Normal Distribution and Estimation

Computational Statistics

October 17, 2023

```
set.seed(2013)
library(tidyverse)
```

Normal distribution

Assume that the distribution of female heights is approximated by a normal distribution with a mean of 64 inches and a standard deviation of 3.5 inches.

1a. Picking a female at random, what is the probability that she is 5 feet or shorter? What is the probability that she is 6 feet or taller?

Solution: Assuming that the distribution of female heights is approximated by normal distribution with mean 64 inches and standard deviation 3.5 inches, we compute the probability for events using the Cumulative Distribution Function(CDF) which denotes the probability of occurrence of an event is probability of values that are less than or equal to a fixed value say 'a'. i.e F(a) = Pr(x ≤ a)

Probability that a woman is 5 feet or shorter can be computed in R using the pnorm() function in the following manner,

5 feet = 5x12 inches = *60 inches*

```
cat("Probability of a random female being 5 feet or shorter is: ",pnorm(60,mean=64, sd=3.5))
```

```
## Probability of a random female being 5 feet or shorter is:  0.126549
```

Probability that a woman is 6 feet or taller can be computed in R using the pnorm() function in the following manner,

6 feet = 6x12 inches = *72 inches*

Probability of a random female being 6 feet or taller is the equal to [1- (Probability that female is shorter than 6 feet)]

```
prob_short<-pnorm(72, mean=64, sd=3.5,lower.tail=TRUE)
prob_tall<-1-prob_short
cat("Probability of a random female being 6 feet or taller is: ",prob_tall)
```

```
## Probability of a random female being 6 feet or taller is:  0.01113549
```

The above result will only provide the probability of females who are strictly taller than 6 feet. The probability of a random female being 6 feet or taller can be equal to (Probability that female is strictly taller than 6 feet)+ (Probability that female is 6 feet)

```
prob_tall_2<-pnorm(72, mean=64, sd=3.5,lower.tail = FALSE) + dnorm(72, mean=64,sd=3.5)
#dnorm for probability that female is exactly 6 feet
#pnorm+lower.tail=FALSE for probability that female is taller than 6 feet
cat("Alternate probability of a random female being 6 feet or taller is: ",prob_tall_2)
```

```
## Alternate probability of a random female being 6 feet or taller is:  0.01949842
```

We observe that the probability might not be equal in both the cases but since the difference is significantly small, this can be neglected.

1b. How tall is the female in the 95th percentile?

Solution: The height of the female in the 95th percentile corresponds to the value at which 95% of the data is below it.

```
he<-qnorm(0.95,mean=64, sd=3.5)
cat("The female at 95% percentile is ", he, "inches tall i.e approximately ", round(he/12,1) ,"f
eet." )
```

```
## The female at 95% percentile is  69.75699 inches tall i.e approximately  5.8 feet.
```

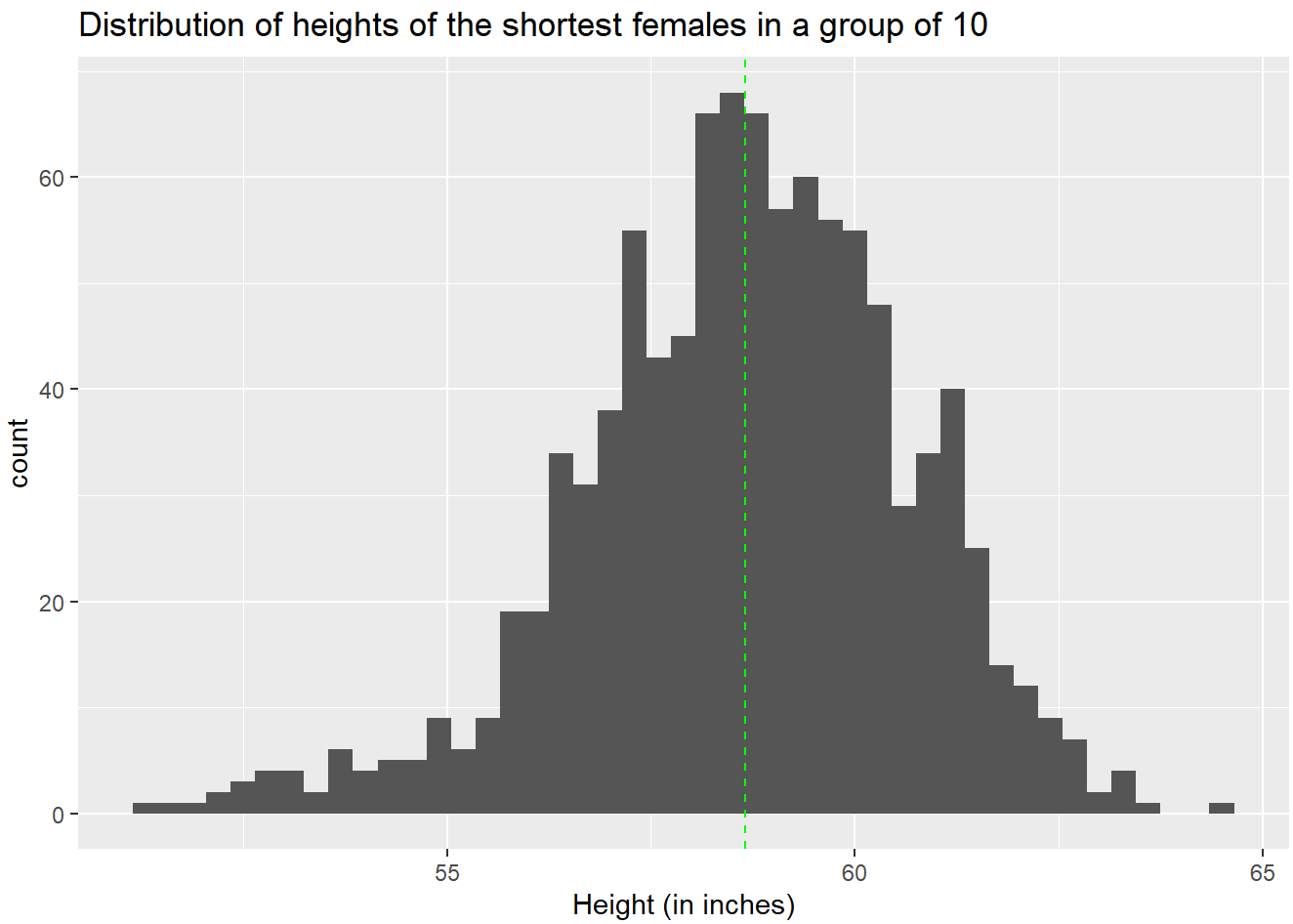
1c. What is the distribution of the height of the shortest person in a group of 10 randomly selected females? Run $B = 1000$ Monte Carlo simulation, and make a histogram.

Solution: Here we will simulate selecting 10 females at random from the given distribution and record the height of the shortest person in each simulation run. After running this for a large number of simulations (B = 1000), we create a histogram to visualize the distribution.

```
B=1000
#Simulation of shortest height of 1000 females
shortest<-replicate(B,
          {height_female<-rnorm(10,mean=64, sd=3.5)
          min(height_female)})
s<-summary(shortest)
s
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   51.16   57.43   58.75   58.66   60.02   64.48
```

```
mean_s <- s[4]
tibble(shortest) |>
      ggplot(aes(shortest))+
      labs(x="Height (in inches)" )+
      ggtitle("Distribution of heights of the shortest females in a group of 10")+
      geom_histogram(binwidth=0.3)+
      geom_vline(xintercept = mean(shortest), color="green", linetype="dashed")
```



From the summary statistics and the histogram, we can sufficiently say that the distribution of shortest heights in a random sample of 10 females repeated 1000 times also follows normal distribution with mean=58.6614942inches indicated by the green dotted line.This indicates that, in a random sample of 10 females, the average height of the shortest person is 58.6614942 inches.

Estimating the value of π

One way to estimate the value of π is by using Monte Carlo simulations. The idea is to randomly generate (x, y) points within a square with corners $(1, 1)$, $(-1, 1)$, $(-1, -1)$, and $(1, -1)$. Imagine a circle inside the square with radius of 1. We can then calculate the ratio of the number of points that lie inside the circle over the number of all the generated points. This ratio should approximate the ratio of the circle area over the square area, that is, $\pi/4$.

2a. Implement the idea to estimate the value of π. **Hint:** Use `runif()` .

Solution: Firstly we will generate a random sample of 10000 (x,y) points within a square with vertices (1,1), (-1,1), (-1,-1), (1,-1). To check if the point lies within the circle with radius 1 unit we compute the distance of each point from the origin and check if the distance is less than or equal to 1.

To estimate the value of π, we compute the ratio of the number of points that lie inside the circle over the number of all the generated points and multiply this ratio by 4 as it has been mentioned in the problem that the ratio is almost equal to $\pi/4$.

```r
# Generate a random sample of (x, y) points within a square with corners (1, 1), (-1, 1), (-1, -1), and (1, -1)
x <- runif(10000, -1, 1)
y <- runif(10000, -1, 1)

# Calculate the dist of each point from the origin
dist <- sqrt(x^2 + y^2)

# Count the no of points that lie inside the circle
count_of_points <- sum(dist <= 1)
cat("The number of (x,y) points that lie within the circle are : ", count_of_points)
```

```
## The number of (x,y) points that lie within the circle are :  7909
```

```r
# Estimate pi by multiplying the ratio of the number of points inside the circle to the total number of points by 4
pi_calculated <- 4 * count_of_points / 10000
cat("\nThe estimated value of pi is: ", pi_calculated)
```

```
##
## The estimated value of pi is:  3.1636
```

The estimated value of π is very close to the actual value (3.1415926…) To increase the accuracy of the estimated value, we can generate 100000 or more points.

Matching problem

We have a well-shuffled deck of 52 cards, labeled 1 through 52. A card is a *match* if the card's position in the deck matches the card's label. Use Monte Carlo simulations to answer the following questions.

3a. What is the probability that there is at least one match?

Solution: We generate a random sample of size 52 from the deck of cards without replacement as we are comparing the position of each card to the actual card drawn and there is only one card of a kind in the deck of cards.

```r
# Set the number of simulations
B <- 10000

# Create a vector to store whether there was a match in each simulation
log_vector <- vector(length = B)

# Simulate shuffling the deck and checking for matches
for (i in 1:B) {
  # Shuffle the deck
  deck <- sample(1:52, 52,replace=FALSE)

  # Check for matches
  log_vector[i] <- any(deck == 1:52)
}
# Calculate the proportion of simulations in which there was at least one match
p_match <- mean(log_vector>=1)

# Print the estimated probability of at least one match
print(p_match)
```

```
## [1] 0.6326
```

Therefore, the estimated probability that there is at least one match in a well-shuffled deck of 52 cards is 0.6326 .

This is a very high probability, which suggests that it is very likely that there will be at least one match in a well-shuffled deck of 52 cards.

3b. What is the the expected number of matches?

Solution:

```r
# Set the number of simulations
B <- 5000

# Create a vector to store the number of matches in each simulation
num_matches <- vector(length = B)

# Simulate shuffling the deck and counting matches
for (i in 1:B) {
  # Shuffle the deck
  deck <- sample(1:52, 52, replace=FALSE)

  # Count the number of matches
  num_matches[i] <- sum(deck == 1:52)
}

# Calculate the expected number of matches
mean_num_matches <- mean(num_matches)

# Print the expected number of matches
print(mean_num_matches)
```

```
## [1] 0.9962
```

Therefore, the estimated expected number of matches in a well-shuffled deck of 52 cards is 0.9962. Since the expected number of matches is nothing but the mean of the matches, we can infer that when a random sample is picked and checked for matches, there is high chance that there is atleast one match in the sample generated.

Waiting time until `HH` vs. `HT`

Suppose you toss a fair coin repeatedly, and you want to use Monte Carlo simulations to find the expected number of tosses until the pattern `HH` appears for the first time and the expected number of tosses until the first `HT`.

To answer this question, you start by generating a long sequence of fair coin tosses:

```
set.seed(2013)
(tosses <- paste(sample(c("H", "T"), 100, replace = TRUE), collapse = ""))
```

```
## [1] "THHHTHHHHTHHTHTTTTHTHTTHTTTHTHHTTHTHHHHHHTTTTHHHHHTTHHTHHHTHHHTHHHTTTTTHTHHHTHTTTHTHTTTH
HHTHHTTHTHHH"
```

A sequence of length 100 is long enough to nearly guarantee that both `HH` and `HT` would have appeared at least once.

Once the sequence is generated, you can use the `stringr::str_locate()` function to locate the pattern `HH`:

```
t <- str_locate(tosses, "HH")
t
```

```
##      start end
## [1,]     2   3
```

Based on `t`, the number of tosses until the first appearance of `HH` in this sequence is:

```
t[, 2]
```

```
## end
##   3
```

4a. Complete the rest of the Monte Carlo approach to find the expected number of tosses until the pattern `HH` appears for the first time?

Solution: Using Monte Carlo simulation we generate random sequences of coin tosses for 10000 times, looking for the “HH” pattern, and record the number of tosses it took for the pattern to appear in a vector *result_hh*. Then we calculate the mean of these results to estimate the expected number of tosses until “HH” appears.

```
set.seed(2013)
num_of_simulations<-10000
result_hh<-replicate(num_of_simulations, {
  tosses_hh<-paste(sample(c("H", "T"), 100, replace = TRUE), collapse = "")
  t_hh <- stringr::str_locate(tosses_hh, "HH")
  return(t_hh[, 2])  # Return the number of tosses until "HH" pattern
  })
# Calculate the mean of the results to estimate the expected number of tosses until "HH" pattern
exp_no_hh<-mean(result_hh)

# Print the result
cat("Expected number of tosses until HH appears:", exp_no_hh, "\n")
```

```
## Expected number of tosses until HH appears: 6.0272
```

```
cat("Since tosses cannot be expressed in decimals we can round this value : ", round(exp_no_hh,
0))
```

```
## Since tosses cannot be expressed in decimals we can round this value :  6
```

4b. What is the expected number of tosses until the pattern `HT` appears for the first time?

Solution: Using Monte Carlo simulation we generate random sequences of coin tosses for 10000 times, looking for the “HT” pattern, and record the number of tosses it took for the pattern to appear in a vector *result_ht*. Then we calculate the mean of these results to estimate the expected number of tosses until “HT” appears.

```
num_of_simulations<-10000
result_ht<-replicate(num_of_simulations, {
  tosses_ht<-paste(sample(c("H", "T"), 100, replace = TRUE), collapse = "")
  t_ht <- stringr::str_locate(tosses_ht, "HT")
  return(t_ht[, 2])  # Return the number of tosses until "HH" pattern
  })
# Calculate the mean of the results to estimate the expected number of tosses until "HH" pattern
exp_no_ht<-mean(result_ht)

# Print the result
cat("Expected number of tosses until HT appears:", exp_no_ht, "\n")
```

```
## Expected number of tosses until HT appears: 4.0137
```

```
cat("Since tosses cannot be expressed in decimals we can round this value to : ", round(exp_no_h
t,0))
```

```
## Since tosses cannot be expressed in decimals we can round this value to :  4
```