LDA PRACTICAL:

PRACTECAL 1

1. Generate sufficient random numbers from exponential distribution where the parameter is of your choice. Let consider these generated numbers as a lifetime of the patients then obtain the patient lifetime using the following censoring schemes:
Right Censoring
Type-I Censoring Scheme
Type-II Censoring Scheme
Random Censoring
Left Censoring
Interval Censoring
Also compare the censored data with the complete data.

CODE:

```
rate <- 0.1
lifetimes <- rexp(100, rate)
lifetimes
summary(lifetimes)
# Type 1 censoring
ctime <- 15
ctime
cdata <- ifelse(lifetimes<=ctime,lifetimes, ctime)
cstuts <- ifelse(lifetimes<=ctime,1, 0)
type1.data <- cbind(cdata, cstuts)
print(type1.data)
# Type 2 Censoring
ctime = sort(lifetimes)[80]
ctime
cdata <- ifelse(lifetimes<=ctime,lifetimes, ctime)
cstuts <- ifelse(lifetimes<=ctime,1, 0)
type2.data <- cbind(cdata, cstuts)
print(type2.data)
# Random Censoring
ctime <- rexp(100, rate)
ctime
cdata <- ifelse(lifetimes<=ctime,lifetimes, ctime)
cstuts <- ifelse(lifetimes<=ctime,1, 0)
random.data <- cbind(cdata, cstuts)
print(random.data)
# Left Censoring
ctime <- 3
ctime
cdata <- ifelse(lifetimes<=ctime, ctime, lifetimes)
cstuts <- ifelse(lifetimes<=ctime, 0,1)
left.data <- cbind(cdata, cstuts)
print(left.data)
# Interval Censoring
intervals <- seq(0,60, 6)
intervals
freq <- table(cut(lifetimes, breaks = intervals))
print(freq)
```

Practical 02

Obtain a sufficiently large sample from an exponential distribution under a Type II
censoring scheme. After generating the sample, calculate the maximum likelihood
(ML) estimate of the distribution parameter. Also, evaluate the performance of the
estimate by computing the bias, variance, and mean squared error (MSE) for different
sample sizes.

CODE:

```
table <- data.frame(
  no_obs = c(),
  no_cen_obs = c(),
  lamda = c(),
  lamda_hat = c(),
  bias = c(),
  variance = c(),
  MSE = c()
)
lamda = 0.1
no_of_cen_obs = c(10,20,30,40,50,60,70,80,90,100)

for( k in no_of_cen_obs){
  lamda_tem <- numeric(100)
  for(i in 1:100){
    set.seed(i+1)
    lifetimes <- rexp(100, lamda)
    ctime = sort(lifetimes)[k]
    cdata <- ifelse(lifetimes<=ctime,lifetimes, ctime)
    lamda_tem[i] = k/sum(cdata)
  }
  var = var(lamda_tem)
  mse = sum((lamda_tem- lamda)^2)/100
  lamda_hat= mean(lamda_tem)
  bias = lamda_hat - lamda
  new_row = data.frame(no_obs = 100,
                       no_cen_obs = k,
                       lamda = lamda,
                       lamda_hat = lamda_hat,
                       bias = bias,
                       variance = var,
                       MSE = mse)
  table = rbind(table, new_row)

}
table
```

#practical 3

Obtain a sufficiently large sample from a Weibull distribution under a Type I censoring scheme. After generating the sample, calculate the maximum likelihood (ML) estimate of the distribution parameters. Also, evaluate the performance of the estimate by computing the bias, variance, and mean squared error (MSE) for different sample sizes.

CODE:

```
# Parameters
n_values <- c(100, 200, 300)      # Sample sizes
true_shape <- 2                   # True Weibull shape parameter
true_scale <- 3                   # True Weibull scale parameter
censoring_time <- 5               # Censoring time for Type-I censoring
simulation <- 1000                # Number of simulations

# Log-likelihood function for censored Weibull data
weibull_log_likelihood <- function(params, data, censoring_time) {
  shape <- params[1]; scale <- params[2]
  uncensored <- data[data <= censoring_time]
  censored <- data[data > censoring_time]

  # Log-likelihood: uncensored and censored contributions
  ll_uncensored <- sum(dweibull(uncensored, shape, scale, log = TRUE))
  ll_censored <- sum(pweibull(censored, shape, scale, lower.tail = FALSE, log.p = TRUE))

  return(-(ll_uncensored + ll_censored))  # Negative log-likelihood
}

# Performance Evaluation for Type-I Censoring
evaluate_performance <- function(n, shape, scale, censoring_time, simulation) {
  shape_estimates <- scale_estimates <- numeric(simulation)

  for (i in 1:simulation) {
    # Generate and censor data
    life <- rweibull(n, shape, scale)
    censored_life <- pmin(life, censoring_time)

    # Estimate parameters via MLE
    fit <- optim(par = c(1, 1), fn = weibull_log_likelihood, data = censored_life,
                 censoring_time = censoring_time, method = "L-BFGS-B", lower = c(0.1, 0.1))
    shape_estimates[i] <- fit$par[1]
    scale_estimates[i] <- fit$par[2]
  }

  # Calculate bias, variance, MSE
  return(list(
    bias_shape = mean(shape_estimates) - shape,
    bias_scale = mean(scale_estimates) - scale,
    var_shape = var(shape_estimates),
    var_scale = var(scale_estimates),
    mse_shape = mean((shape_estimates - shape)^2),
    mse_scale = mean((scale_estimates - scale)^2)
  ))
}
# Run simulations and collect results
results <- data.frame()
for (n in n_values) {
  perf <- evaluate_performance(n, true_shape, true_scale, censoring_time, simulation)
  results <- rbind(results, data.frame(
    n = n, shape_hat = true_shape - perf$bias_shape, scale_hat = true_scale - perf$bias_scale,
    bias_shape = perf$bias_shape, bias_scale = perf$bias_scale,
    var_shape = perf$var_shape, var_scale = perf$var_scale,
    mse_shape = perf$mse_shape, mse_scale = perf$mse_scale
```

```
  ))
}

print(results)

#prac4 type2 wibull

set.seed(123)
n_values <- c(100, 200, 300)     # Sample sizes
true_shape <- 2                  # True Weibull shape parameter
true_scale <- 3                  # True Weibull scale parameter
c2 <- 80                         # Number of observed events before censoring (Type-II)
simulation <- 1000               # Number of simulations

# Log-likelihood function for Type-II censored Weibull data
weibull_log_likelihood <- function(params, data, c2) {
  shape <- params[1]; scale <- params[2]
  uncensored <- data[1:c2]                  # First c2 events are observed (uncensored)
  censored_part <- (length(data) - c2) * log(pweibull(data[c2], shape, scale, lower.tail = FALSE))
  ll_uncensored <- sum(dweibull(uncensored, shape, scale, log = TRUE))
  return(-(ll_uncensored + censored_part))    # Negative log-likelihood
}

# Performance Evaluation for Type-II Censoring
evaluate_performance <- function(n, shape, scale, c2, simulation) {
  shape_estimates <- scale_estimates <- numeric(simulation)

  for (i in 1:simulation) {
    # Generate and sort Weibull sample
    life <- sort(rweibull(n, shape, scale))

    # Estimate parameters using MLE on censored data
    fit <- optim(par = c(1, 1), fn = weibull_log_likelihood, data = life,
                 c2 = c2, method = "L-BFGS-B", lower = c(0.1, 0.1))
    shape_estimates[i] <- fit$par[1]
    scale_estimates[i] <- fit$par[2]
  }

  # Calculate bias, variance, and MSE
  return(list(
    bias_shape = mean(shape_estimates) - shape,
    bias_scale = mean(scale_estimates) - scale,
    var_shape = var(shape_estimates),
    var_scale = var(scale_estimates),
    mse_shape = mean((shape_estimates - shape)^2),
    mse_scale = mean((scale_estimates - scale)^2)
  ))
}

# Run simulations and collect results for different sample sizes
results <- data.frame()
for (n in n_values) {
  perf <- evaluate_performance(n, true_shape, true_scale, c2, simulation)
  results <- rbind(results, data.frame(
    n = n, shape_hat = true_shape - perf$bias_shape, scale_hat = true_scale - perf$bias_scale,
    bias_shape = perf$bias_shape, bias_scale = perf$bias_scale,
```

```
    var_shape = perf$var_shape, var_scale = perf$var_scale,
    mse_shape = perf$mse_shape, mse_scale = perf$mse_scale
  ))
}

print(results)
```

PRACTECAL 4:
CODE:

```
# Short version of the Weibull MLE analysis code
# Generate censored Weibull samples
generate_censored_weibull <- function(n, shape, scale, r) {
  sample <- sort(rweibull(n, shape, scale))[1:r]
  list(sample = sample, indicator = c(rep(1, r), rep(0, n - r)))
}

# MLE for Weibull parameters
mle_weibull <- function(sample, indicator) {
  log_likelihood <- function(params) {
    shape <- params[1]; scale <- params[2]
    uncensored <- sum(log(shape) - shape * log(scale) + (shape - 1) * log(sample) - (sample / scale)^shape)
    censored <- sum(- (sample / scale)^shape)
    -(uncensored + censored)
  }
  optim(c(1, mean(sample)), log_likelihood, method = "L-BFGS-B", lower = c(0.01, 0.01))$par
}

# Performance evaluation
evaluate_performance <- function(true_shape, true_scale, n, r, sims) {
  results <- replicate(sims, {
    data <- generate_censored_weibull(n, true_shape, true_scale, r)
    mle_weibull(data$sample, data$indicator)
  })
  shape <- results[1, ]; scale <- results[2, ]
  list(
    shape = c(bias = mean(shape) - true_shape, var = var(shape), mse = mean((shape - true_shape)^2)),
    scale = c(bias = mean(scale) - true_scale, var = var(scale), mse = mean((scale - true_scale)^2))
  )
}

# Parameters
true_shape <- 2; true_scale <- 5
sizes <- c(50, 100, 200); censoring <- 0.8; sims <- 100
results <- lapply(sizes, function(n) {
  r <- floor(n * (1 - censoring))
  evaluate_performance(true_shape, true_scale, n, r, sims)
})
Results
```

#Practical 05

The following data represents the readmission times(weeks) for 21 people of
leukemia patients. Group 1 is the treatment group and group 2 is the placebo group

Group 1(treatment): 6,6,6,7,10,13,16,22,23,6+,9+,10+,11+,17+,19+,20+,25+,32+,32+,34+,35+

Group 2(placebo) : 1,1,2,2,3,4,4,5,5,8,8,8,8,11,11,12,12,15,17,22,23

Note: + denotes censored
Estimate the survival and hazard curve for both group by the Kaplan-Meier method.

CODE :

```
G1 <- c(6, 6, 6, 7, 10, 13, 16, 22, 23, 6, 9, 10, 11, 17, 19, 20, 25, 32, 32, 34, 35)
G1_event <- c(rep(1,9), rep(0, 12))
G1_data <- data.frame(time = G1, event = G1_event)

G2 <- c(1, 1, 2, 2, 3, 4, 4, 5, 5, 8, 8, 8, 8, 11, 11, 12, 12, 15, 17, 22, 23)
G2_event <- c(rep(1,21))
G2_data <- data.frame(time = G2, event = G2_event)

km_estimator <- function(data){
  data = data[order(data$time),]
  n <- length(data$time)
  data$r <- 1:n
  for(i in 1:n){
    if(data$event[i] == 1){
      data$prob[i] <- (n-data$r[i])/(n-data$r[i]+1)
    }
    else{
      data$prob[i] =0
    }
  }
  data1 <- data[data$event==1,]
  data1$sur <- cumprod(data1$prob)
  return(data1)
}
km1 <- km_estimator(G1_data)
km1
km2 <- km_estimator(G2_data)
km2
plot(km1$time, km1$sur, type= 's', xlim = c(0,35), ylim = c(0,1), ylab = 'Probability',
     xlab = "Time")
lines(km2$time, km2$sur, type= 's', col = "blue")
```

#Practical 6

a)

A clinical trial investigates the effectiveness of a new drug on 10 lung cancer patients. The study started in January 2011 and continued till December 2012. The data is divided into two groups. Group A (treated with the new drug) and group B (treated with the standard treatment). The survival time is given in months. During the trial, some people joined. The data looks like below:

Patient id Group Survival time(months)

Event(1=death,0=censored)

1 A 5 1
2 A 12 1
3 A 18 0(censored)
4 A 20 0(censored)
5 A 24 1
6 B 8 1
7 B 10 0(censored)
8 B 15 1
9 B 16 1
10 B 22 1

Calculate and plot the PL estimate of the probability of surviving 2 years or more for both groups. Also, comment on the plots.

b) Suppose that 10 patients joined at the beginning of 24 months; during those months 4 patients died and 6 patients survived. Estimate the proportion of patients in the population surviving for 24 months or more if the study terminates at 24 months.

CODE:

```
# Input data
time <- c(5, 12, 18, 20, 24, 8, 10, 15, 18, 22)
event <- c(1, 1, 0, 0, 1, 1, 0, 1, 1, 1)  # 1 = death, 0 = censored
group <- c("A", "A", "A", "A", "A", "B", "B", "B", "B", "B")

# Split data by group
group_A <- which(group == "A")
group_B <- which(group == "B")

# Function to calculate Kaplan-Meier estimates manually
km_manual <- function(times, events) {
  # Sort by time
  order_idx <- order(times)
  times <- times[order_idx]
 events <- events[order_idx]

  # Initialize variables
  survival <- 1
  km_survival <- c(survival)
  n_risk <- length(times)  # Start with all patients at risk

  # Iterate through time points
  for (i in 1:length(times)) {
    if (events[i] == 1) {  # Only consider uncensored data (death)
      survival <- survival * (1 - 1 / n_risk)  # Update survival probability
    }
    km_survival <- c(km_survival, survival)
    n_risk <- n_risk - 1  # Reduce the number of people at risk
  }

  return(km_survival[-1])  # Return survival probabilities (excluding initial 1)
}

# Calculate KM estimates for Group A and Group B
km_A <- km_manual(time[group_A], event[group_A])
km_B <- km_manual(time[group_B], event[group_B])
```

```
# Plot KM estimates for Group A
plot(c(0, time[group_A]), c(1, km_A), type = "s", col = "blue", ylim = c(0, 1),
    xlab = "Time in months", ylab = "Survival Probability",
    main = "Kaplan-Meier Survival Curves", lwd = 2, las=1)

# Add KM estimates for Group B
lines(c(0, time[group_B]), c(1, km_B), type = "s", col = "red", lwd = 2)

# Annotate points on Group A curve with survival probabilities
text(c(0, time[group_A]), c(1, km_A), labels = round(c(1, km_A), 2),
    col = "blue", pos = 3, cex = 0.8)

# Annotate points on Group B curve with survival probabilities
text(c(0, time[group_B]), c(1, km_B), labels = round(c(1, km_B), 2),
    col = "red", pos = 3, cex = 0.8)

# Add legend to distinguish between groups
legend("bottomleft", legend = c("Group A", "Group B"), col = c("blue", "red"), lwd = 2)

# Display the KM estimates in the console
cat("Kaplan-Meier survival estimates for Group A:\n", round(km_A, 2), "\n")
cat("Kaplan-Meier survival estimates for Group B:\n", round(km_B, 2), "\n")

# Part b data
total_patients <- 10
deaths <- 4
survivors <- total_patients - deaths

# Proportion of patients surviving after 24 months
proportion_surviving <- survivors / total_patients
cat("Proportion of patients surviving after 24 months: ", proportion_surviving, "\n")
```

#PRACTECAL 7-prabability paper

1. Generate sufficient random numbers from a normal distribution where the parameters are of your choices. Construct a normal probability paper and show that the generated sample gives the evidence to belong to the normal distribution. Also, comment on your result.
2. Generate sufficient random numbers from a distribution as per your roll number. Let's consider these generated numbers as the failure time of machines. First construct a probability paper for fitting purpose and then verify that the generated data well fits on the probability paper. Also, comment on your result.

CODE:

```
x1<-qnorm(seq(0.01,0.99,0.01))
x2<-qnorm(seq(0.01,0.99,0.01))
plot(x1,x2,type="n")
abline(h=x1)
abline(v=x1)
```

```
n=300
s1<-sort(rnorm(n))
tq<-qnorm(seq(0.01,0.99,length=n))
points(s1,tq,col=2,pch=16)

#q2exp distn
x1<-qexp(seq(0.01,0.99,0.01))
x2<-qexp(seq(0.01,0.99,0.01))
plot(x1,x2,type="n")
abline(h=x1)
abline(v=x1)
n=300
s1<-sort(rexp(n))
tq<-qexp(seq(0.01,0.99,length=n))
points(s1,tq,col=2,pch=16)

# Lognormal Distribution
x1 <- qlnorm(seq(0.01, 0.99, 0.01), meanlog = 0, sdlog = 1)  # Theoretical Lognormal quantiles
x2 <- qlnorm(seq(0.01, 0.99, 0.01), meanlog = 0, sdlog = 1)
plot(x1, x2, type = "n", main = "Lognormal Q-Q Plot")
abline(h = x1)
abline(v = x1)
n <- 300
s1 <- sort(rlnorm(n, meanlog = 0, sdlog = 1))  # Sample Lognormal data
theoretical_quantile <- qlnorm(seq(0.01, 0.99, length = n), meanlog = 0, sdlog = 1)
points(s1, theoretical_quantile, col = 2, pch = 16)

# Weibull Distribution
x1 <- qweibull(seq(0.01, 0.99, 0.01), shape = 2)  # Theoretical Weibull quantiles with shape parameter 2
x2 <- qweibull(seq(0.01, 0.99, 0.01), shape = 2)
plot(x1, x2, type = "n", main = "Weibull Q-Q Plot")
abline(h = x1)
abline(v = x1)
n <- 300
s1 <- sort(rweibull(n, shape = 2)) #sample points
theoretical_quantile <- qweibull(seq(0.01, 0.99, length = n), shape = 2)
points(s1, theoretical_quantile, col = 2, pch = 16)

# Gamma Distribution
x1 <- qgamma(seq(0.01, 0.99, 0.01), shape = 2)  # Theoretical Gamma quantiles with shape parameter 2
x2 <- qgamma(seq(0.01, 0.99, 0.01), shape = 2)
plot(x1, x2, type = "n", main = "Gamma Q-Q Plot")
abline(h = x1)
abline(v = x1)
n <- 300
s1 <- sort(rgamma(n, shape = 2))  # Sample Gamma data with shape parameter 2
theoretical_quantile <- qgamma(seq(0.01, 0.99, length = n), shape = 2)
points(s1, theoretical_quantile, col = 2, pch = 16)
```

PRACTECAL 8

Patient ID Failure time(months) Patient ID Failure time(months)

1 16.5 16 15.4

2 11.7 17 9.9

3 25.3 18 18.7

4 7.8 19 30.6
5 19.2 20 12.3
6 10.6 21 21.1
7 22.7 22 4.9
8 5.1 23 13.5
9 13.9 24 16.1
10 24.6 25 6.2
11 17.3 26 19.8
12 8.4 27 27.4
13 28.2 28 14.7
14 6.7 29 23.9
15 20.5 30 26.5

Suppose that the above data has been extracted from an experiment. The data represents
the lifetime of patients. First, make a probability plot for the data and verify the distribution from which the data has been generated, and then estimate the parameter by using a graphical method.

CODE:

```
##pplot

f_t=c(16.5, 11.7, 25.3, 7.8, 19.2, 10.6, 22.7, 5.1, 13.9, 24.6, 17.3, 8.4, 28.2, 6.7, 20.5, 15.4, 9.9, 18.7, 30.6,
12.3,21.1, 4.9, 13.5, 16.1, 6.2, 19.8, 27.4, 14.7, 23.9, 26.5)
n = length(f_t)
f_t1 = sort(f_t)
prob <- (1:n - 0.5) / n
mean_time = mean(f_t1)
sd_time = sd(f_t1)
t_q =qnorm(prob,mean = mean_time, sd = sd_time)

x1=qnorm(seq(0.01,.99,.01),mean_time,sd_time)
x2=qnorm(seq(0.01,.99,.01),mean_time,sd_time)
plot(x1,x2, type="n",xlim=c(0,35),ylim=c(0,35))
abline(h=x1,xlim=c(0,35))
abline(v=x2,ylim=c(0,35))

points(t_q, f_t1, main = "Normal Probability Plot", xlab = "Theoretical Quantiles", ylab = "Ordered Failure Times"
,col="red",pch=18)
abline(0,1,col="blue",lwd=2)
```

PRACTECAL 9

The remission times of 42 patients with acute leukemia were reported in a clinical trial to assess the ability of 6-mercaptopurine(6-MP) to maintain remission. Patients were randomly selected to receive 6-MP or placebo. The study was terminated after one year. The following remission times, in weeks, were recorded:

6-MP (21 patients) Placebo (21 patients)
6,6,6,7,10,13,16,22,23,

6+,9+,10+,11+,17+,19+,
20+,25+,32+,32+,34+,35+

1,1,2,2,3,4,4,
5,5,8,8,8,8,11,11,
12,12,15,17,22,23

a) Now, fit a distribution to the remission duration of 6-MP patients using the hazard plotting technique.
Estimate the parameter/parameters of the distribution.

CODE:

```
#9hazard plotting/estimation
# Data: Remission times for 6-MP group
remission_times_6MP <- c(6, 6, 6, 7, 10, 13, 16, 22, 23, 6, 9, 10, 11, 17, 19, 20, 25, 32, 32, 34, 35)
status_6MP <- c(rep(1, 9), rep(0, 12))  # 1 = observed, 0 = censored

# Sort remission times in increasing order
sorted_times <- sort(remission_times_6MP)

# Hazard plotting: Create ranks for plotting
ranks <- rank(sorted_times)

# Calculate the cumulative hazard H(t)
n <- length(sorted_times)
cum_hazard <- (ranks - 0.5) / n

# Plot the empirical hazard function (log-log plot)
plot(log(sorted_times), log(-log(1 - cum_hazard)),
    xlab = "log(Time (weeks))", ylab = "log(-log(1 - F(t)))",
    main = "Hazard Plot for 6-MP Patients", pch = 16)

# Estimate Weibull parameters (Shape and Scale)
# Linear regression on log-log transformed data
X <- log(sorted_times)
Y <- log(-log(1 - cum_hazard))

# Linear regression (manually without using lm() function)
n <- length(X)
x_mean <- mean(X)
y_mean <- mean(Y)

# Calculate slope (beta) and intercept (alpha)
beta <- sum((X - x_mean) * (Y - y_mean)) / sum((X - x_mean)^2)
alpha <- y_mean - beta * x_mean

# Weibull parameters:
# Shape parameter (k) is the slope
shape_weibull <- beta

# Scale parameter (lambda) can be derived from intercept
scale_weibull <- exp(-alpha / shape_weibull)
```

```
cat("Estimated Weibull Parameters: \n")
cat("Shape (k):", shape_weibull, "\n")
cat("Scale (lambda):", scale_weibull, "\n")

# Plot the fitted Weibull line
lines(X, alpha + beta * X, col = "red")
legend("bottomright", legend = c("Data", "Weibull Fit"), col = c("black", "red"), lty = 1)
```

practical 10

The following dataset is collected from a clinical trial in which researchers are testing the effectiveness of a new drug compared to a standard drug in increasing the survival time of cancer patients. Use a non-parametric method such as the Cox-Mantel test to determine if the new drug prolongs survival significantly compared to the standard drug.

New drug Standard drug
10 7
22 11
12+ 14
15+ 17
17+ 18
19+ 18
23+ 19

New drug : 10, 22, 12+ , 15+, 17+, 19+, 23+
Standard drug: 7,11,14,17,18,18,19

CODE:

```
lifetimes <- data.frame(
  time = c(10, 22, 12, 15, 17, 19, 23, 7, 11, 14, 17, 18, 18, 19),
  event = c(1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1),
  group = c(rep("A", 7), rep("B", 7))
)


n1 <- sum(lifetimes$group == "A")
n2 <- sum(lifetimes$group == "B")
n <- n1 + n2

r1 <- sum(lifetimes$group =="A" & lifetimes$event == 1)
r2 <- sum(lifetimes$group =="B" & lifetimes$event == 1)

df_cen <- lifetimes[lifetimes$event == 1, ]
time_counts <- data.frame(table(df_cen$time))
time_counts$Var1 <- as.numeric(as.character(time_counts$Var1))

df <- data.frame(time = c(), m = c(), G1 = c(), G2 = c(), R = c(), A = c())


for (i in 1:nrow(time_counts)) {
  time_point <- time_counts$Var1[i]

  G1 <- sum(lifetimes$group == "A" & lifetimes$time >= time_point)
  G2 <- sum(lifetimes$group == "B" & lifetimes$time >= time_point)
```

```
  R <- G1 + G2
  A <- G2 / R

  df_row <- data.frame(
    time = time_point,
    m = time_counts$Freq[i],
    G1 = G1,
    G2 = G2,
    R = R,
    A = A
  )
  df <- rbind(df, df_row)
}
print(df)

U <- r2 - sum(df$m * df$A)

I <- sum(df$m* (df$R - df$m)*df$A*(1-df$A)/(df$R - 1))

Z <- U/sqrt(I)
cat("Cox-Mantel Test Statistic (Z):", Z)
```

PRACTECAL 11

The following dataset is collected from a clinical trial in which researchers are testing the effectiveness of a new drug compared to a standard drug in increasing the survival time of cancer patients. Use a non-parametric method such as the Cox-Mantel test to determine if the new drug prolongs survival significantly compared to the standard drug.

New drug Standard drug
10 7
22 11
12+ 14
15+ 17
17+ 118

CODE:

```
# Input the survival times and event status
new_drug <- c(10, 22, 12, 15, 17, 19, 23)  # Survival times for the new drug
new_status <- c(1, 1, 0, 0, 0, 0, 0)       # Event status (1 = event, 0 = censored)
standard_drug <- c(7, 11, 14, 17, 18, 18, 19) # Survival times for the standard drug
standard_status <- c(1, 1, 1, 0, 0, 0, 0)    # Event status (1 = event, 0 = censored)

# Combine data for both groups
time <- c(new_drug, standard_drug)
status <- c(new_status, standard_status)
group <- c(rep("New Drug", length(new_drug)), rep("Standard Drug", length(standard_drug)))

# Create a data frame
data <- data.frame(time = time, status = status, group = group)

# Sort the data by time
```

```
data <- data[order(data$time), ]

# Initialize variables for the log-rank test
O_new <- 0   # Observed events for the new drug
E_new <- 0   # Expected events for the new drug
V_new <- 0   # Variance of observed - expected

# Loop through unique event times
unique_times <- unique(data$time)
for (t in unique_times) {
  # Number at risk in each group at time t
  at_risk_new <- sum(data$time >= t & data$group == "New Drug")
  at_risk_standard <- sum(data$time >= t & data$group == "Standard Drug")
  at_risk_total <- at_risk_new + at_risk_standard

  # Observed events at time t
  observed_new <- sum(data$time == t & data$status == 1 & data$group == "New Drug")
  observed_standard <- sum(data$time == t & data$status == 1 & data$group == "Standard Drug")
  observed_total <- observed_new + observed_standard

  # Expected events under null hypothesis
  expected_new <- (at_risk_new / at_risk_total) * observed_total
  expected_standard <- (at_risk_standard / at_risk_total) * observed_total

  # Update observed, expected, and variance
  O_new <- O_new + observed_new
  E_new <- E_new + expected_new
  V_new <- V_new + (at_risk_new / at_risk_total) *
              (1 - at_risk_new / at_risk_total) *
              (observed_total) *
              ((at_risk_total - observed_total) / (at_risk_total - 1))
}

# Calculate the log-rank test statistic
log_rank_stat <- (O_new - E_new)^2 / V_new

# Calculate p-value using chi-squared distribution
p_value <- 1 - pchisq(log_rank_stat, df = 1)

# Display results
cat("Log-Rank Test Statistic:", log_rank_stat, "\n")
cat("P-value:", p_value, "\n")

if (p_value < 0.05) {
  cat("Result: The new drug significantly prolongs survival compared to the standard drug.\n")
} else {
  cat("Result: There is no significant difference in survival between the new drug and the standard
drug.\n")
}
```

PRACTECAL-13

Question:

In a competing risks model with two causes of failure, the times to failure T1 and T2 follow exponential distributions with cause-specific hazard rates λ1 = 0.02 and λ2 = 0.03, respectively. The observed failure time T is given by T = min(T1, T2, C), where C is an independent censoring time uniformly distributed on [0, 100]. δ is event indicator.
(a) Simulate n = 1000 observations of (T, δ).
(b) Using the simulated data, estimate the cause-specific hazards λ1 and λ2.
(c) Compare the estimated hazards with the true values λ1 = 0.02 and λ2 = 0.03.

CODE:

#(a) Simulate n = 1000 observations of (T, δ).

```
# Set parameters
n <- 1000  # Number of observations
lambda1 <- 0.02  # Cause-specific hazard rate for failure 1
lambda2 <- 0.03  # Cause-specific hazard rate for failure 2
censoring_min <- 0  # Minimum censoring time
censoring_max <- 100  # Maximum censoring time

# Simulate failure times
T1 <- rexp(n, rate = lambda1)  # Exponential with rate lambda1
T2 <- rexp(n, rate = lambda2)  # Exponential with rate lambda2
C <- runif(n, min = censoring_min, max = censoring_max)  # Censoring time

# Observed failure time and event indicator
T <- pmin(T1, T2, C)  # Observed time
delta <- ifelse(T == T1, 1, ifelse(T == T2, 2, 0))  # Event indicator (1 = cause 1, 2 = cause 2, 0 =
censored)

# Create a data frame
data <- data.frame(T = T, delta = delta)
head(data)  # Display the first few rows
```

#(b) Using the simulated data, estimate the cause-specific hazards λ1 and λ2.

```
# Total time at risk for all individuals
total_time_at_risk <- sum(data$T)

# Number of events by cause
num_events_1 <- sum(data$delta == 1)
num_events_2 <- sum(data$delta == 2)

# Cause-specific hazard estimates
lambda1_hat <- num_events_1 / total_time_at_risk
lambda2_hat <- num_events_2 / total_time_at_risk

# Display results
cat("Estimated lambda1:", lambda1_hat, "\n")
```

```
cat("Estimated lambda2:", lambda2_hat, "\n")
```

#(c) Compare the estimated hazards with the true values λ1 = 0.02 and λ2 = 0.03.

```
# True values
cat("True lambda1:", lambda1, "\n")
cat("True lambda2:", lambda2, "\n")

# Compare estimated and true values
cat("Difference for lambda1:", abs(lambda1_hat - lambda1), "\n")
cat("Difference for lambda2:", abs(lambda2_hat - lambda2), "\n")
```