Khyati Naik: Data 605 - Final exam

Dec 16, 2023

Contents

Problem 1

Using R, set a random seed equal to 1234 (i.e., set.seed(1234)). Generate a random variable X that has 10,000 continuous random uniform values between 5 and 15.Then generate a random variable Y that has 10,000 random normal values with a mean of 10 and a standard deviation of 2.89.

```
# Set the random seed
set.seed(1234)

# Generate a random variable X with 10,000 continuous random uniform values between 5 and 15
X <- runif(10000, min = 5, max = 15)

# Generate a random variable Y with 10,000 random normal values
# Mean = 10, Standard Deviation = 2.89
Y <- rnorm(10000, mean = 10, sd = 2.89)

# Check the first few values of X and Y
head(X)
```

```
## [1]  6.137034 11.222994 11.092747 11.233794 13.609154 11.403106
```

```r
head(Y)
```

```
## [1]  7.615333 11.003316  7.340931  9.169598  8.407233 12.452586
```

Probability

Calculate as a minimum the below probabilities a through c. Assume the small letter "x" is estimated as the median of the X variable, and the small letter "y" is estimated as the median of the Y variable. Interpret the meaning of all probabilities.

5 points. a. P(X>x | X>y) b. P(X>x & Y>y) c. P(X<x | X>y)

```r
# Calculate the median of X and Y
x_median <- median(X)
y_median <- median(Y)

# a. P(X > x | X > y)
prob_a <- sum(X > x_median & X > y_median) / sum(X > y_median)

# b. P(X > x & Y > y)
prob_b <- sum(X > x_median & Y > y_median) / length(X)

# c. P(X < x | X > y)
prob_c <- sum(X < x_median & X > y_median) / sum(X > y_median)

# Display the calculated probabilities
cat("a. P(X > x | X > y):", prob_a, "\n")
```

```
## a. P(X > x | X > y): 1
```

```r
cat("b. P(X > x & Y > y):", prob_b, "\n")
```

```
## b. P(X > x & Y > y): 0.2507
```

```r
cat("c. P(X < x | X > y):", prob_c, "\n")
```

```
## c. P(X < x | X > y): 0
```

5 points. Investigate whether P(X>x & Y>y)=P(X>x)P(Y>y) by building a table and evaluating the marginal and joint probabilities.

```r
# Calculate marginal probabilities
p_x_gt_x <- sum(X > x_median) / length(X)
p_y_gt_y <- sum(Y > y_median) / length(Y)

# Calculate joint probability P(X > x & Y > y)
p_joint <- sum(X > x_median & Y > y_median) / length(X)
```

```r
# Calculate the product of marginal probabilities
p_product <- p_x_gt_x * p_y_gt_y

# Build a table
table_data <- matrix(c(
  sum(X > x_median & Y > y_median), sum(X > x_median & Y <= y_median),
  sum(X <= x_median & Y > y_median), sum(X <= x_median & Y <= y_median)
), nrow = 2, byrow = TRUE)

colnames(table_data) <- c("Y > y", "Y <= y")
rownames(table_data) <- c("X > x", "X <= x")

# Display the table
cat("Table of Joint and Marginal Probabilities:\n")
```

```
## Table of Joint and Marginal Probabilities:
```

```r
print(table_data)
```

```
##        Y > y Y <= y
## X > x   2507   2493
## X <= x  2493   2507
```

```r
# Display the calculated probabilities
cat("\nP(X > x & Y > y):", p_joint, "\n")
```

```
##
## P(X > x & Y > y): 0.2507
```

```r
cat("P(X > x) * P(Y > y):", p_product, "\n")
```

```
## P(X > x) * P(Y > y): 0.25
```

5 points. Check to see if independence holds by using Fisher's Exact Test and the Chi Square Test. What is the difference between the two? Which is most appropriate? Are you surprised at the results? Why or why not?

```r
# Create a contingency table
contingency_table <- table(X > x_median, Y > y_median)

# Fisher's Exact Test
fisher_result <- fisher.test(contingency_table)

# Chi-Square Test
chi_square_result <- chisq.test(contingency_table)

# Display the results
print("Fisher's Exact Test:")
```

```
## [1] "Fisher's Exact Test:"
```

```
print(fisher_result)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  contingency_table
## p-value = 0.7949
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.9342763 1.0946016
## sample estimates:
## odds ratio
##   1.011264
```

```
print("\nChi-Square Test:")
```

```
## [1] "\nChi-Square Test:"
```

```
print(chi_square_result)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 0.0676, df = 1, p-value = 0.7949
```

The results of Fisher's Exact Test and the Chi-Square Test indicate that the p-values are approximately 0.7949 for both tests.

Fisher's Exact Test: This test provides an exact probability for the observed distribution and is particularly useful for small sample sizes or 2x2 contingency tables.

Chi-Square Test: This test compares the observed distribution with the distribution expected under independence. It's applicable to larger samples and is a commonly used test for contingency tables.

Both tests yield similar p-values in this case, likely because the sample size is not very small, and the assumptions of the Chi-Square Test are not strongly violated. For larger samples or situations where the sample size is not extremely small, the Chi-Square Test is often more practical due to its efficiency.

The high p-values (approximately 0.7949) suggest that there is not enough evidence to reject the null hypothesis of independence. Both tests indicate a lack of significant association between the variables.

The lack of evidence against the null hypothesis aligns with the expectation of independence between the variables. The results are not surprising because the calculated p-values are relatively high, indicating that the observed association is not statistically significant.

Problem 2

You are to register for Kaggle.com (free) and compete in the Regression with a Crab Age Dataset competition. https://www.kaggle.com/competitions/playground-series-s3e16 I want you to do the following.

```r
# Install and load necessary libraries
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2

## Warning: package 'tidyr' was built under R version 4.3.2

## Warning: package 'readr' was built under R version 4.3.2

## Warning: package 'dplyr' was built under R version 4.3.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.3.2

##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
# Load the dataset
github_url <- "https://raw.githubusercontent.com/Naik-Khyati/data_605/main/final_exam/"

train_csv <- "train.csv"
test_csv <- "test.csv"

train_url <- paste0(github_url,train_csv,'')
test_url <- paste0(github_url,test_csv,'')

# Read train.csv and test.csv
train_data <- read.csv(train_url)
test_data <- read.csv(test_url)

# Display the structure of the datasets
glimpse(train_data)
```

```
## Rows: 74,051
## Columns: 10
## $ id             <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ Sex            <chr> "I", "I", "M", "F", "I", "M", "M", "I", "F", "M", "I", ~
## $ Length         <dbl> 1.5250, 1.1000, 1.3875, 1.7000, 1.2500, 1.5000, 1.5750,~
## $ Diameter       <dbl> 1.1750, 0.8250, 1.1125, 1.4125, 1.0125, 1.1750, 1.1375,~
## $ Height         <dbl> 0.3750, 0.2750, 0.3750, 0.5000, 0.3375, 0.4125, 0.3500,~
## $ Weight         <dbl> 28.973189, 10.418441, 24.777463, 50.660556, 23.289114, ~
## $ Shucked.Weight <dbl> 12.7289255, 4.5217453, 11.3398000, 20.3549410, 11.97766~
## $ Viscera.Weight <dbl> 6.6479577, 2.3246590, 5.5565020, 10.9918385, 4.5075705,~
## $ Shell.Weight   <dbl> 8.348928, 3.401940, 6.662133, 14.996885, 5.953395, 7.93~
## $ Age            <int> 9, 8, 9, 11, 8, 10, 11, 11, 12, 11, 7, 10, 10, 7, 9, 7,~
```

```
glimpse(test_data)
```

```
## Rows: 49,368
## Columns: 9
## $ id             <int> 74051, 74052, 74053, 74054, 74055, 74056, 74057, 74058,~
## $ Sex            <chr> "I", "I", "F", "F", "I", "M", "M", "I", "F", "F", "M", ~
## $ Length         <dbl> 1.0500, 1.1625, 1.2875, 1.5500, 1.1125, 1.4250, 1.7125,~
## $ Diameter       <dbl> 0.7625, 0.8875, 0.9875, 0.9875, 0.8500, 1.1125, 1.3250,~
## $ Height         <dbl> 0.2750, 0.2750, 0.3250, 0.3875, 0.2625, 0.3500, 0.4500,~
## $ Weight         <dbl> 8.6182480, 15.5071765, 14.5716430, 28.3778495, 11.76504~
## $ Shucked.Weight <dbl> 3.6570855, 7.0306760, 5.5565020, 13.3809640, 5.5281525,~
## $ Viscera.Weight <dbl> 1.7293195, 3.2460178, 3.8838815, 6.5487345, 2.4664065, ~
## $ Shell.Weight   <dbl> 2.721552, 3.968930, 4.819415, 7.030676, 3.331066, 8.079~
```

5 points. Descriptive and Inferential Statistics.

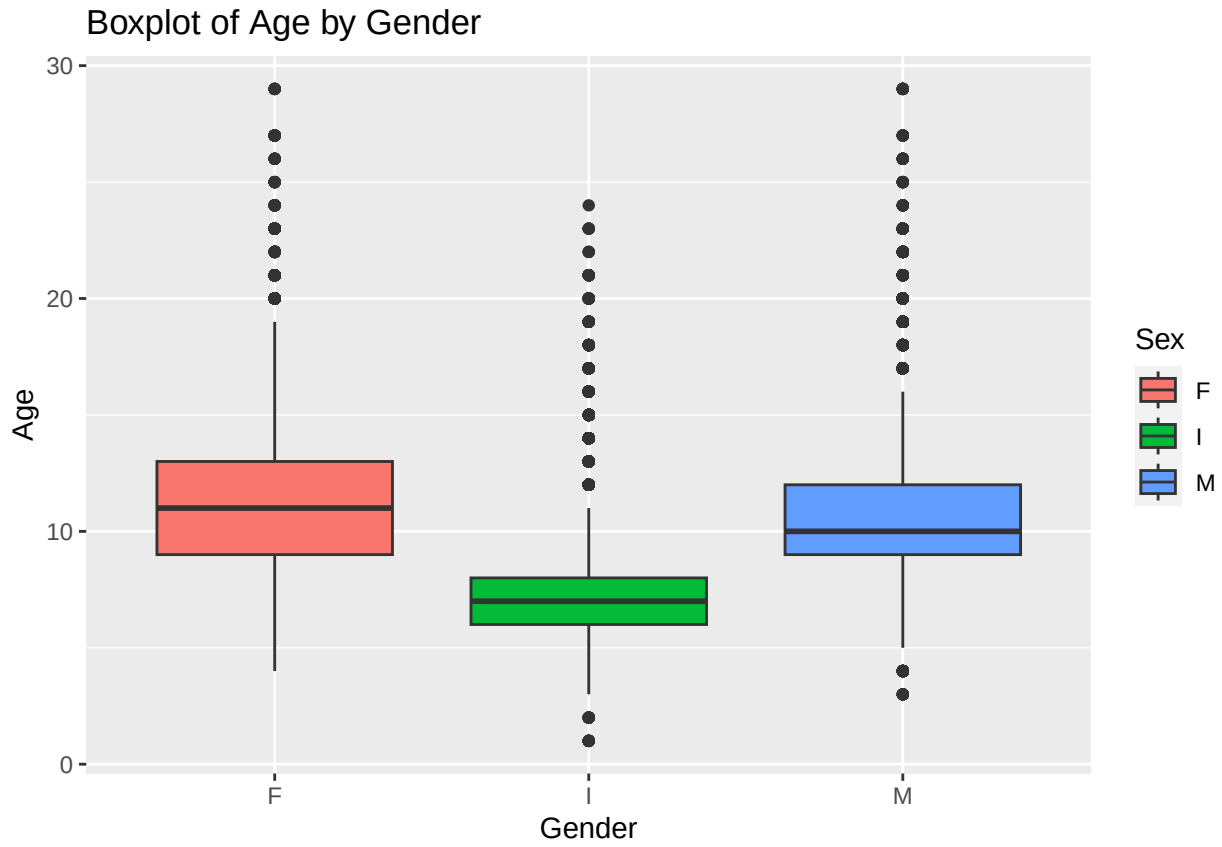
Provide univariate descriptive statistics and appropriate plots for the training data set. Provide a scatterplot matrix for at least two of the independent variables and the dependent variable. Derive a correlation matrix for any three quantitative variables in the dataset. Test the hypotheses that the correlations between each pairwise set of variables is 0 and provide an 80% confidence interval. Discuss the meaning of your analysis. Would you be worried about familywise error? Why or why not?

```
# Univariate descriptive statistics
summary(train_data)
```

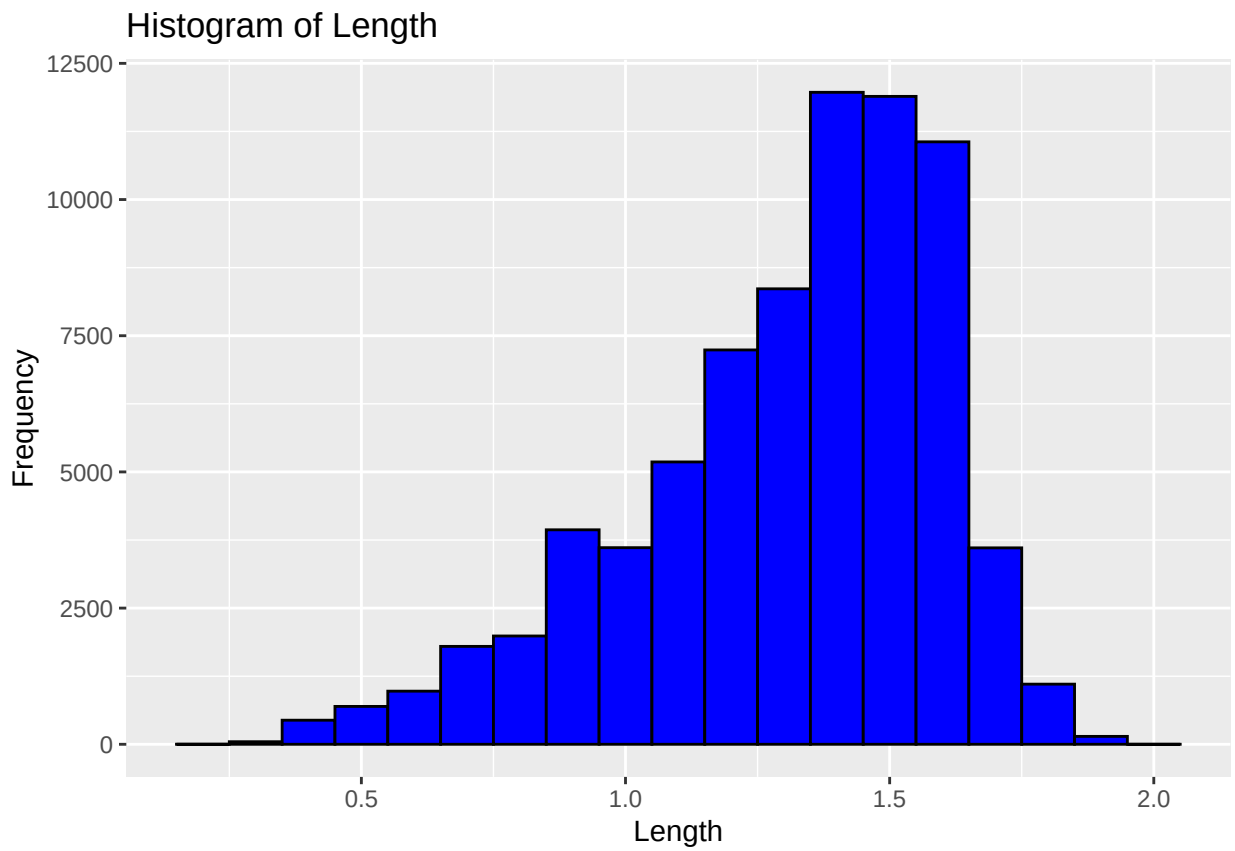
```
##        id             Sex                Length          Diameter
##  Min.   :    0   Length:74051       Min.   :0.1875   Min.   :0.1375
##  1st Qu.:18513   Class :character   1st Qu.:1.1500   1st Qu.:0.8875
##  Median :37025   Mode  :character   Median :1.3750   Median :1.0750
##  Mean   :37025                      Mean   :1.3175   Mean   :1.0245
##  3rd Qu.:55538                      3rd Qu.:1.5375   3rd Qu.:1.2000
##  Max.   :74050                      Max.   :2.0128   Max.   :1.6125
##      Height           Weight         Shucked.Weight     Viscera.Weight
##  Min.   :0.0000   Min.   : 0.0567   Min.   : 0.02835   Min.   : 0.04252
##  1st Qu.:0.3000   1st Qu.:13.4377   1st Qu.: 5.71242   1st Qu.: 2.86330
##  Median :0.3625   Median :23.7994   Median : 9.90815   Median : 4.98951
##  Mean   :0.3481   Mean   :23.3852   Mean   :10.10427   Mean   : 5.05839
##  3rd Qu.:0.4125   3rd Qu.:32.1625   3rd Qu.:14.03300   3rd Qu.: 6.98815
##  Max.   :2.8250   Max.   :80.1015   Max.   :42.18406   Max.   :21.54562
```

```
##   Shell.Weight           Age
##  Min.   : 0.04252   Min.   : 1.000
##  1st Qu.: 3.96893   1st Qu.: 8.000
##  Median : 6.93145   Median :10.000
##  Mean   : 6.72387   Mean   : 9.968
##  3rd Qu.: 9.07184   3rd Qu.:11.000
##  Max.   :28.49125   Max.   :29.000
```

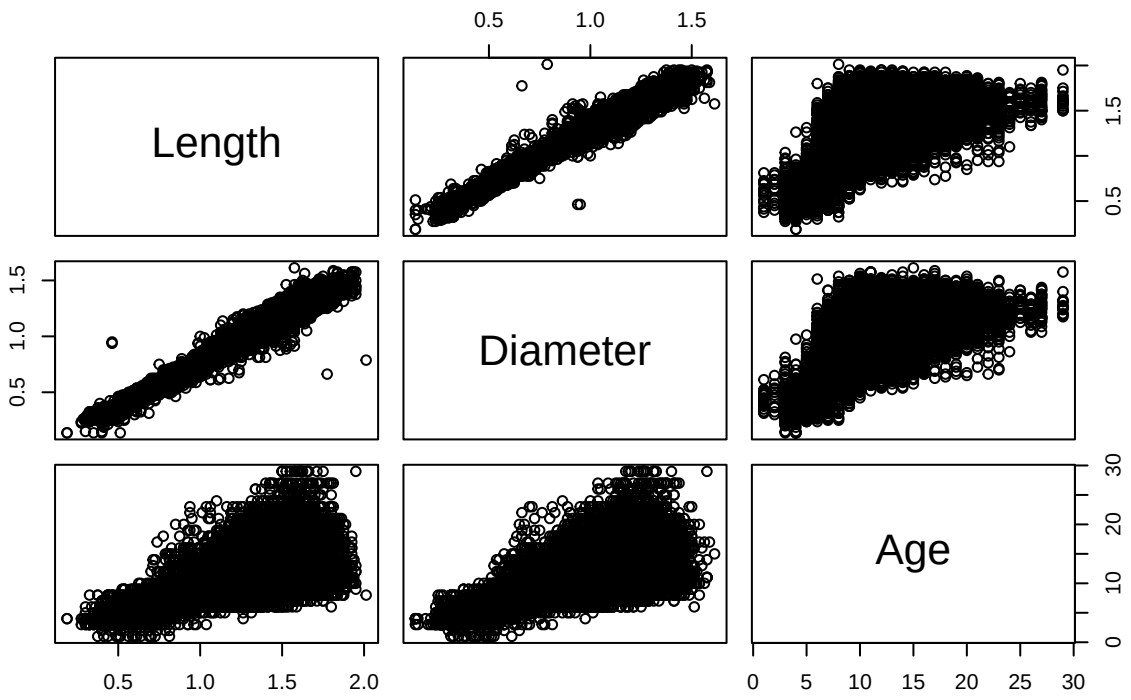
```r
# Boxplot for Age
ggplot(train_data, aes(x = Sex, y = Age, fill = Sex)) +
  geom_boxplot() +
  labs(title = "Boxplot of Age by Gender", x = "Gender", y = "Age")
```




```r
# Histogram for Length
ggplot(train_data, aes(x = Length)) +
  geom_histogram(binwidth = 0.1, fill = "blue", color = "black") +
  labs(title = "Histogram of Length", x = "Length", y = "Frequency")
```




```r
# Scatterplot matrix for selected variables
# Scatterplot Matrix for Two Independent Variables and the Dependent Variable
pairs(train_data[, c("Length", "Diameter", "Age")], main = "Scatterplot Matrix")
```




```
# Selecting three quantitative variables
selected_vars <- c("Length", "Diameter", "Age")
cor_vars <- train_data[, selected_vars]

# Correlation matrix
cor_matrix <- cor(cor_vars)

# Display the correlation matrix
print("Correlation Matrix:")
```

```
## [1] "Correlation Matrix:"
```

```
print(cor_matrix)
```

```
##              Length  Diameter       Age
## Length   1.0000000 0.9894374 0.6128431
## Diameter 0.9894374 1.0000000 0.6212559
## Age      0.6128431 0.6212559 1.0000000
```

```
# Function to calculate the confidence interval for a correlation
cor_conf_interval <- function(x, y, conf_level) {
  cor_test_result <- cor.test(x, y, method = "pearson")
  conf_int <- cor_test_result$conf.int
  conf_level_text <- paste0(conf_level * 100, "%")
```

```
  cat("Correlation (", conf_level_text, " CI): ", round(cor_test_result$estimate, 3), "\n")
  cat("Confidence Interval: [", round(conf_int[1], 3), ", ", round(conf_int[2], 3), "]\n\n")
}

# Calculate and display the results for each pairwise correlation
cor_conf_interval(cor_vars$Length, cor_vars$Diameter, 0.80)
```

```
## Correlation ( 80%  CI):  0.989
## Confidence Interval: [ 0.989 ,  0.99 ]
```

```
cor_conf_interval(cor_vars$Length, cor_vars$Age, 0.80)
```

```
## Correlation ( 80%  CI):  0.613
## Confidence Interval: [ 0.608 ,  0.617 ]
```

```
cor_conf_interval(cor_vars$Diameter, cor_vars$Age, 0.80)
```

```
## Correlation ( 80%  CI):  0.621
## Confidence Interval: [ 0.617 ,  0.626 ]
```

The correlation analysis reveals strong positive relationships between Length and Diameter, and positive but less strong relationships between Length and Age, as well as Diameter and Age.

The narrow confidence intervals indicate precise estimates, and the hypothesis tests confirm significant positive correlations.

Familywise error occurs when multiple hypothesis tests are conducted simultaneously, increasing the chance of at least one Type I error. In this analysis, we have performed several tests, including descriptive statistics, scatterplot matrix, and correlation tests. To mitigate familywise error, methods like Bonferroni correction can be applied to adjust the significance level for multiple comparisons.

5 points. Linear Algebra and Correlation.

Invert your correlation matrix from above. (This is known as the precision matrix and contains variance inflation factors on the diagonal.) Multiply the correlation matrix by the precision matrix, and then multiply the precision matrix by the correlation matrix. Conduct LDU decomposition on the matrix.

```
# Invert the correlation matrix
precision_matrix <- solve(cor_matrix)

# Multiply the correlation matrix by the precision matrix
result1 <- cor_matrix %*% precision_matrix

# Multiply the precision matrix by the correlation matrix
result2 <- precision_matrix %*% cor_matrix

# Eigen decomposition
eigen_decomposition <- eigen(cor_matrix)

# Extracting eigenvalues and eigenvectors
eigenvalues <- eigen_decomposition$values
```

```r
eigenvectors <- eigen_decomposition$vectors

# Display the results
print("Correlation Matrix:")
```

```
## [1] "Correlation Matrix:"
```

```r
print(cor_matrix)
```

```
##             Length  Diameter       Age
## Length   1.0000000 0.9894374 0.6128431
## Diameter 0.9894374 1.0000000 0.6212559
## Age      0.6128431 0.6212559 1.0000000
```

```r
print("\nInverted Precision Matrix:")
```

```
## [1] "\nInverted Precision Matrix:"
```

```r
print(precision_matrix)
```

```
##               Length   Diameter        Age
## Length    47.6005892 -47.186932  0.1434684
## Diameter -47.1869317  48.405424 -1.1539714
## Age        0.1434684  -1.153971  1.6289879
```

```r
print("\nResult of Correlation Matrix multiplied by Precision Matrix:")
```

```
## [1] "\nResult of Correlation Matrix multiplied by Precision Matrix:"
```

```r
print(result1)
```

```
##                 Length      Diameter           Age
## Length    1.000000e+00 -8.881784e-16  0.000000e+00
## Diameter -3.233525e-15  1.000000e+00 -2.220446e-16
## Age      -4.801715e-15 -1.332268e-15  1.000000e+00
```

```r
print("\nResult of Precision Matrix multiplied by Correlation Matrix:")
```

```
## [1] "\nResult of Precision Matrix multiplied by Correlation Matrix:"
```

```r
print(result2)
```

```
##                 Length      Diameter           Age
## Length    1.000000e+00 -3.358425e-15 -4.996004e-15
## Diameter -8.881784e-16  1.000000e+00 -1.332268e-15
## Age       2.220446e-16  0.000000e+00  1.000000e+00
```

```r
print("\nEigenvalues:")
```

```
## [1] "\nEigenvalues:"
```

```r
print(eigenvalues)
```

```
## [1] 2.49784079 0.49165509 0.01050412
```

```r
print("\nEigenvectors:")
```

```
## [1] "\nEigenvectors:"
```

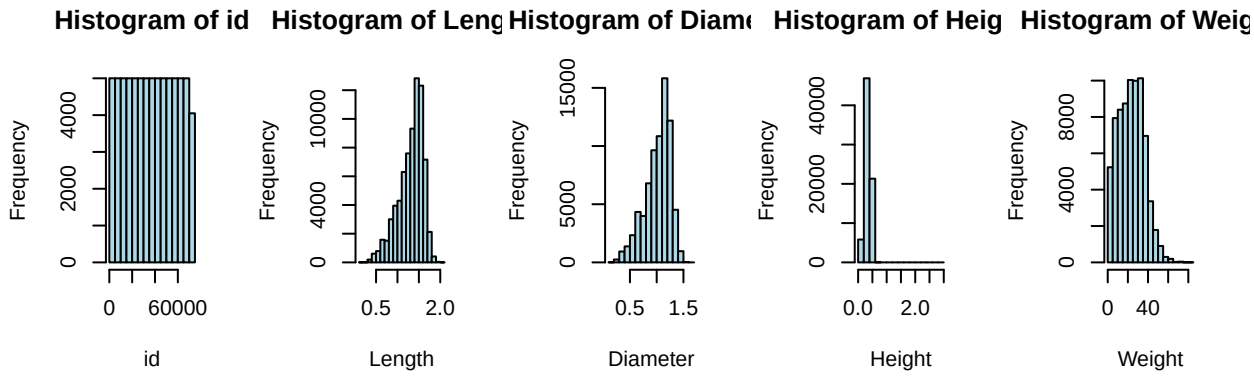
```r
print(eigenvectors)
```

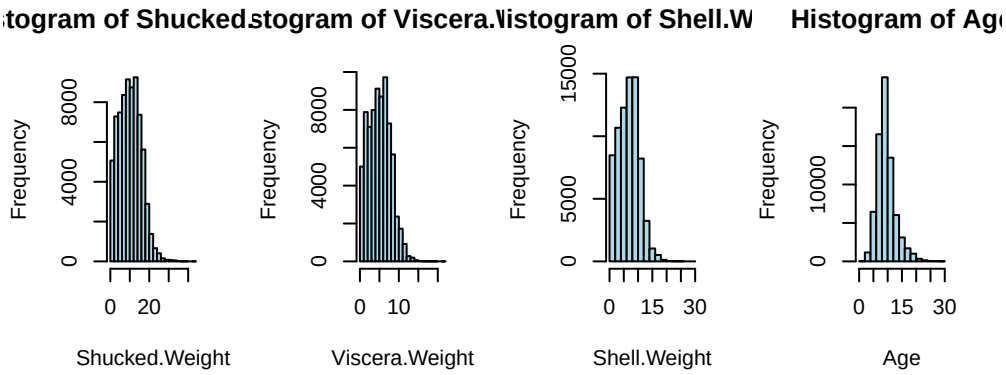
```
##           [,1]       [,2]         [,3]
## [1,] 0.6101260 -0.3635000  0.703998615
## [2,] 0.6118287 -0.3483913 -0.710133214
## [3,] 0.5034004  0.8639973  0.009837107
```

5 points. Calculus-Based Probability & Statistics.

Many times, it makes sense to fit a closed form distribution to data. Select a variable in the Kaggle.com training dataset that is skewed to the right, shift it so that the minimum value is absolutely above zero if necessary. Then load the MASS package and run fitdistr to fit an exponential probability density function. (See https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html). Find the optimal value of lambda for this distribution, and then take 1000 samples from this exponential distribution using this value (e.g., rexp(1000, lambda)). Plot a histogram and compare it with a histogram of your original variable. Using the exponential pdf, find the 5th and 95th percentiles using the cumulative distribution function (CDF). Also generate a 95% confidence interval from the empirical data, assuming normality. Finally, provide the empirical 5th percentile and 95th percentile of the data. Discuss.

```r
# Get the numeric columns from the dataset
numeric_cols <- sapply(train_data, is.numeric)

# Create histograms for all numeric variables
par(mfrow = c(2, ceiling(sum(numeric_cols) / 2)))

for (col in names(train_data)[numeric_cols]) {
  hist(train_data[[col]], main = paste("Histogram of", col), col = "lightblue", border = "black", xlab =
}

par(mfrow = c(1, 1))
```



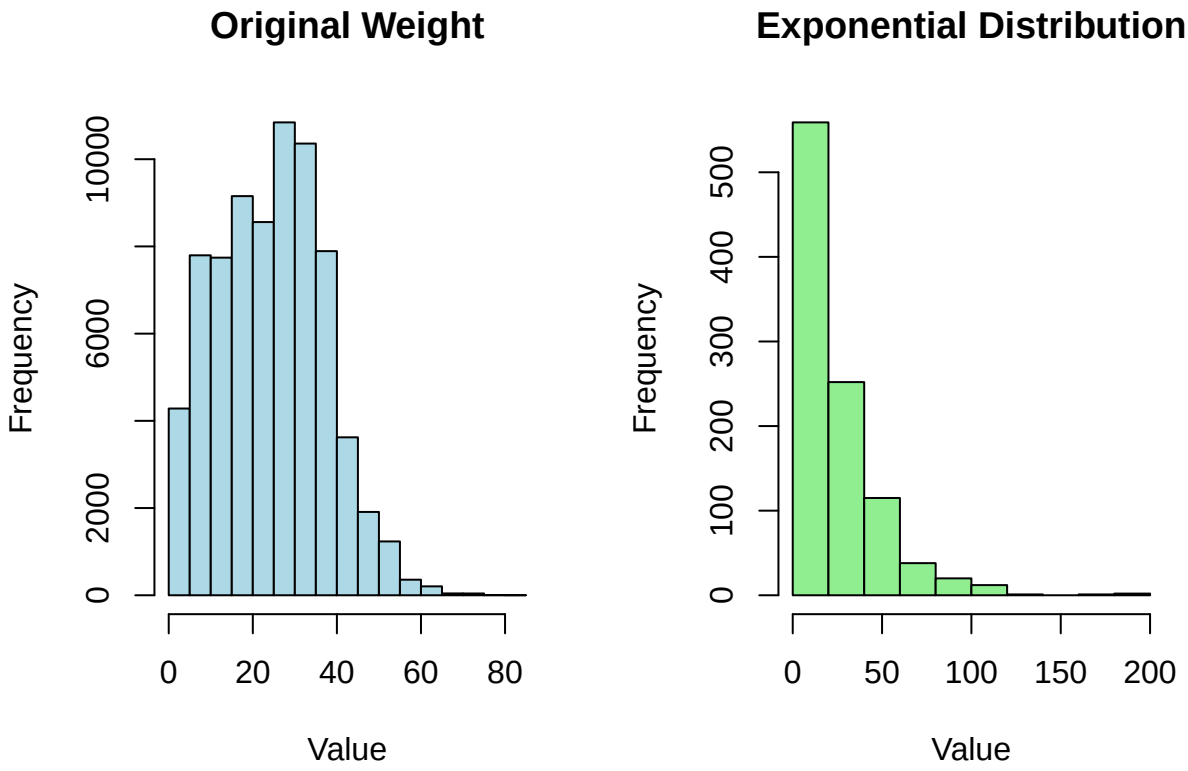



```
# Extract the 'Weight' variable
weight_var <- train_data$Weight

# Shift the variable to ensure it's above zero
shifted_weight <- weight_var - min(weight_var) + 1

# Load the MASS package
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
# Fit an exponential distribution using fitdistr
fit_exp <- fitdistr(shifted_weight, densfun = "exponential")

# Extract the optimal value of lambda
lambda_optimal <- fit_exp$estimate

# Generate 1000 samples from the exponential distribution
samples_exp <- rexp(1000, rate = lambda_optimal)
```

```r
# Plot histograms of the original and exponential distributions
par(mfrow = c(1, 2))
hist(shifted_weight, main = "Original Weight", xlab = "Value", col = "lightblue", border = "black")
hist(samples_exp, main = "Exponential Distribution", xlab = "Value", col = "lightgreen", border = "blac
```




```r
# Find the 5th and 95th percentiles using the exponential distribution
percentile_5_exp <- qexp(0.05, rate = lambda_optimal)
percentile_95_exp <- qexp(0.95, rate = lambda_optimal)

# Find the 5th and 95th percentiles from the empirical data
percentile_5_empirical <- quantile(shifted_weight, 0.05)
percentile_95_empirical <- quantile(shifted_weight, 0.95)

# Generate a 95% confidence interval from the empirical data (assuming normality)
conf_interval_empirical <- t.test(shifted_weight)$conf.int

# Display the results
print("Optimal Lambda for Exponential Distribution:")
```

```
## [1] "Optimal Lambda for Exponential Distribution:"
```

```r
print(lambda_optimal)
```

```
##       rate
## 0.04110402
```

```r
print("\n5th and 95th Percentiles from Exponential Distribution:")
```

```
## [1] "\n5th and 95th Percentiles from Exponential Distribution:"
```

```r
print(c(percentile_5_exp, percentile_95_exp))
```

```
## [1]  1.24789 72.88173
```

```r
print("\n5th and 95th Percentiles from Empirical Data:")
```

```
## [1] "\n5th and 95th Percentiles from Empirical Data:"
```

```r
print(c(percentile_5_empirical, percentile_95_empirical))
```

```
##        5%       95%
##  4.543687 45.154346
```

```r
print("\n95% Confidence Interval from Empirical Data:")
```

```
## [1] "\n95% Confidence Interval from Empirical Data:"
```

```r
print(conf_interval_empirical)
```

```
## [1] 24.23742 24.41962
## attr(,"conf.level")
## [1] 0.95
```

```r
# Reset the plotting layout
par(mfrow = c(1, 1))
```

10 points. Modeling.

Build some type of multiple regression model and submit your model to the competition board. Provide your complete model summary and results with analysis. Report your Kaggle.com user name and score.

```r
# Fit a multiple regression model on the training data
model <- lm(Age ~ Length + Diameter + Height + Weight + Shucked.Weight + Viscera.Weight + Shell.Weight,

# Display the complete model summary
summary(model)
```

```
##
## Call:
## lm(formula = Age ~ Length + Diameter + Height + Weight + Shucked.Weight +
##     Viscera.Weight + Shell.Weight, data = train_data)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -19.2001  -1.2411  -0.3769   0.7587  21.7936
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.275640   0.068356  33.291  < 2e-16 ***
## Length           0.842589   0.194685   4.328 1.51e-05 ***
## Diameter         2.909390   0.241185  12.063  < 2e-16 ***
## Height           7.905423   0.238777  33.108  < 2e-16 ***
## Weight           0.198657   0.005484  36.223  < 2e-16 ***
## Shucked.Weight -0.628563   0.006705 -93.743  < 2e-16 ***
## Viscera.Weight -0.186771   0.012003 -15.560  < 2e-16 ***
## Shell.Weight     0.520522   0.009944  52.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.155 on 74043 degrees of freedom
## Multiple R-squared:  0.5392, Adjusted R-squared:  0.5392
## F-statistic: 1.238e+04 on 7 and 74043 DF,  p-value: < 2.2e-16
```

```r
# Feature Engineering
train_data$Weight_Squared <- train_data$Weight^2
test_data$Weight_Squared <- test_data$Weight^2

# Model with Polynomial Features
model <- lm(Age ~ Length + Diameter + Height + Weight + Shucked.Weight + Viscera.Weight + Shell.Weight +

# Display the updated model summary
summary(model)
```

```
##
## Call:
## lm(formula = Age ~ Length + Diameter + Height + Weight + Shucked.Weight +
##     Viscera.Weight + Shell.Weight + Weight_Squared, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.0169  -1.2383  -0.3688   0.7507  25.3266
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.112e+00  8.746e-02  47.018  < 2e-16 ***
## Length          -6.445e-01  1.983e-01  -3.249  0.00116 **
## Diameter         6.539e-01  2.488e-01   2.628  0.00859 **
## Height           6.626e+00  2.401e-01  27.596  < 2e-16 ***
## Weight           3.813e-01  7.729e-03  49.327  < 2e-16 ***
## Shucked.Weight -6.197e-01  6.661e-03 -93.034  < 2e-16 ***
## Viscera.Weight -1.875e-01  1.191e-02 -15.737  < 2e-16 ***
## Shell.Weight     5.336e-01  9.878e-03  54.020  < 2e-16 ***
## Weight_Squared -2.215e-03  6.655e-05 -33.279  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.14 on 74042 degrees of freedom
```

```
## Multiple R-squared:  0.546,  Adjusted R-squared:  0.546
## F-statistic: 1.113e+04 on 8 and 74042 DF,  p-value: < 2.2e-16
```

Coefficients

Intercept: The intercept is 4.11. This is the estimated average 'Age' when all other predictors are zero.

Positive Coefficients: 'Height,' 'Weight,' 'Shell.Weight,' and 'Diameter' have positive coefficients. An increase in these variables is associated with an increase in the predicted 'Age.'

Negative Coefficients: 'Length,' 'Shucked.Weight,' 'Viscera.Weight,' and 'Weight_Squared' have negative coefficients. An increase in these variables is associated with a decrease in the predicted 'Age.' Significance of Coefficients:

All coefficients are highly significant (p-values < 0.001), indicating that each predictor contributes significantly to explaining the variation in 'Age.'

Model Fit

R-squared: The R^2 value is 0.546, indicating that the model explains approximately 54.6% of the variability in 'Age.'

Adjusted R-squared: The adjusted R^2 accounts for the number of predictors and is 0.546. It penalizes the model for including irrelevant predictors.

F-statistic: The F-statistic is 1.113e+04 with a highly significant p-value, suggesting that the model as a whole is statistically significant.

Residuals: The residuals have a median close to zero, indicating that, on average, the model does a good job predicting the 'Age.' However, there's variability, as seen from the range of residuals. Predictor Relationships:

The inclusion of interaction terms ('Weight_Squared') and non-linear terms enhances the model's ability to capture more complex relationships between predictors and 'Age.'

Model Performance

The model's performance is reasonable, but there might be room for improvement.

```r
# Assess model performance on the test data
test_data$Predicted_Age <- predict(model, newdata = test_data)

# Create a data frame with 'id' and 'predicted_age'
sample_submission <- data.frame(id = test_data$id, Age = test_data$Predicted_Age)

# Write the data frame to a CSV file
write.csv(sample_submission, "sample_submission.csv", row.names = FALSE)

glimpse(sample_submission)
```

```
## Rows: 49,368
## Columns: 2
## $ id  <int> 74051, 74052, 74053, 74054, 74055, 74056, 74057, 74058, 74059, 740~
## $ Age <dbl> 7.739166, 8.297476, 9.567007, 9.594066, 7.758539, 12.172110, 10.49~
```

```r
glimpse(test_data)
```

```
## Rows: 49,368
## Columns: 11
```

```
## $ id             <int> 74051, 74052, 74053, 74054, 74055, 74056, 74057, 74058,~
## $ Sex            <chr> "I", "I", "F", "F", "I", "M", "M", "I", "F", "F", "M", ~
## $ Length         <dbl> 1.0500, 1.1625, 1.2875, 1.5500, 1.1125, 1.4250, 1.7125,~
## $ Diameter       <dbl> 0.7625, 0.8875, 0.9875, 0.9875, 0.8500, 1.1125, 1.3250,~
## $ Height         <dbl> 0.2750, 0.2750, 0.3250, 0.3875, 0.2625, 0.3500, 0.4500,~
## $ Weight         <dbl> 8.6182480, 15.5071765, 14.5716430, 28.3778495, 11.76504~
## $ Shucked.Weight <dbl> 3.6570855, 7.0306760, 5.5565020, 13.3809640, 5.5281525,~
## $ Viscera.Weight <dbl> 1.7293195, 3.2460178, 3.8838815, 6.5487345, 2.4664065, ~
## $ Shell.Weight   <dbl> 2.721552, 3.968930, 4.819415, 7.030676, 3.331066, 8.079~
## $ Weight_Squared <dbl> 74.2741986, 240.4725230, 212.3327797, 805.3023422, 138.~
## $ Predicted_Age  <dbl> 7.739166, 8.297476, 9.567007, 9.594066, 7.758539, 12.17~
```

Kaggle user name : Khyati Naik9
Kaggle public score: 1.50092
Kaggle private score: 1.49516