hw4

May 8, 2024

1

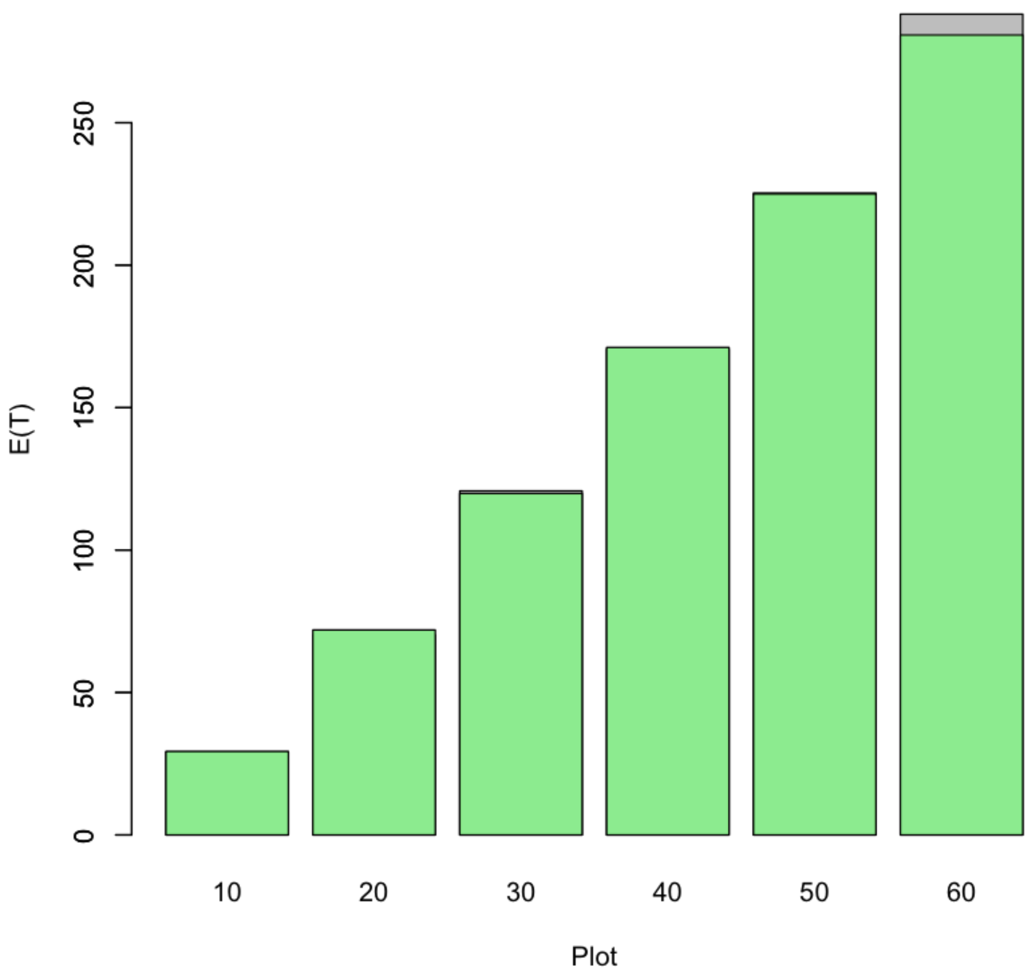
[59]:
```
# N:      the number of unique coupons
# NSim:   the number of simulations that we will perform
# num:    record the number of trials needed to get a complete
#         set of one of each type of  coupon for each simulation

data = numeric(6)
theoretical = numeric(6)
NSim=1000                                    # Number of simulations         ␣
 ↪                  # This is  a vector initialized to 0;
                                            # function rep(0,NSim) replicates the value␣
 ↪0, NSim times
for(j in c(10,20,30,40,50,60)){
    N=j
    num=rep(0,NSim)
for (i in 1:NSim){
  trials <-rep(0,0)                          # for a simulation intialize trials to␣
 ↪empty
  while (length(unique(as.vector(trials)))<N){    # until all coupons collected
    trials<-cbind(sample(1:N,1),trials)  # withdraw a coupon and add to trials␣
 ↪using cbind function
    num[i]=num[i]+1                          # increment trials
  }
}
data[j/10] = mean(num)
theoretical[j/10] = N*log(N) + 0.5771*N +0.5
print(data[j/10])
print(theoretical[j/10])
}


bar_names <- c(10, 20, 30, 40, 50, 60)
barplot(data, names.arg=bar_names, ylab="E(T)", xlab="Plot")
barplot(theoretical, add = TRUE, col = "lightgreen")
```

```
[1] 29.184
[1] 29.29685
```

```
[1] 70.242
[1] 71.95665
[1] 120.764
[1] 119.8489
[1] 170.479
[1] 171.1392
[1] 225.329
[1] 224.9562
[1] 288.108
[1] 280.7867
```



Obversation: Based on the plot, we can see that the difference bewtween the simulation result and the theoretical result are very small. So, the accuracy of the approximation is great.

2

2.1

$$P(X \geq 2) = 1 - P(X = 0) - P(X = 1) = 1 - \frac{e^{-1}(1)^0}{0!} - \frac{e^{-1}(1)^1}{1!} = 0.2642411$$

2.2

$$P(X \leq 2) = P(X = 1) + P(X = 1) + P(X = 0) = \frac{e^{-1\times 3}(1 \times 3)^2}{2!} + \frac{e^{-1\times 3}(1 \times 3)^0}{0!} + \frac{e^{-1\times 3}(1 \times 3)^1}{1!} = 0.423190$$

2.3

$$\begin{aligned} P\{X \geq 3 | X \geq 1\} &= \frac{P\{X \geq 2 \cap X \geq 1\}}{P\{X \geq 1\}} \\ &= \frac{P\{X \geq 2\}}{P\{X \geq 1\}} \\ &= \frac{0.0.2642411}{1 - (P\{X = 0\})} \\ &= 0.4180 \end{aligned}$$

3

$$\begin{aligned} \sum_{k=0}^{\infty} p_k &= \frac{e^{-\lambda t}(\lambda t)^0}{0!} + \frac{e^{-\lambda t}(\lambda t)^1}{1!} + \frac{e^{-\lambda t}(\lambda t)^2}{2!} + \frac{e^{-\lambda t}(\lambda t)^3}{3!} + \ldots\ldots \\ &= e^{-\lambda t} \times e^{\lambda t} \\ &= 1 \end{aligned}$$

4

4.1

$$P(X = 9) = \frac{e^{-9}(3 + 6)^9}{9!} = 0.1317$$

4.2

This is a Geomertic Distribution. So, we use the formula of Expection.

$$E(9) = \frac{1}{p} = \frac{1}{0.1317} = 7.59$$

5

[69]:
```
data <- read.table("./Old_Faithful.txt", header=TRUE)
head(data)
tail(data)
```

A data.frame: 6 × 3

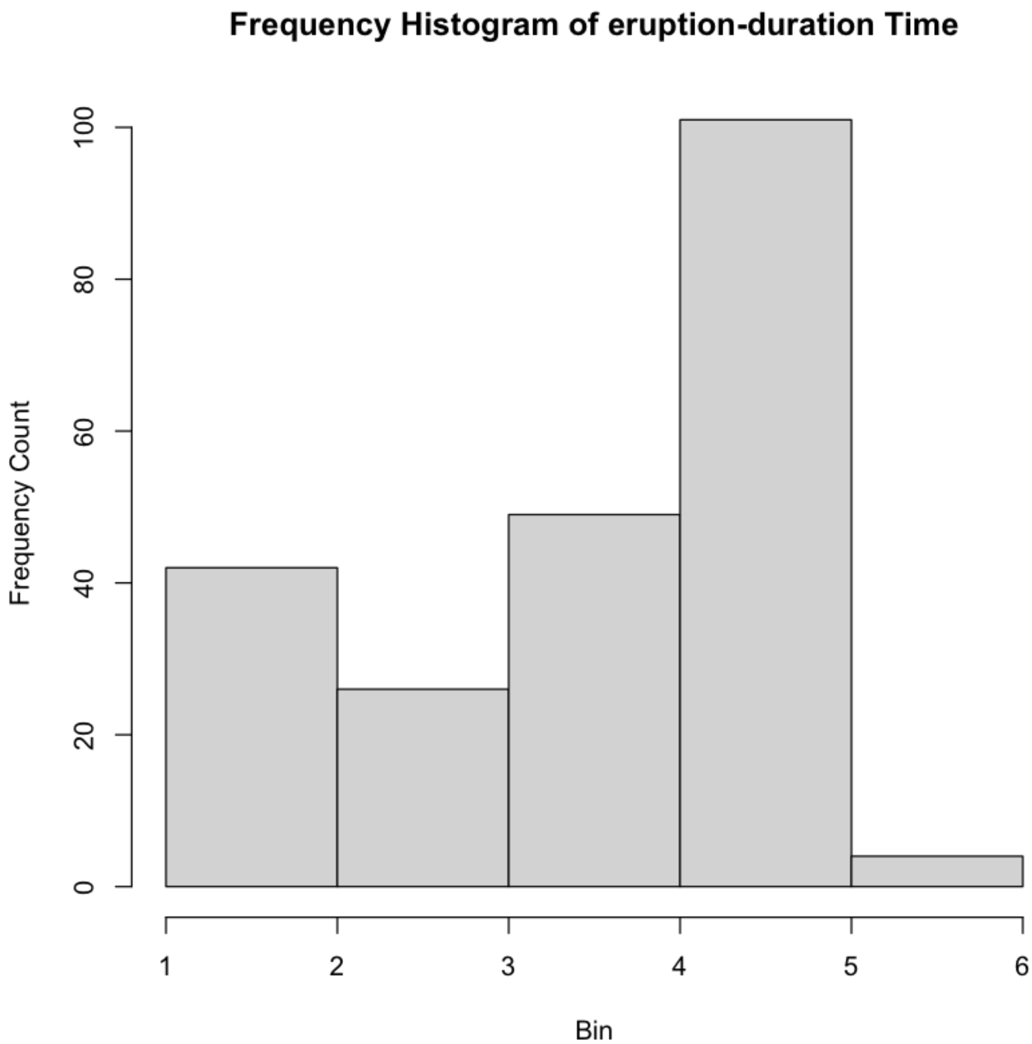
	date	time_between	duration
	<int>	<int>	<dbl>
1	1	78	4.4
2	1	74	3.9
3	1	68	4.0
4	1	76	4.0
5	1	80	3.5
6	1	84	4.1

A data.frame: 6 × 3

	date	time_between	duration
	<int>	<int>	<dbl>
217	23	79	4.5
218	23	61	2.1
219	23	81	4.2
220	23	48	2.1
221	23	84	5.2
222	23	63	2.0

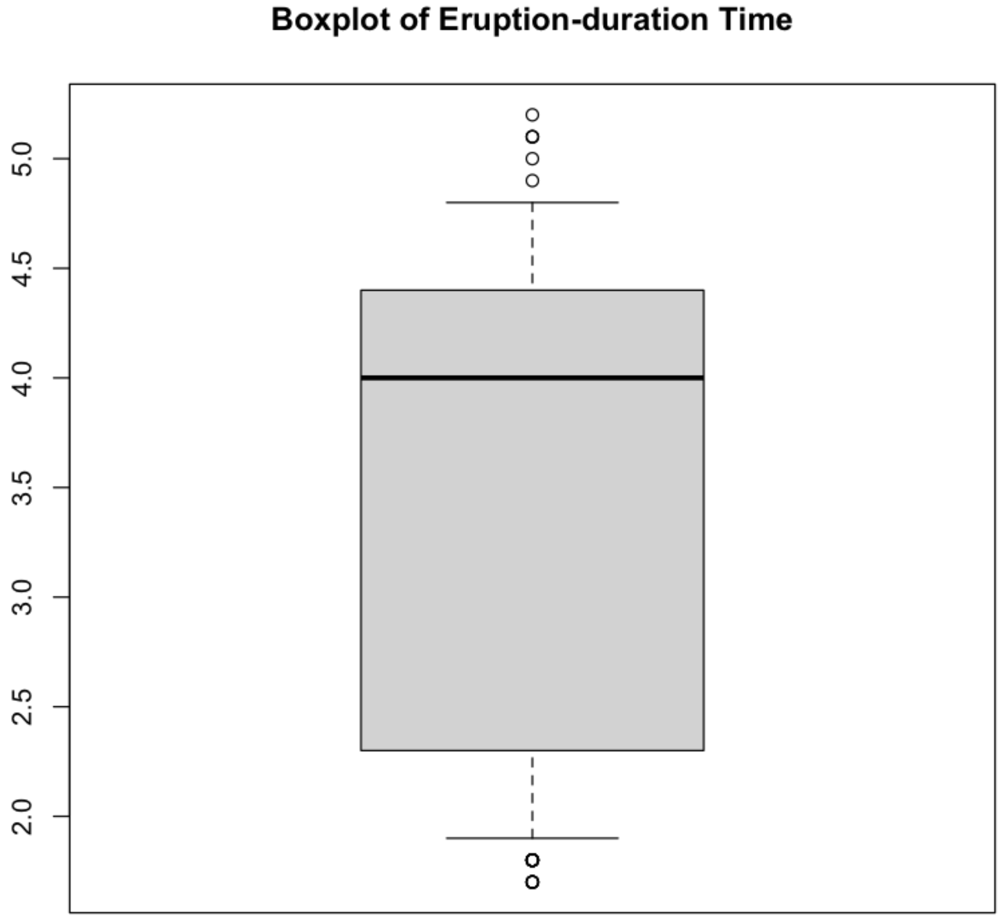
5.1

[90]:
```
low = min(data[,3])
high = max(data[,3])
hist(data[,3], breaks = seq(1,6,1), xlab = "Bin", ylab = "Frequency Count",
 ↪main = "Frequency Histogram of eruption-duration Time")
```



5.2

[91]:
```
boxplot(data[,3], range=0.2, main = "Boxplot of Eruption-duration Time")
```



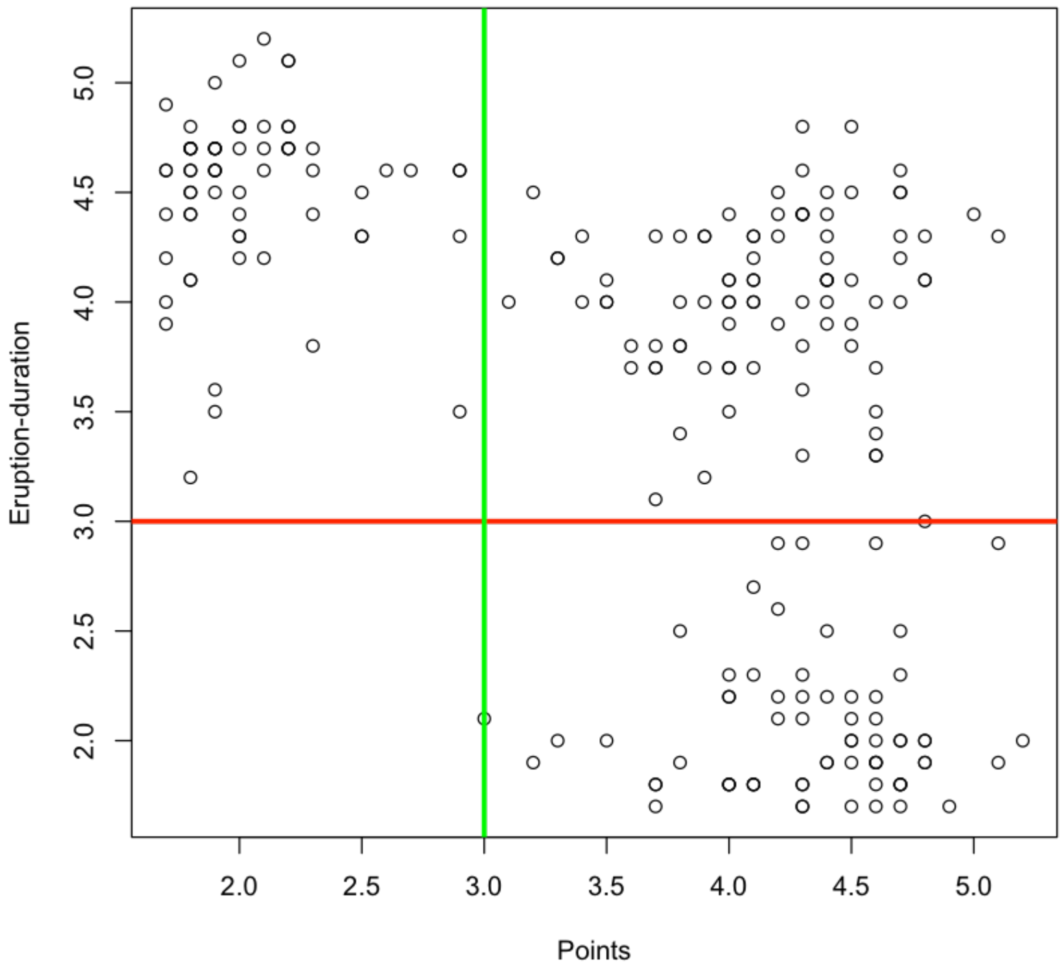
5.3

[92]:
```
q = c(.95,0.97, .99)
quantile(data[,3], q)
```

95\% 4.8 **97\%** 4.8 **99\%** 5.1

[125]:
```
a = data[,3]
b = data[,3]
for(i in 1:length(b)-1){
    b[i] = b[i+1]
}
a = a[-length(a)]
```

```r
b = b[-length(b)]
plot(x = a,y = b,ylab = "Eruption-duration", xlab="Points")
abline(h = 3, col = "red", lwd = 3)
abline(v = 3, col = "green", lwd = 3)
```



5.4

[145]:
```r
result = 0
for(i in 1:length(a)){
    if(a[i]>3 && b[i]>3){
        result = result + 1
    }
}
```

```r
print(result/length(a))

result = 0
for(i in 1:length(a)){
    if(a[i]>3 && b[i]<=3){
        result = result + 1
    }
}
print(result/length(a))

result = 0
for(i in 1:length(a)){
    if(a[i]<=3 && b[i]>3){
        result = result + 1
    }
}

print(result/length(a))

result = 0
for(i in 1:length(a)){
    if(a[i]<=3 && b[i]<=3){
        result = result + 1
    }
}

print(result/length(a))
```

```
[1] 0.3936652
[1] 0.3031674
[1] 0.2986425
[1] 0.004524887
```

- a long eruption is followed by a long eruption

$$P = 0.3936652$$

- a long eruption is followed by a short eruption

$$P = 0.3031674$$

- a short eruption is followed by a long eruption

$$P = 0.2986425$$

- a short eruption is followed by a short erupt

$$P = 0.004524887$$

```
[ ]:
```