~~~
---
title: "Estimations"
author: "S20426"
date: "2025-03-15"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r}
# Generate synthetic data from an exponential distribution
set.seed(123)
data <- rexp(100, rate = 0.1)

# Likelihood function for the exponential distribution
log_likelihood <- function(lambda) {
  sum(dexp(data, rate = lambda, log = TRUE))
}

# Maximum Likelihood Estimation using optim
result <- optim(par = 0.1, log_likelihood, method = "Brent", lower = 0.001, upper = 10)
mle_lambda <- result$par

# Print the MLE estimate
cat("Maximum Likelihood Estimate for Lambda:", mle_lambda, "\n")


```

```{r}
library(ggplot2)
# Plot the histogram of the data with the estimated lambda
ggplot(data = NULL, aes(x = data)) +
  geom_histogram(bins = 15, fill = "skyblue", color = "black", alpha = 0.7) +
  geom_vline(xintercept = mle_lambda, color = "red", linetype = "dashed") +
  annotate("text", x = mle_lambda + 0.1, y = 10,
           label = paste("MLE Lambda:", round(mle_lambda, 2)), color = "red") +
  labs(title = "Histogram with Estimated Lambda", x = "Time", y = "Frequency")

```

### Example 01

```{r}
# ?optimize
~~~

~~~
log_likelihood <- function(p) {

  x <- 103
  n <- 200

  if (p == 0 || p == 1) {
    return(-Inf)  # To avoid log(0) which is undefined
  }
  return(x * log(p) + (n - x) * log(1 - p))
}

result <- optimize(log_likelihood, interval = c(0, 1), maximum = TRUE)
result
```

### Example 02

```{r}

weights <- c(59.001, 38.267, 41.025, 35.555, 46.690, 20.994, 39.407, 52.780,
             57.495, 52.416, 60.062, 48.149, 40.182, 50.929, 49.472, 49.197,
             43.459, 40.493, 60.196, 58.590, 53.645, 53.837, 61.134, 62.115,
             46.517, 41.404, 56.500, 53.281, 44.821, 47.610, 51.178, 58.315,
             34.411, 47.795, 41.828, 60.767, 60.797, 51.421, 51.570, 48.313,
             47.310, 58.078, 38.753, 35.692, 50.604, 42.070, 53.403, 47.405,
             36.952, 53.682)

normal <- function(x,data)(

  return (
    sum(dnorm(x,mean(data),sd(data),log = TRUE))
  )
)
result <- optimize(normal,interval = c(-100,+100),data=weights, maximum = TRUE)
result

```

### Confidence Interval for Mean

#### Case 1: When data is normal/ large sample and σ is known.

```{r}

# set.seed(42) Ensuring reproducibility in simulations.
#same sequence of random numbers is generated every time you run the code.


set.seed(20)
~~~

~~~
sample_size<-500
pop_div<-10
weight<-sample(45:80,size=sample_size,replace = T)
sample_mean<- mean(weight)
z_critical <- qnorm(0.975) # calculate the z - critical value
margin_error <-z_critical*(pop_div/sqrt(sample_size))


vec<-c(sample_mean - margin_error,sample_mean+ margin_error)
vec
```

#### Case 2 : When data is normal / large sample and σ is unknown

```{r}

set.seed(20)
large_sample_weight <- sample(weight,150)
large_sample_t_critical <- qt(0.975,df=149)  # find the t value
large_sample_mean <- mean(large_sample_weight)
large_sample_stdev <- sd(large_sample_weight)


large_sample_margin_of_error <- large_sample_t_critical*(large_sample_stdev/sqrt(150))

large_sample_confidence_interval <- c(large_sample_mean- large_sample_margin_of_error,large_sample_mean +
large_sample_margin_of_error)

large_sample_confidence_interval

```

#### Case 3 : When data is non-normal /small samples

```{r}
#When the data is non-normal or when sample sizes are small, traditional parametric methods (like #using normal
distribution assumptions) may not be reliable. Bootstrapping is a resampling technique #that helps estimate the sampling
distribution of a statistic (e.g., the mean) without assuming #normality.



library(boot)

blood_pressure <- c(72,66,64,66,40,74,50,70,96,92,74,80,60,72,84,74,80,88,94)

mean_fn <- function(x ,indices){
  return (mean(x[indices]))
}
#Performs bootstrap resampling 999 times (R=999).
~~~

~~~
#The argument indices represents the randomly selected indices in each bootstrap resample.

level.boot <- boot(blood_pressure  , mean_fn ,R=999)
boot.ci(level.boot,conf = 0.95)
```

### Confidence Interval for Differnce of Means

#### Case 1: Sampling from two independent normal distributions with known variances.

x            ---> A numeric vector representing the sample data for a one-sample test or the first sample in a two-
sample test.

y           --->  (Optional) A numeric vector for the second sample in a two-sample Z-test. If  NULL, a one-sample Z-
test is performed.

alternative --->  Specifies the alternative hypothesis. Options: "two.sided" (default), "less" (one-tailed, left-side
test), or "greater" (one-tailed, right-side test).

sigma.x     --->  The known population standard deviation for sample x. If NULL, an estimate from   the sample is used,
but this is not a true Z-test.

sigma.y     --->  The known population standard deviation for sample y (if performing a two-sample test). If NULL, an
estimate from y is used.

mu          ---> The hypothesized population mean for a one-sample test (default is 0). conf.level  The confidence level
for the confidence interval (default is 0.95 or 95%).
```{r include=FALSE}
library("BSDA")
```


```{r}
#library("BSDA")
sample1 <- c(52, 55, 49, 50, 53)
sample2 <- c(47, 50, 48, 51, 49)

# Perform two-sample Z-test
z.test(x = sample1, y = sample2, sigma.x = 3, sigma.y = 2, alternative = "two.sided")


```

#### Case 2: Sampling from two independent normal distributions with unknown variances (small samples).

##### when population variances are equal

```{r}
sample1 <- c(52, 55, 49, 50, 53)
~~~

~~~
sample2 <- c(47, 50, 48, 51, 49)

# Perform two-sample Z-test
t.test(x = sample1, y = sample2,  alternative = "two.sided",var.equal=TRUE)
```

##### when population variances are unequal

```{r}
sample1 <- c(52, 55, 49, 50, 53)
sample2 <- c(47, 50, 48, 51, 49)

# Perform two-sample Z-test
? t.test()
t.test(x = sample1, y = sample2,  alternative = "two.sided",var.equal=FALSE)
```

```{r}
set.seed(123456)                  # Create example data
data <- data.frame(x = c("A","B","C"),
                   y = round(runif(3, 10, 20),2),
                   lower = round(runif(3, 0, 10),2),
                   upper = round(runif(3, 20, 30),2))

library(ggplot2)
ggplot(data, aes(x, y)) +         # ggplot2 plot with confidence intervals
  geom_point() +
  geom_errorbar(aes(ymin = lower, ymax = upper))
```

### Confidence Intervals for Proportion

#### Case 1: For large sample (Using Normal approximation)

```{r}
set.seed(10)
hair_col <- c(rep("black",1500),rep("brown",1000),rep("blonde",500))
sampleP <- sample(hair_col,1000)
Htable <- table(sampleP)
Htable
prop.table(Htable)

z=qnorm(0.975)
p=0.498
n=1000

margin_error = z*sqrt(p*(1-p)/n)
Interval=c(p-margin_error,p+margin_error)
Interval
~~~

~~~
```
#### Case 1: For large sample (Using Binomial Distribution)


```{r}
library(epitools)

binom.exact(x=48,n=100,conf.level=0.95)
binom.wilson(x=498,n=100,conf.level = 0.95)
binom.approx(x=498,n=100,conf.level = 0.95)

```
#### Case 2: For small sample (Using Binomial Distribution)

When sample size is small, confidence interval for population can be calculated using binom.test() function.

```{r}

gender = c("f","f","f","m","m","f","f","m","m","f")
table(gender)
binom.test(6,10,conf.level = 0.95)
```
### Confidence Intervals for Variance

#### Case 1: Under normality assumption

User defined function to obtain confidence interval for variance.

```{r}
 var.interval = function(data, conf.level = 0.95) {
 df = length(data) - 1
 chilower = qchisq((1 - conf.level)/2, df)
 chiupper = qchisq((1 - conf.level)/2, df, lower.tail = FALSE)
 v = var(data)
 c(df * v/chiupper, df * v/chilower)
 }

 lizard = c(6.2, 6.6, 7.1, 7.4, 7.6, 7.9, 8, 8.3, 8.4, 8.5, 8.6, 8.8, 8.8, 9.1, 9.2, 9.4, 9.4, 9.7, 9.9, 10.2, 10.4,
10.8, 11.3, 11.9)

 var.interval(lizard)
```

#### Case 2: Under non-normality assumption

When no assumption is made about data, a bootstrap method is used to obtain confidence intervals for the population
variance.
~~~

~~~
```{r}

library(boot)

blood_pressure <- c(72,66,64,66,40,74,50,70,96,92,74,80,60,72,84,74,80,70,88,94)

variance <- function(x,indicies) var(x[indicies])

level.boot <- boot(blood_pressure,variance,R=999)
boot.ci(level.boot,conf = 0.95)


```
~~~