MATH 3070 Lab Project 3

Prachi Aswani

September 12, 2024

Contents

Remember: I expect to see commentary either in the text, in the code with comments created using **#**, *or (preferably) both!* ***Failing to do so may result in lost points!***

Problem 1

The **faithful** *(Package* ***datasets*** *is built in Base R) dataset records the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park. Find the mean waiting time, median waiting time, longest waiting time and shortest waiting time in minutes.*

```
# Load the dataset faithful, which is built into Base R
data("faithful")

# Extract the waiting time column
waiting_times <- faithful$waiting

# Calculate the mean waiting time
mean_waiting <- mean(waiting_times)
mean_waiting # Outputs the mean waiting time
```

```
## [1] 70.89706
```

```
# Calculate the median waiting time
median_waiting <- median(waiting_times)
median_waiting # Outputs the median waiting time
```

```
## [1] 76
```

```
# Find the longest waiting time
longest_waiting <- max(waiting_times)
longest_waiting # Outputs the maximum (longest) waiting time
```

```
## [1] 96
```

```
# Find the shortest waiting time
shortest_waiting <- min(waiting_times)
shortest_waiting # Outputs the minimum (shortest) waiting time
```

```
## [1] 43
```

```
# OR
summary(waiting_times)
```

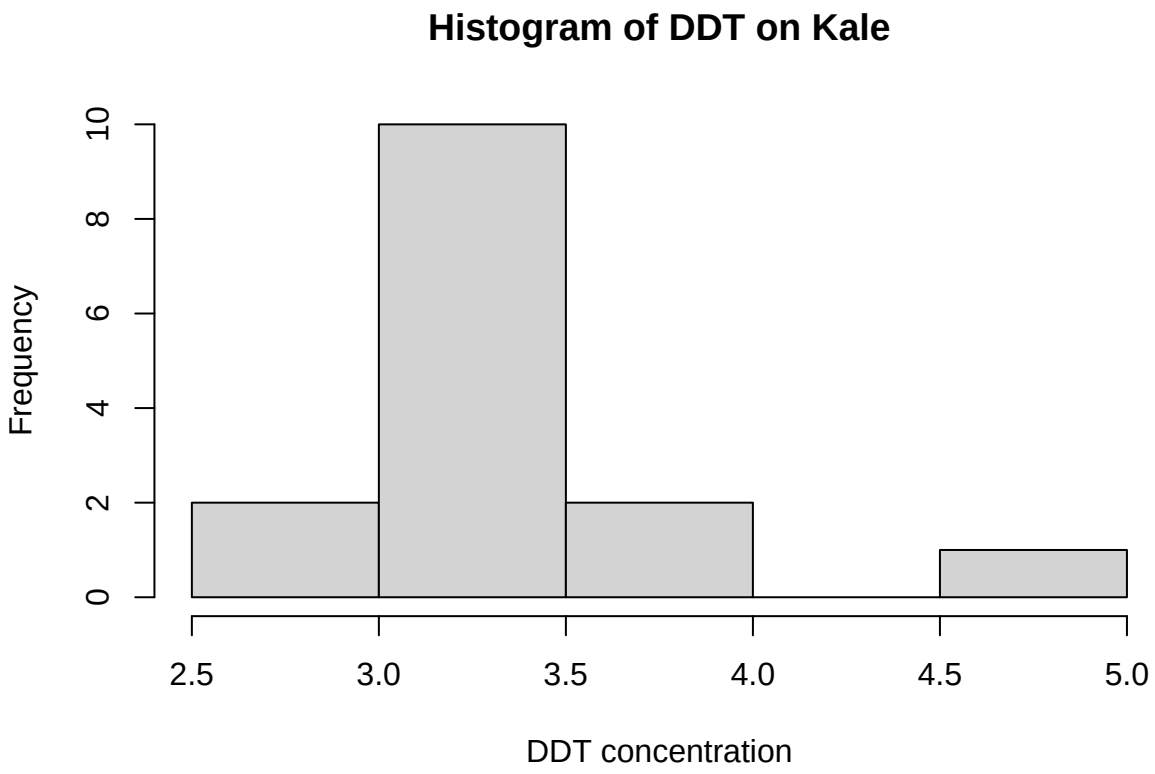
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    43.0    58.0    76.0    70.9    82.0    96.0
```

Problem 2 (Verzani problem 2.34)

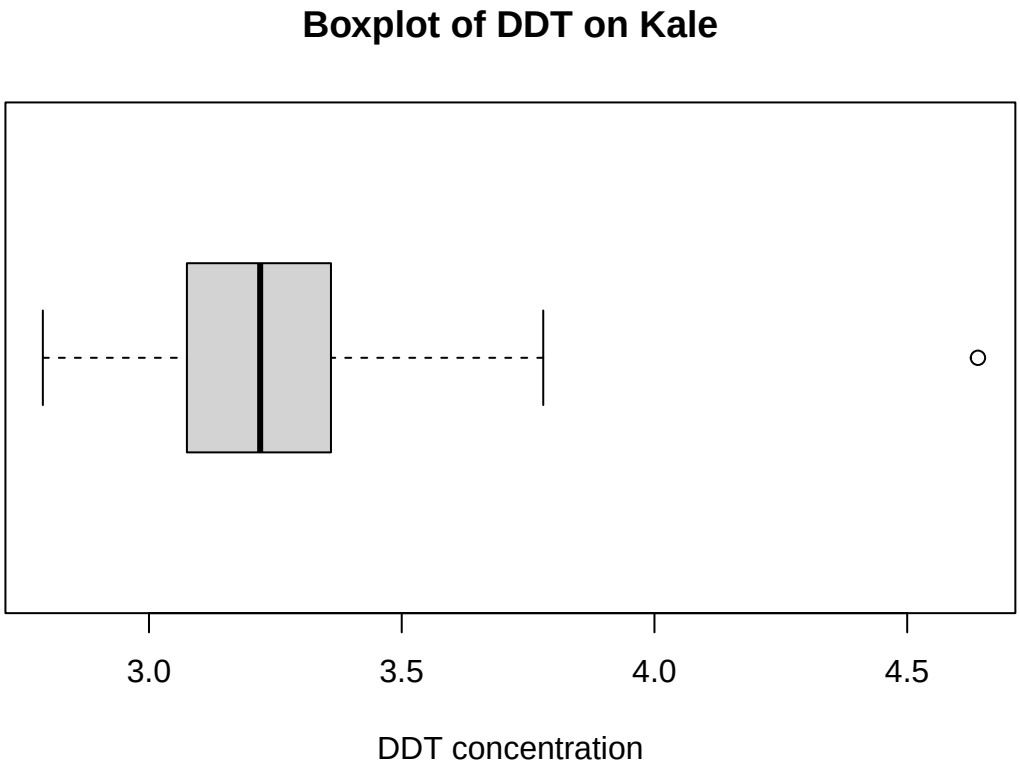
The data set `DDT` *(**MASS**) contains independent measurements of the pesticide DDT on kale. Make a histogram and a boxplot of the data. From these, estimate the mean and standard deviation. Check your answers with the appropriate functions.*

```
# Load the dataset from the MASS package
library(MASS)
data("DDT")

# Plot the histogram of DDT measurements
hist(DDT, main = "Histogram of DDT on Kale", xlab="DDT concentration", ylab = "Frequency")
```




```r
# Plot the boxplot of DDT measurements
boxplot(DDT, main="Boxplot of DDT on Kale", xlab="DDT concentration", horizontal = TRUE)
```




```
# Estimate from the plots: mean and standard deviation

# Mean is between 3.0 and 3.5
# Standard Deviation is 0.5

# Now calculate mean and standard deviation using appropriate functions
mean_ddt <- mean(DDT)
sd_ddt <- sd(DDT)

mean_ddt  # Outputs the calculated mean
```

```
## [1] 3.328
```

```
sd_ddt    # Outputs the calculated standard deviation
```

```
## [1] 0.4371531
```

Problem 3 (Verzani problem 2.32 modified)

Write a function 'density_compare()' that fits a density estimate to a given data set and plots that estimate along with the appropriate histogram of the given data set. Try your function with the data set `pi2000` *(**UsingR**). Why might you want to add an argument like* `breaks = 0:10-.5` *to* `hist()`*? (Hint: read the documentation of* `hist()` *to see what setting this argument does). Feel free to add other parameters to your plot methods to see how they can be changed.*

```r
# Load the pi2000 dataset from UsingR package
library(UsingR)
```

```
## Warning: package 'UsingR' was built under R version 4.3.3
```

```
## Loading required package: HistData
```

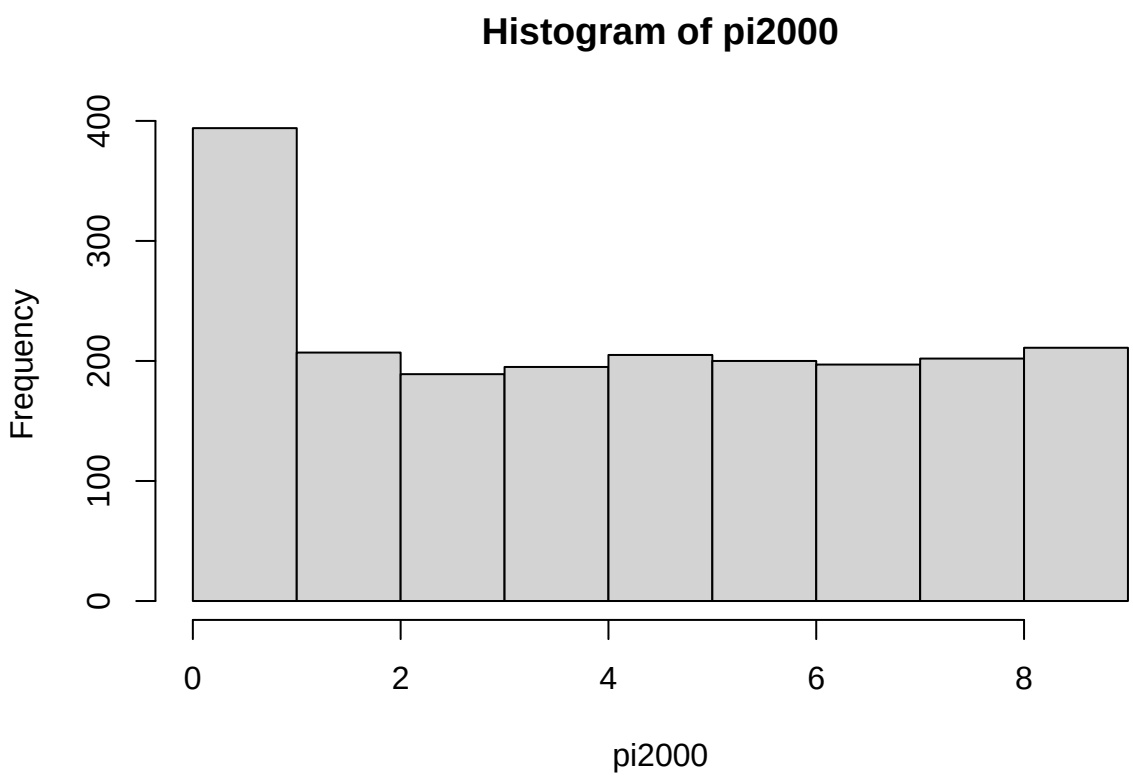
```
## Warning: package 'HistData' was built under R version 4.3.3
```

```
## Loading required package: Hmisc
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

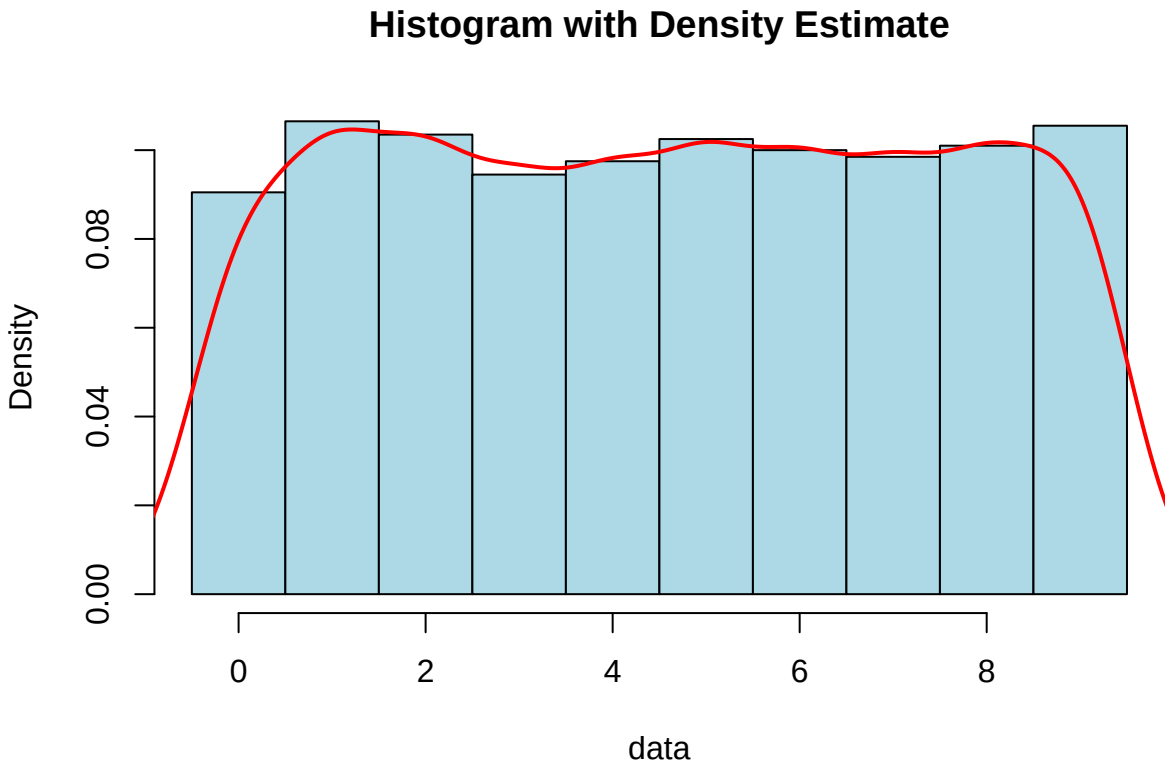
```r
data("pi2000")
hist(pi2000)
```



```r
# Define the density_compare function
density_compare <- function(data) {
  # Create a histogram
  hist(data, breaks = 0:10 - 0.5, col = "lightblue", main = "Histogram with Density Estimate", freq = F

  # Add the density estimate on top of the histogram
  density_estimate <- density(data)
  lines(density_estimate, col = "red", lwd = 2)  # Adds the density plot in red
}

# Call the function with the pi2000 dataset
density_compare(pi2000)
```




```r
# Explanation:
# The 'breaks = 0:10 - 0.5' argument in hist() creates bins with specific ranges, helping avoid strange
# It adjusts the placement of bins to better match the data distribution, especially when dealing with
```

BONUS (Verzani problem 2.25)

Write a function `isprime()` *that checks if a number* x *is prime by dividing* x *by all the values in* $2, ..., x-1$ *then checking to see if there is a remainder of 0. The expression* `a %% b` *returns the remainder of* `a` *divided by* `b`.

```r
# Define the isprime function (inefficient version)
isprime <- function(x) {
  if (x < 2) {
    return(FALSE)  # Numbers less than 2 are not prime
  }
  for (i in 2:(x-1)) {
    if (x %% i == 0) {
      return(FALSE)  # If divisible by any number other than 1 and itself, it's not prime
    }
  }
  return(TRUE)  # If no divisors are found, it's prime
}

# Test the isprime function
isprime(29)  # Should return TRUE
```

```
## [1] TRUE
```

You do not need to check all numbers from 2 to $x - 1$ to see if a number is prime. What is the largest you would need to go for an arbitrary x? Create a new function, ***`isprime2()`****, that implements this better (yet still slow) method.*

```r
# Optimized prime function using square root
isprime2 <- function(x) {
  if (x < 2) {
    return(FALSE)  # Numbers less than 2 are not prime
  }
  for (i in 2:sqrt(x)) {  # Only check divisors up to the square root of x
    if (x %% i == 0) {
      return(FALSE)  # If divisible, it's not prime
    }
  }
  return(TRUE)
}

# Test the isprime2 function
isprime2(29)  # Should return TRUE
```

```
## [1] TRUE
```