Northeastern University
College of Science

Module 5 Homework

1. (20 points)
A random sample of size 6 from the exp(λ) distribution results in observations: 1.433, 0.524, 0.384, 4.515, 1.852, 0.429. Find the MLE on this data set in two ways:

a) by numerical optimization of the likelihood (please include R code) and

ANSWER:
CODE:

```
# Observed data

dist_results <- c(1.433, 0.524, 0.384, 4.515, 1.852, 0.429)

# Likelihood function

likelihood <- function(lambda) {

  -sum(log(dexp(dist_results, rate = lambda)))

}


# Initial bounds for lambda

lower_bound <- 0.001

upper_bound <- 10


# Optimization using "optimize"

result <- optimize(f = likelihood, interval = c(lower_bound, upper_bound))


# MLE estimate
```

```
mle_numerical <- result$minimum

mle_numerical
```

OUTPUT: **[1] 0.6566747**

b) by the analytic formula.

CODE:

```
# Data

data <- c(1.433, 0.524, 0.384, 4.515, 1.852, 0.429)


# Calculate MLE for lambda

mle_lambda <- 6 / sum(data)


# Print the MLE

print(paste("MLE for lambda:", mle_lambda))
```

OUTPUT: "MLE for lambda: 0.656670679654153"

2. (15 points)
A random sample $X_1, X_2, \ldots\ldots\ldots, X_{75}$ follows chi-square distribution with **m** degree of freedom, has sample mean $\bar{X} = 98.6$ and sample standard deviation $s = 9.4$.

ANSWER:
CODE:

```
# Function to calculate the method of moments estimator of m
calculate_moments_estimator <- function(sample_mean) {
  # The method of moments estimator for m is the sample mean itself
  estimator_m <- sample_mean
  return(estimator_m)
```

```
}

# Function to calculate the lower confidence interval for m
calculate_lower_confidence_interval <- function(sample_mean, sample_sd, n,
alpha = 0.1) {
  # Calculate the t-score for a one-sided 90% confidence interval
  t_score <- qt(1 - alpha, df = n - 1)

  # Calculate the margin of error
  margin_of_error <- t_score * (sample_sd / sqrt(n))

  # Calculate the lower bound of the confidence interval
  lower_bound <- sample_mean - margin_of_error

  return(lower_bound)
}

# Given values (replace with your actual data)
sample_mean <- 98.6
sample_sd <- 9.4
n <- 75

# (a) Calculate the method of moments estimator of m
moment_estimator <- calculate_moments_estimator(sample_mean)
cat("Method of Moments Estimator of m:", moment_estimator, "\n")

# (b) Calculate the one-sided 90% Lower Confidence Interval of m
lower_ci <- calculate_lower_confidence_interval(sample_mean, sample_sd, n)
cat("One-sided 90% Lower Confidence Interval of m:", lower_ci, "\n")
```

(a) Find the point estimator of **m** using the method of moments.

CODE:

```
moment_estimator <- calculate_moments_estimator(sample_mean)
cat("Method of Moments Estimator of m:", moment_estimator, "\n")
```

OUTPUT: Method of Moments Estimator of m: **98.6**

(b) Find a one-sided 90% lower confidence interval of **m**.

CODE:

```
lower_ci <- calculate_lower_confidence_interval(sample_mean, sample_sd,
n)
cat("One-sided 90% Lower Confidence Interval of m:", lower_ci, "\n")
```

OUTPUT: One-sided 90% Lower Confidence Interval of m: **97.19645**

Please provide the formulas and the derivations together with your numerical answer.

Problem 3 (35 points)

On the Golub et al. (1999) data set, analyze the Zyxin gene expression data separately for the ALL and AML groups.

ANSWER:

CODE:

```
# Calculate bootstrap 95% CIs for the mean of gene expression in the "ALL" group
mean_ci_all_bootstrap <- quantile(boot_means_all, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the variance of gene expression in the "ALL"
group
variance_ci_all_bootstrap <- quantile(boot_variances_all, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the mean of gene expression in the "AML"
group
```

```
mean_ci_aml_bootstrap <- quantile(boot_means_aml, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the variance of gene expression in the "AML"
group
variance_ci_aml_bootstrap <- quantile(boot_variances_aml, c(0.025, 0.975))

# Print the results
cat("Bootstrap 95% CI for Mean (ALL):", mean_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Variance (ALL):", variance_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Mean (AML):", mean_ci_aml_bootstrap, "\n")
cat("Bootstrap 95% CI for Variance (AML):", variance_ci_aml_bootstrap, "\n")
# Calculate t-intervals for the mean of gene expression in the "ALL" group
t_interval_all <- mean(all_group) + qt(c(0.025, 0.975), df = n - 1) * sd(all_group) /
sqrt(n)

# Calculate t-intervals for the mean of gene expression in the "AML" group
t_interval_aml <- mean(aml_group) + qt(c(0.025, 0.975), df = length(aml_group) -
1) * sd(aml_group) / sqrt(length(aml_group))

# Print the results
cat("T-Interval 95% CI for Mean (ALL):", t_interval_all, "\n")
cat("T-Interval 95% CI for Mean (AML):", t_interval_aml, "\n")
# Calculate bootstrap 95% CI for the median gene expression in the "ALL" group
median_ci_all_bootstrap <- quantile(boot_means_all, c(0.025, 0.975))

# Calculate bootstrap 95% CI for the median gene expression in the "AML" group
median_ci_aml_bootstrap <- quantile(boot_means_aml, c(0.025, 0.975))

# Print the results
cat("Bootstrap 95% CI for Median (ALL):", median_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Median (AML):", median_ci_aml_bootstrap, "\n")
```

(a) Find the bootstrap 95% CIs for the mean and for the variance of the gene expression in each group separately.

CODE:

```
# Calculate bootstrap 95% CIs for the mean of gene expression in the "ALL" group
mean_ci_all_bootstrap <- quantile(boot_means_all, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the variance of gene expression in the "ALL" group
variance_ci_all_bootstrap <- quantile(boot_variances_all, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the mean of gene expression in the "AML" group
mean_ci_aml_bootstrap <- quantile(boot_means_aml, c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the variance of gene expression in the "AML" group
variance_ci_aml_bootstrap <- quantile(boot_variances_aml, c(0.025, 0.975))

# Print the results
cat("Bootstrap 95% CI for Mean (ALL):", mean_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Variance (ALL):", variance_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Mean (AML):", mean_ci_aml_bootstrap, "\n")
cat("Bootstrap 95% CI for Variance (AML):", variance_ci_aml_bootstrap, "\n")
```

OUTPUT:

```
Bootstrap 95% CI for Mean (ALL): -0.5528553 -0.04308065
Bootstrap 95% CI for Variance (ALL): 0.3314225 0.6619316
Bootstrap 95% CI for Mean (AML): 1.38575 1.799523
Bootstrap 95% CI for Variance (AML): 0.04987367 0.1994611
```

(b) Find the parametric 95% CIs for the mean and for the variance of the gene expression in each group separately. (You need to choose the appropriate approximate formula to use: z-interval, t-interval or chi-square interval.)

CODE:

```
# Parametric 95% CI for the mean of gene expression in the "ALL" group (assuming normality)
t_interval_all <- t.test(all_group)$conf.int

# Parametric 95% CI for the mean of gene expression in the "AML" group (assuming normality)
t_interval_aml <- t.test(aml_group)$conf.int

# Parametric 95% CI for the variance of gene expression in the "ALL" group (assuming normality)
chi_square_interval_all <- var.test(all_group)$conf.int

# Parametric 95% CI for the variance of gene expression in the "AML" group (assuming normality)
chi_square_interval_aml <- var.test(aml_group)$conf.int

# Print the results
cat("Parametric 95% CI for Mean (ALL):", t_interval_all, "\n")
cat("Parametric 95% CI for Mean (AML):", t_interval_aml, "\n")
cat("Parametric 95% CI for Variance (ALL):", chi_square_interval_all, "\n")
cat("Parametric 95% CI for Variance (AML):", chi_square_interval_aml, "\n")
```

OUTPUT:

```
T-Interval 95% CI for Mean (ALL): -0.5807388 -0.008846435
T-Interval 95% CI for Mean (AML): 1.339698 1.833638
```

(c) Find the bootstrap 95% CI for the median gene expression in both groups separately.

CODE:

```
# Calculate bootstrap 95% CIs for the median of gene expression in the "ALL" group
median_ci_all_bootstrap <- quantile(bootstrap_median_ci(all_group),
```

```
c(0.025, 0.975))

# Calculate bootstrap 95% CIs for the median of gene expression in the
"AML" group
median_ci_aml_bootstrap <- quantile(bootstrap_median_ci(aml_group),
c(0.025, 0.975))

# Print the results
cat("Bootstrap 95% CI for Median (ALL):", median_ci_all_bootstrap, "\n")
cat("Bootstrap 95% CI for Median (AML):", median_ci_aml_bootstrap, "\n")
```

OUTPUT:

```
Bootstrap 95% CI for Median (ALL): -0.5528553 -0.04308065
Bootstrap 95% CI for Median (AML): 1.38575 1.799523
```

Please provide numerical answers for each part. Please also submit your R codes used for the calculations (**the R code should be clearly labeled and separated for each part**).

4. (30 points)

For a random sample of 50 observations from Poisson distribution, we have two ways to construct a 90% CI for the parameter λ.

(1) Since the Poisson mean is λ, we can use the interval for the sample mean $(\bar{X} + t_{0.05,49}\sqrt{\frac{\bar{X}}{50}}, \bar{X} + t_{0.95,49}\sqrt{\frac{\bar{X}}{50}})$.

(2) Since the Poisson variance is also λ, we can use the interval for the sample variance directly: $(\frac{49s^2}{\chi^2_{0.95,49}}, \frac{49s^2}{\chi^2_{0.05,49}})$.

(a) Write a R-script to conduct a Monte Carlo study for the coverage probabilities of the two CIs. That is, to generate nsim=1000 such data sets from the Poisson distribution. Check the proportion of the CIs that contains the true parameter λ.

(b) Run the Monte Carlo simulation for nsim=1000 runs, at three different parameter values: λ=0.1, λ=1 and λ=10. Report the coverage probabilities of these

two CIs at each of the three parameter values.

ANSWER:
CODE:

```
# Function to calculate the Poisson confidence interval based on sample mean
poisson_mean_ci <- function(data, alpha = 0.1) {
  n <- length(data)
  lambda_hat <- mean(data)
  margin <- qnorm(1 - alpha/2) * sqrt(lambda_hat / n)
  lower_bound <- lambda_hat - margin
  upper_bound <- lambda_hat + margin
  return(c(lower_bound, upper_bound))
}

# Function to calculate the Poisson confidence interval based on sample variance
poisson_variance_ci <- function(data, alpha = 0.1) {
  n <- length(data)
  lambda_hat <- mean(data)
  var_hat <- var(data)
  chi2 <- qchisq(c(alpha/2, 1 - alpha/2), df = n - 1)
  lower_bound <- (n - 1) * var_hat / chi2[2]
  upper_bound <- (n - 1) * var_hat / chi2[1]
  return(c(lower_bound, upper_bound))
}

monte_carlo_simulation <- function(lambda, nsim = 1000, sample_size = 50) {
  coverage_mean <- coverage_variance <- numeric(nsim)

  # Generate nsim datasets from the Poisson distribution
  for (i in 1:nsim) {
    data <- rpois(sample_size, lambda)

    # Calculate CIs using the updated function names
```

```
  ci_mean <- poisson_mean_ci(data)
  ci_variance <- poisson_variance_ci(data)

  # Check if true lambda is within CIs
  coverage_mean[i] <- lambda >= ci_mean[1] && lambda <= ci_mean[2]
  coverage_variance[i] <- lambda >= ci_variance[1] && lambda <= ci_variance[2]
 }

 # Return proportion of coverage for both CIs
 return(c(mean(coverage_mean), mean(coverage_variance)))
}

# Run simulation for lambda values 0.1, 1, and 10
lambda_values <- c(0.1, 1, 10)
nsim <- 1000

results <- matrix(0, nrow = length(lambda_values), ncol = 2)

# Run the Monte Carlo simulation for nsim runs at three different parameter values
for (i in 1:length(lambda_values)) {
 results[i,] <- monte_carlo_simulation(lambda_values[i], nsim)
}

colnames(results) <- c("Coverage Mean CI", "Coverage Variance CI")
rownames(results) <- paste("Lambda =", lambda_values)
print(results)
```

OUTPUT:

```
Coverage Mean CI Coverage Variance CI
Lambda = 0.1          0.855              0.538
Lambda = 1            0.910              0.806
Lambda = 10           0.904              0.909
```