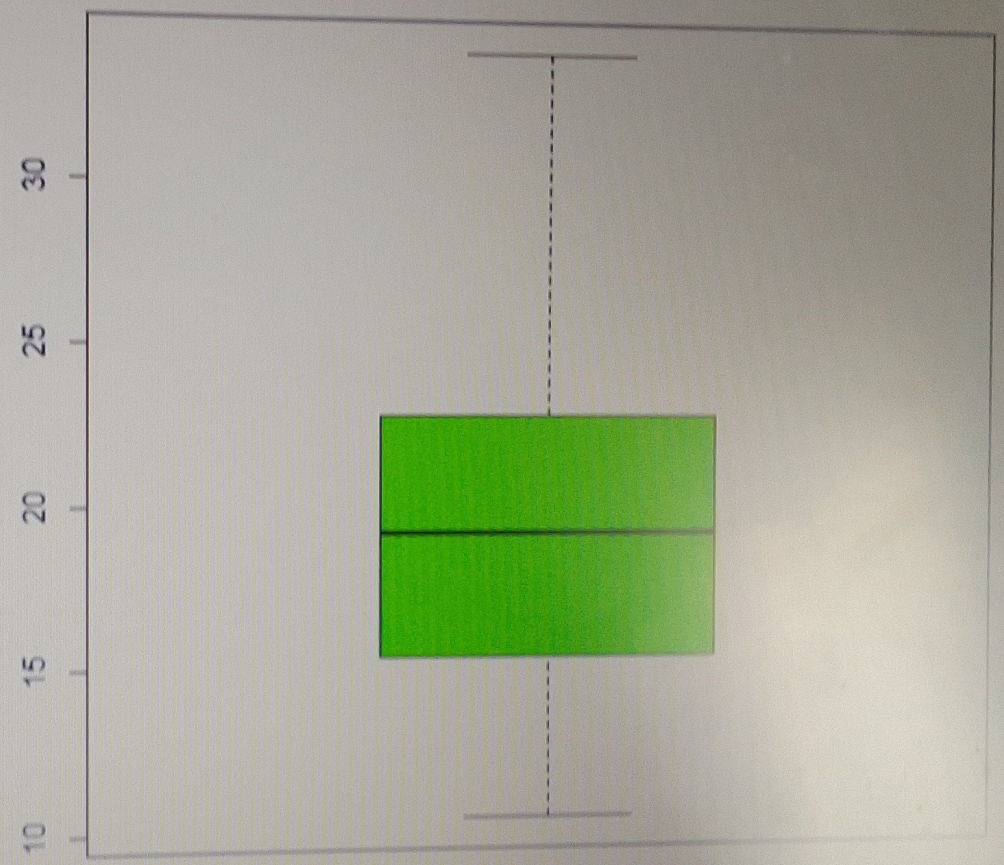
```
(2022-06-23 ucrt) -- "Funny-Looking Kid"
2023 The R Foundation for Statistical Computing
64-w64-mingw32/x64 (64-bit)
```

Untitled - R Editor

```
boxplot(mtcars$mpg, col="green")
```

R Graphics: Device 2 (ACTIVE)



Untitled - R Editor

```
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))
```

```
"# Defining vector
x"
>
> # Print Harmonic Mean
> print(1 / mean(1 / x))
[1] 11.66388
>
```

RGui (64-bit)

e Edit Packages Windows Help

C:\Users\pavani.k\Documents\central tendency - R Editor

```
###mean
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))

###mode
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
       23, 40, 23, 14, 12, 56,
       23, 29, 56, 37, 45, 1, 25, 8)

# Generate frequency table
y <- table(x)

# Print frequency table
print(y)

# Mode of x
m <- names(y)[which(y == max(y))]

# Print mode
print(m)
```

```
x
 1  3  5  7  8 12 13 14 20 23 25 29 37 39 40 45 56
 1  1  1  1  1  1  1  1  1  4  1  1  1  1  1  1  2
>
> # Mode of x
> m <- names(y)[which(y == max(y))]
>
> # Print mode
> print(m)
[1] "23"
>
```

```
###mean::
# Defining vector
x <- c(1, 5, 8, 10)

# Print Harmonic Mean
print(1 / mean(1 / x))

#####median::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
       23, 40, 23, 14, 12, 56, 23)

# Print Median
median(x)

###mode::
# Defining vector
x <- c(3, 7, 5, 13, 20, 23, 39,
       23, 40, 23, 14, 12, 56,
       23, 29, 56, 37, 45, 1, 25, 8)

# Generate frequency table
y <- table(x)

# Print frequency table
print(y)
```

```
> print(m)
[1] "23"
> # Defining vector
> x <- c(3, 7, 5, 13, 20, 23, 39,
+        23, 40, 23, 14, 12, 56, 23)
>
> # Print Median
> median(x)
[1] 21.5
>
```

```
-2.0743  -0.9062  -0.5223   0.8847   1.7964

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -13.8090     5.3626  -2.575   0.0100 *
IQ            0.4534     0.1771   2.560   0.0105 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family take

    Null deviance: 55.352  on 39  degrees of f
Residual deviance: 46.996  on 38  degrees of f
AIC: 50.996

Number of Fisher Scoring iterations: 3

>
> # saving the file
> dev.off()
null device
          1
>
```

Untitled - R Editor

```
# Print data frame
print(df)

# output to be present as PNG file
png(file="LogisticRegressionGFG.png")

# Plotting IQ on x-axis and result on y-axis
plot(IQ, result, xlab = "IQ Level",
ylab = "Probability of Passing")

# Create a logistic model
g = glm(result~IQ, family=binomial, df)

# Create a curve based on prediction using the regression model
curve(predict(g, data.frame(IQ=x), type="resp"), add=TRUE)

# This Draws a set of points
# Based on fit to the regression model
points(IQ, fitted(g), pch=30)

# Summary of the regression model
summary(g)

# saving the file
dev.off()
```

```
> df <- data.frame(x = 182)
> res <-  predict(model, df)
> cat("\nPredicted value of a person
+                  with height = 182")

Predicted value of a person
                 with height = 182> print(res)
       1
84.9098
>
> # Output to be present as PNG file
> png(file = "linearRegGFG.png")
>
> # Plot
> plot(x, y, main = "Height vs Weight
+                    Regression model")
> abline(lm(y~x))
>
> # Save the file.
> dev.off()
null device
          1
>
```

Untitled - R Editor

```
       75, 72, 62, 49)

# Create a linear regression model
model <- lm(y~x)

# Print regression model
print(model)

# Find the weight of a person
# With height 182
df <- data.frame(x = 182)
res <-  predict(model, df)
cat("\nPredicted value of a person
                with height = 182")
print(res)

# Output to be present as PNG file
png(file = "linearRegGFG.png")

# Plot
plot(x, y, main = "Height vs Weight
                   Regression model")

abline(lm(y~x))

# Save the file.
dev.off()
```

Help

Untitled - R Editor

```
#define Min-Max normalization function
min_max_norm <- function(x) {
    (x - min(x)) / (max(x) - min(x))
  }

#apply Min-Max normalization to first four columns in iris dataset
iris_norm <- as.data.frame(lapply(iris[1:4], min_max_norm))

#view first six rows of normalized iris dataset
head(iris_norm)
```

```
+   }
>
> #apply Min-Max normalization to first four columns in iris dataset
> iris_norm <- as.data.frame(lapply(iris[1:4], min_max_norm))
>
> #view first six rows of normalized iris dataset
> head(iris_norm)
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   0.22222222   0.6250000   0.06779661  0.04166667
2   0.16666667   0.4166667   0.06779661  0.04166667
3   0.11111111   0.5000000   0.05084746  0.04166667
4   0.08333333   0.4583333   0.08474576  0.04166667
5   0.19444444   0.6666667   0.06779661  0.04166667
6   0.30555556   0.7916667   0.11864407  0.12500000
>
```