Fitting MLE and LM Weeks 5-6

Shadan Khan

2024-09-16

```
## Task 1: Define and Print a String
name <- "Shadan Khan"
unit_name <- "SIT741"
task_name <- "Fitting MLE And LM"
# Create and print the string
result_string <- paste("Name:", name, ", Unit Name:", unit_name, ", Task Name:", task_name)
print(result_string)
```

```
## [1] "Name: Shadan Khan , Unit Name: SIT741 , Task Name: Fitting MLE And LM"
```

###Question 2 Use rbinom() to generate 30 random data observations. Set the number of trials ###to 10 and choose your own value of p (the probability of success).

```
# First we Generate 30 random data observations using rbinom
set.seed(123) # for reproducibility
n <- 30
trials <- 10
p <- 0.6

# Generate random binomial data
random_data <- rbinom(n, size = trials, prob = p)
# Print the generated data
print(random_data)
```

```
##  [1] 7 5 6 4 4 9 6 4 6 6 3 6 5 6 8 4 7 9 7 3 4 5 5 2 5 5 6 6 7 8
```

###Question 3 Plot its histogram along with the theoretical distribution. ###- To use stat_function(), you can define a new function of a single variable using your trials ###and probability parameters, e.g.: ###dbinom_10_5 <- function(x) {dbinom(x,size = 10,prob = 0.5)} ###- You may also need to include the value of n as one of your stat_function() arguments, ###which gives the number of points to evaluate on. The default is 101, but you need your ###function to only be evaluated on whole numbers

```
# Load necessary library
library(ggplot2)

# Sample data: replace this with your actual data
data <- rbinom(100, size = 10, prob = 0.6)

# Plot histogram
ggplot(data.frame(x = data), aes(x = x)) +
```

```r
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = "green", color = "black") +

  # Add the theoretical binomial distribution
  stat_function(fun = function(x) dbinom(x, size = 10, prob = 0.6),
                color = "orange", size = 1) +

  labs(title = "Histogram with Theoretical Binomial Distribution",
       x = "Number of Successes",
       y = "Density") +
  theme_minimal()
```

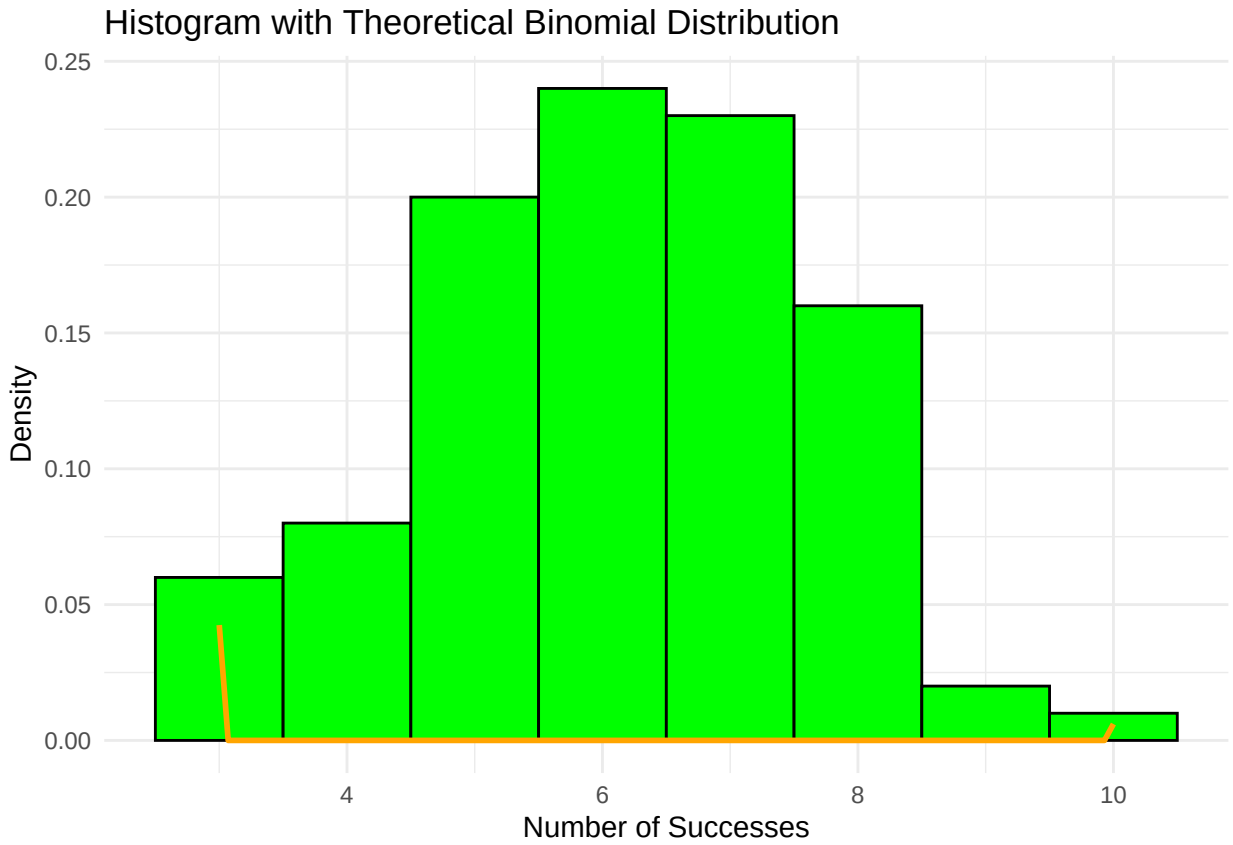
```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.

## Warning: The dot-dot notation ('..density..') was deprecated in ggplot2 3.4.0.
## i Please use 'after_stat(density)' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.070000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.140000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.210000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.280000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.350000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.420000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.490000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.560000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.630000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.700000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.770000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.840000

## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.910000
```

```
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 3.980000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.050000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.120000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.190000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.260000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.330000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.400000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.470000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.540000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.610000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.680000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.750000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.820000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.890000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 4.960000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.030000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.100000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.170000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.240000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.310000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.380000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.450000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.520000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.590000
```

```
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.660000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.730000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.800000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.870000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 5.940000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.010000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.080000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.150000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.220000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.290000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.360000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.430000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.500000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.570000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.640000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.710000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.780000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.850000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.920000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 6.990000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.060000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.130000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.200000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.270000
```

```
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.340000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.410000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.480000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.550000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.620000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.690000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.760000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.830000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.900000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 7.970000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.040000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.110000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.180000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.250000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.320000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.390000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.460000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.530000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.600000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.670000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.740000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.810000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.880000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 8.950000
```

```
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.020000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.090000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.160000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.230000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.300000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.370000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.440000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.510000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.580000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.650000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.720000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.790000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.860000
## Warning in dbinom(x, size = 10, prob = 0.6): non-integer x = 9.930000
```



###4. Use mle2() to estimate the parameters of a normal distribution that matches your data. ###-

Define the negative log-likelihood function ###- Choose appropriate initial parameters and then use mle2() to find your estimates. ###- Comment on how these compare to the theoretical values of the mean and standard ###deviation obtained using np and sqrt(npr). ###- Repeat the process with trials set to 100. How do the estimates for mean and sd ###compare now?

```r
# Load necessary libraries
library(bbmle)
```

```
## Loading required package: stats4
```

```r
# Function to calculate the negative log-likelihood for normal distribution
neg_log_likelihood <- function(mu, sigma, data) {
  -sum(dnorm(data, mean = mu, sd = sigma, log = TRUE))
}

# Generate data with prob = 0.6, size = 10, and trials = 100
set.seed(123)  # For reproducibility
data_100 <- rbinom(100, size = 10, prob = 0.6)

# Perform Maximum Likelihood Estimation
fit <- mle2(minuslogl = function(mu, sigma) neg_log_likelihood(mu, sigma, data_100),
            start = list(mu = mean(data_100), sigma = sd(data_100)),
            data = list(data = data_100))

# Display the results
summary(fit)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = function(mu, sigma) neg_log_likelihood(mu, sigma,
##     data_100), start = list(mu = mean(data_100), sigma = sd(data_100)),
##     data = list(data = data_100))
##
## Coefficients:
##       Estimate Std. Error z value     Pr(z)
## mu     6.02000    0.15361  39.190 < 2.2e-16 ***
## sigma  1.53610    0.10862  14.142 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 369.6369
```

```r
# Compare with theoretical values
mean_theoretical <- mean(data_100)
sd_theoretical <- sd(data_100)
cat("Estimated Mean:", coef(fit)["mu"], "\n")
```

```
## Estimated Mean: 6.02
```

```r
cat("Estimated SD:", coef(fit)["sigma"], "\n")
```

```
## Estimated SD: 1.5361
```

```r
cat("Theoretical Mean:", mean_theoretical, "\n")
```

```
## Theoretical Mean: 6.02
```

```r
cat("Theoretical SD:", sd_theoretical, "\n")
```

```
## Theoretical SD: 1.543838
```

```r
# Repeat with different sample size
data_10 <- rbinom(10, size = 10, prob = 0.6)

fit_10 <- mle2(minuslogl = function(mu, sigma) neg_log_likelihood(mu, sigma, data_10),
               start = list(mu = mean(data_10), sigma = sd(data_10)),
               data = list(data = data_10))

# Display the results for smaller sample size
summary(fit_10)
```

```
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = function(mu, sigma) neg_log_likelihood(mu, sigma,
##     data_10), start = list(mu = mean(data_10), sigma = sd(data_10)),
##     data = list(data = data_10))
##
## Coefficients:
##       Estimate Std. Error z value     Pr(z)
## mu     5.60000    0.45166 12.3986 < 2.2e-16 ***
## sigma  1.42829    0.31937  4.4721 7.744e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 35.50827
```

```r
# Compare with theoretical values
mean_theoretical_10 <- mean(data_10)
sd_theoretical_10 <- sd(data_10)
cat("Estimated Mean (10 trials):", coef(fit_10)["mu"], "\n")
```

```
## Estimated Mean (10 trials): 5.6
```

```r
cat("Estimated SD (10 trials):", coef(fit_10)["sigma"], "\n")
```

```
## Estimated SD (10 trials): 1.428286
```

```r
cat("Theoretical Mean (10 trials):", mean_theoretical_10, "\n")
```

```
## Theoretical Mean (10 trials): 5.6
```

```r
cat("Theoretical SD (10 trials):", sd_theoretical_10, "\n")
```

```
## Theoretical SD (10 trials): 1.505545
```

###Question 5 Read the "abalone" data set using: abalone <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/ abalone.data", sep = ",") - Information about the attributes can be found here: https://archive.ics.uci.edu/ml/machinelearning-databases/abalone/abalone.names - replace the variables names V1, V2, etc., with appropriate labels.

```r
# Load the dataset
abalone <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data", s

# Replace default column names (V1, V2, ...) with appropriate labels
colnames(abalone) <- c("Sex", "Length", "Diameter", "Height",
                       "WholeWeight", "ShuckedWeight", "VisceraWeight",
                       "ShellWeight", "Rings")

# Preview the dataset to verify the column names
head(abalone)
```

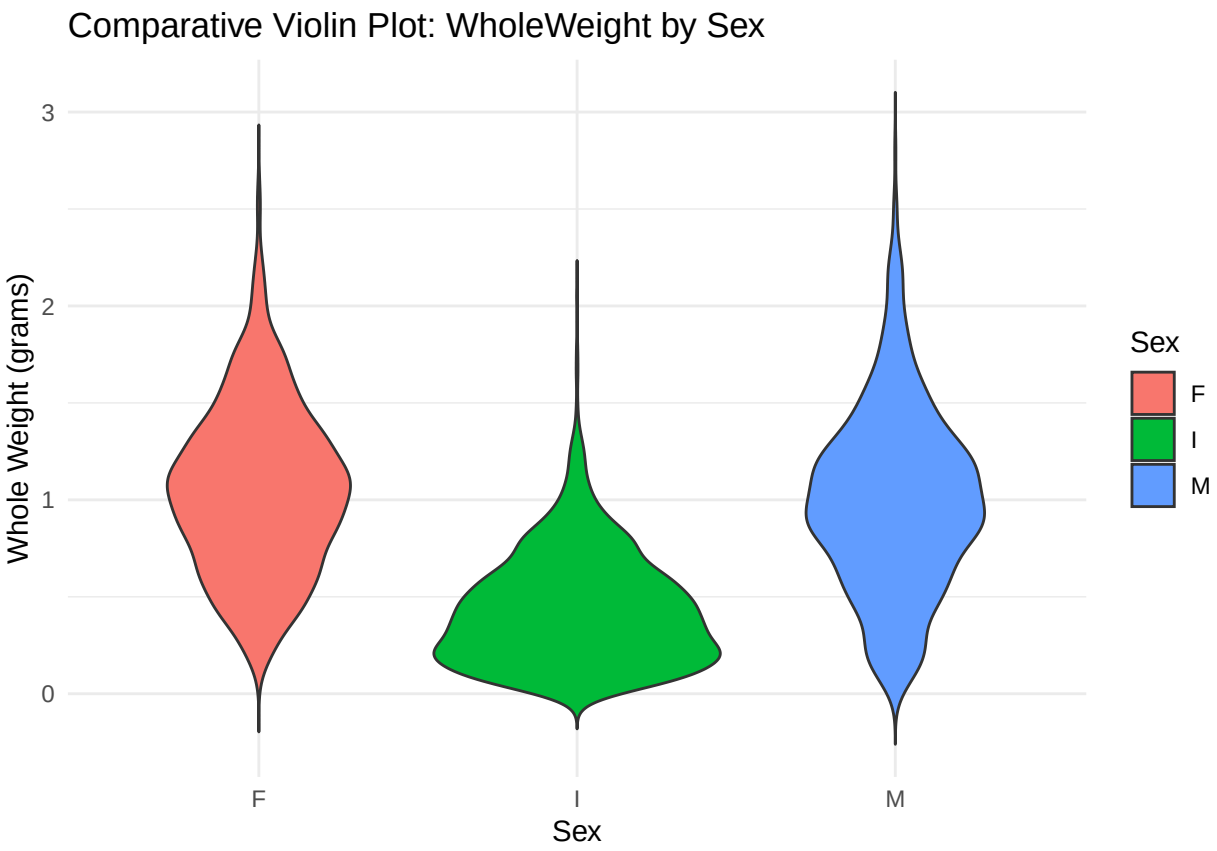
```
##   Sex Length Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1   M  0.455    0.365  0.095      0.5140        0.2245        0.1010
## 2   M  0.350    0.265  0.090      0.2255        0.0995        0.0485
## 3   F  0.530    0.420  0.135      0.6770        0.2565        0.1415
## 4   M  0.440    0.365  0.125      0.5160        0.2155        0.1140
## 5   I  0.330    0.255  0.080      0.2050        0.0895        0.0395
## 6   I  0.425    0.300  0.095      0.3515        0.1410        0.0775
##   ShellWeight Rings
## 1       0.150    15
## 2       0.070     7
## 3       0.210     9
## 4       0.155    10
## 5       0.055     7
## 6       0.120     8
```

###Question 6 Use ggplot to produce: ###- a multi-grid barplot showing the distribution of age (V9) for each of males, females and ###infants ###- comparative violin plot to compare distribution of males, females and infants (V1) with ###respect to one of variables V2-V8. ###- comparative boxplot to compare distribution for each age (V9) with respect to one of the ###variables V2-V8. You will need to ensure that age is a factor variable to use geom_boxplot(). ###- a scatterplot showing the relationship between one of the variables V2-V8 with age (V9), ###filtering the sex (V1) to one of either males, females or infants. Add a geom_smooth() ###trendline using method = "lm"

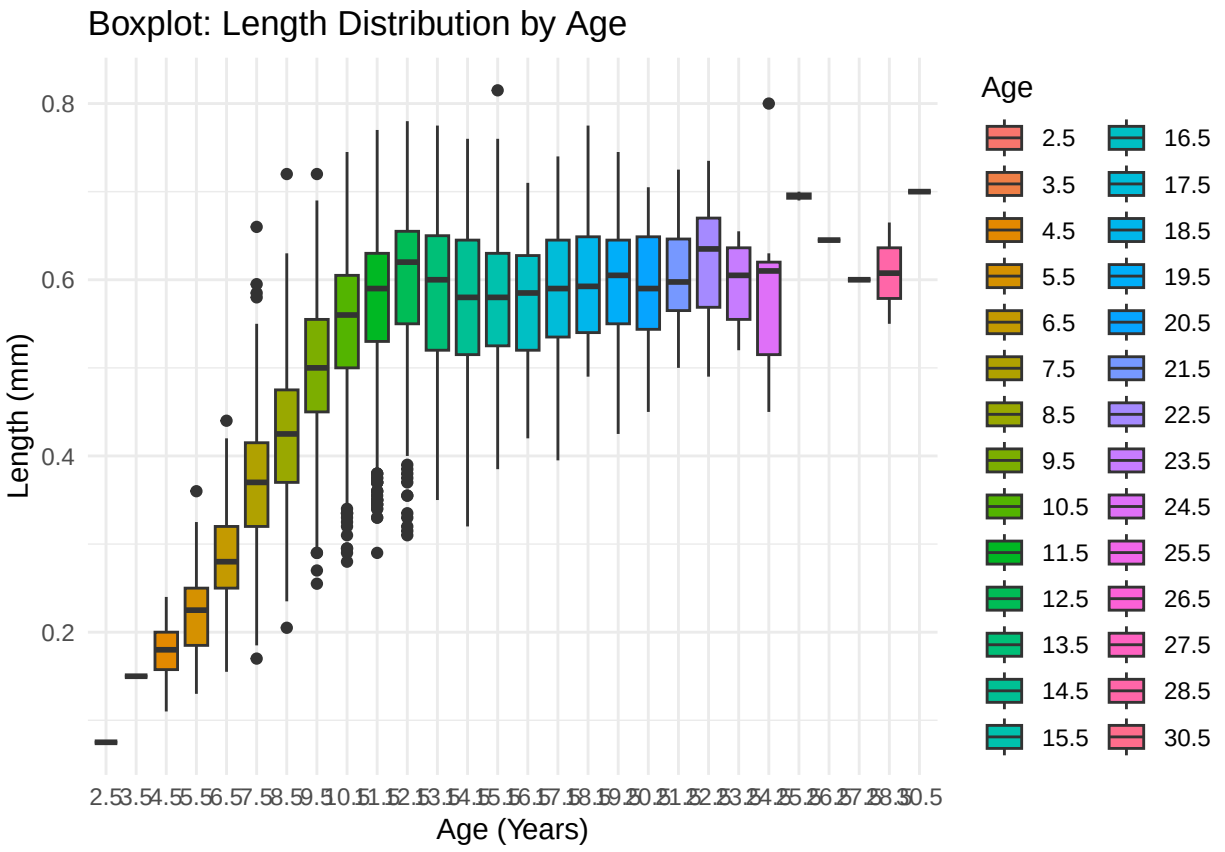
```r
library(ggplot2)
# Convert Rings to Age (Rings + 1.5 years is the age)
abalone$Age <- abalone$Rings + 1.5
```

```r
# Create a barplot showing the distribution of age by Sex
ggplot(abalone, aes(x = Age)) +
  geom_bar() +
  facet_grid(~ Sex) +
  theme_minimal() +
  labs(title = "Age Distribution by Sex (Male, Female, Infant)", x = "Age (Years)", y = "Count")
```


```r
# Violin plot to compare WholeWeight distribution by Sex
ggplot(abalone, aes(x = Sex, y = WholeWeight, fill = Sex)) +
  geom_violin(trim = FALSE) +
  theme_minimal() +
  labs(title = "Comparative Violin Plot: WholeWeight by Sex", x = "Sex", y = "Whole Weight (grams)")
```

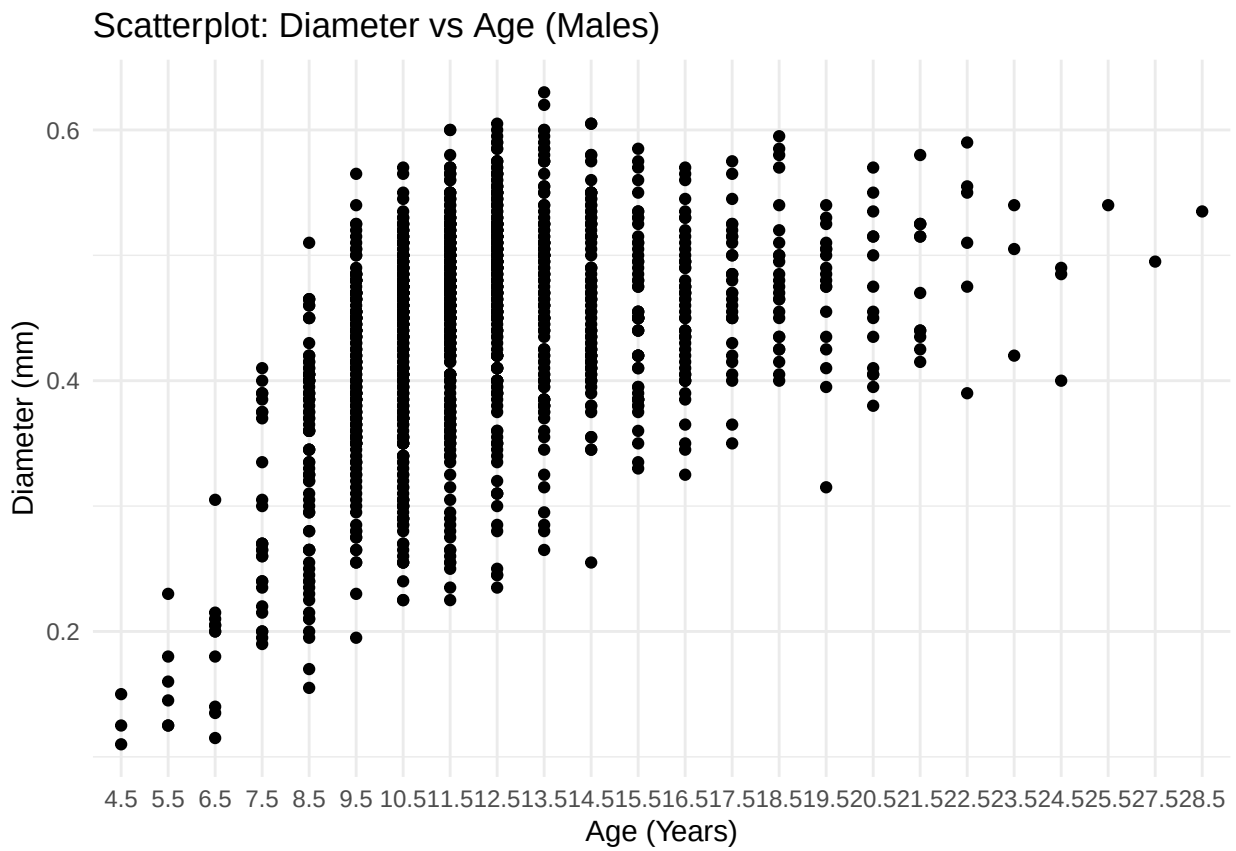



```r
# Ensure Age is a factor variable
abalone$Age <- as.factor(abalone$Age)

# Create a boxplot comparing Length (V2) distribution by Age
ggplot(abalone, aes(x = Age, y = Length, fill = Age)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Boxplot: Length Distribution by Age", x = "Age (Years)", y = "Length (mm)")
```




```
# Filter for males and create a scatterplot
ggplot(subset(abalone, Sex == "M"), aes(x = Age, y = Diameter)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  theme_minimal() +
  labs(title = "Scatterplot: Diameter vs Age (Males)", x = "Age (Years)", y = "Diameter (mm)")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



###Question 7 Build a linear model using lm() for the data in the scatterplot created in step 6. ###Print the coefficients.

```
# Build a linear model using lm() for males, predicting Diameter based on Age
model <- lm(Diameter ~ Age, data = subset(abalone, Sex == "M"))

# Print the coefficients of the linear model
summary(model)$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.12833333 0.03983290 3.2217927 1.301176e-03
## Age5.5      0.03250000 0.04878514 0.6661865 5.053940e-01
## Age6.5      0.06348485 0.04493757 1.4127344 1.579407e-01
## Age7.5      0.16166667 0.04198756 3.8503469 1.228986e-04
## Age8.5      0.21622917 0.04057289 5.3294003 1.135328e-07
## Age9.5      0.27916667 0.04017877 6.9481131 5.495030e-12
## Age10.5     0.30132494 0.04004725 7.5242364 9.082586e-14
## Age11.5     0.32666667 0.04003561 8.1594029 7.037129e-16
## Age12.5     0.34800000 0.04009757 8.6788303 1.020557e-17
## Age13.5     0.33933616 0.04033607 8.4127227 9.189039e-17
## Age14.5     0.33787546 0.04048416 8.3458684 1.580976e-16
## Age15.5     0.32657738 0.04088593 7.9875249 2.713984e-15
## Age16.5     0.32945513 0.04096581 8.0421972 1.771509e-15
## Age17.5     0.34450000 0.04177709 8.2461456 3.526514e-16
## Age18.5     0.35486667 0.04215517 8.4181045 8.794830e-17
## Age19.5     0.34138889 0.04302445 7.9347656 4.086228e-15
```

```
## Age20.5     0.33800000 0.04363475 7.7461195 1.730501e-14
## Age21.5     0.35583333 0.04453453 7.9900544 2.661095e-15
## Age22.5     0.38333333 0.04878514 7.8575847 7.403047e-15
## Age23.5     0.36000000 0.05633222 6.3906586 2.197701e-10
## Age24.5     0.33000000 0.05633222 5.8581037 5.741175e-09
## Age25.5     0.41166667 0.07966579 5.1674207 2.691658e-07
## Age27.5     0.36666667 0.07966579 4.6025610 4.524344e-06
## Age28.5     0.40666667 0.07966579 5.1046585 3.736587e-07
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

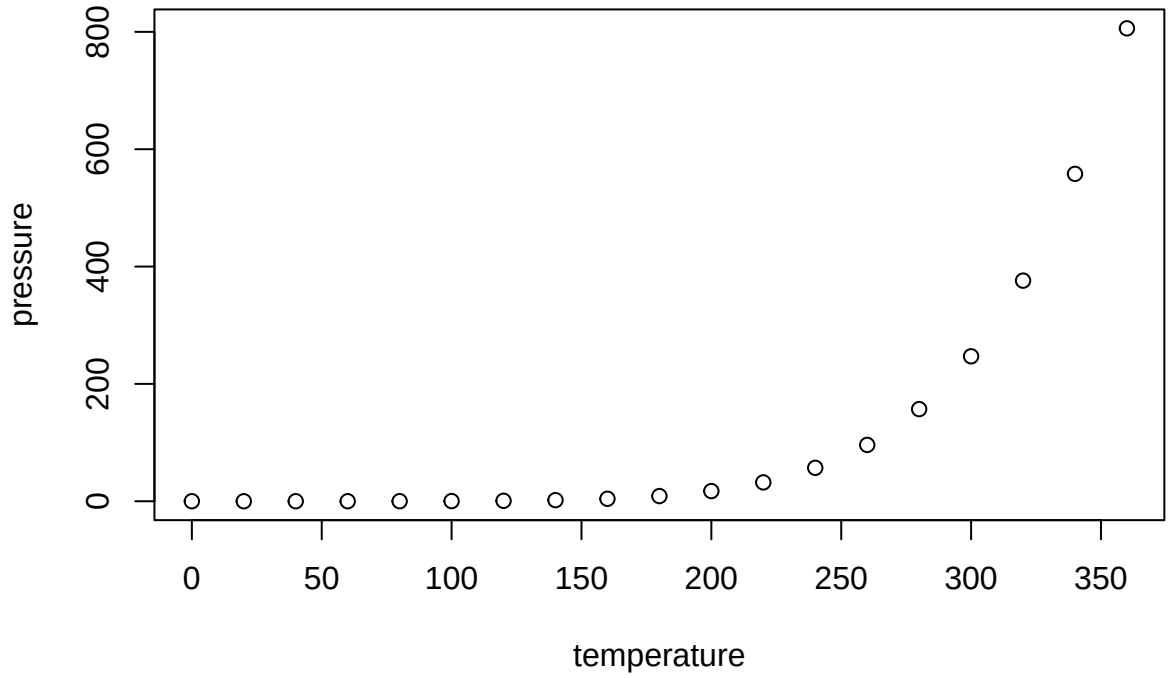
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.