Assignment 1

Emre Beray Boztepe

26 10 2023

Question 1:

1-) Let X1, . . . , Xn be the simple random sample from the distribution with the density $f(x, a) = (a + 1)x^a, for x \in (0,1), a > -1$.

Option D:

Fix a = 5 and generate one random sample of the size n = 20. Calculate both estimators and the respective values of $a - a and (a - a)^2$. Which estimator is more accurate?

Load required libraries

```
library(ggplot2)
library(stringr)
library(gridExtra)
```

Set seed for reproducibility

```
set.seed(1998)
```

Maximum Likelihood Estimator (MLE) function for beta distribution

```
calculate_mle_of_beta = function(X) {
  n = length(X)
  return((-1) * n / sum(sapply(X, log)) - 1)
}
```

Moment estimator function for beta distribution

```
calculate_moment_estimator_of_beta = function(X) {
  u = mean(X)
  return((1 - 2 * u) / (u - 1))
}
```

Function to provide different estimators for beta distribution

```
provide_estimators_1 = function(a, n) {
  X = rbeta(n, a + 1, 1)
  mle = calculate_mle_of_beta(X)
  mom = calculate_moment_estimator_of_beta(X)
  return(array(c(mle, mom, mle - a, mom - a, (mle - a) ^ 2, (mom - a) ^ 2),
c(2, 3)))
}
```

```
result=provide_estimators_1(5, 20)
cat("MLE Estimator (αˆMLE): ", result[1], "\n", "Moment Estimator (αˆMoment):
", result[2], "\n",
"α - αˆMLE: ", result[3], "\n", "(α - αˆMLE)^2: ", result[5], "\n", "α -
αˆMoment: ", result[4], "\n",
"(α - αˆMoment)^2: ", result[6], "\n")
```

```
## MLE Estimator (αˆMLE):  5.909479
##  Moment Estimator (αˆMoment):  5.901606
##  α - αˆMLE:  0.9094786
##  (α - αˆMLE)^2:  0.8271513
##  α - αˆMoment:  0.9016061
##  (α - αˆMoment)^2:  0.8128936
```

Comments:

According to the results, even though they are really close to each other, MOM estimator seems to be more accurate by having values closer to 0.

Option E-F:

Generate 1000 samples of the size n = 20, for n = 200 and compare the results: i) draw histograms, box-plots and q-q plots for both estimators;

Function to calculate confidence intervals for bias

```
calculate_conf_int_bias = function(a_vec, a, m, alph = 0.05) {
  bs = a_vec - a
  b = mean(bs)
  b_sd = sd(bs)
  return (list(conf_int_lower = b - qnorm(1 - alph / 2) * b_sd / sqrt(m),
               conf_int_upper = b + qnorm(1 - alph / 2) * b_sd / sqrt(m),
               est = b))
}
```

Function to calculate confidence intervals for Mean Squared Error (MSE)

```
calculate_conf_int_mse = function(a_vec, a, m, alph = 0.05) {
  mses = (a_vec - a) ^ 2
  mse = mean(mses)
  mse_sd = sd(mses)
  return (list(conf_int_lower = mse - qnorm(1 - alph / 2) * mse_sd / sqrt(m),
               conf_int_upper = mse + qnorm(1 - alph / 2) * mse_sd / sqrt(m),
               est = mse))
}
```

Function to calculate confidence intervals for variance

```
calculate_conf_int_var = function(a_vec, a, m, alph = 0.05) {
  v = var(a_vec)
  return (list(conf_int_lower = (m - 1) * v / qchisq(1 - alph / 2, m - 1),
               conf_int_upper = (m - 1) * v / qchisq(alph / 2, m - 1),
```

```
                est = v))
}
```

Simulation function for different sample sizes and number of iterations

```
run_simulation_1 = function(a, n, m) {
  replicate(m, provide_estimators_1(a, n))
}
```

Run simulations for sample sizes n=20 and n=200

```
sim_1_20 = run_simulation_1(5, 20,  1000)
sim_1_200 = run_simulation_1(5, 200,  1000)
```

Convert simulation results to data frames

```
estimators_20 = as.data.frame.array(t(sim_1_20[,  1, ]))
colnames(estimators_20) = c("mle",  "mom")
estimators_200 = as.data.frame.array(t(sim_1_200[,  1, ]))
colnames(estimators_200) = c("mle",  "mom")
```

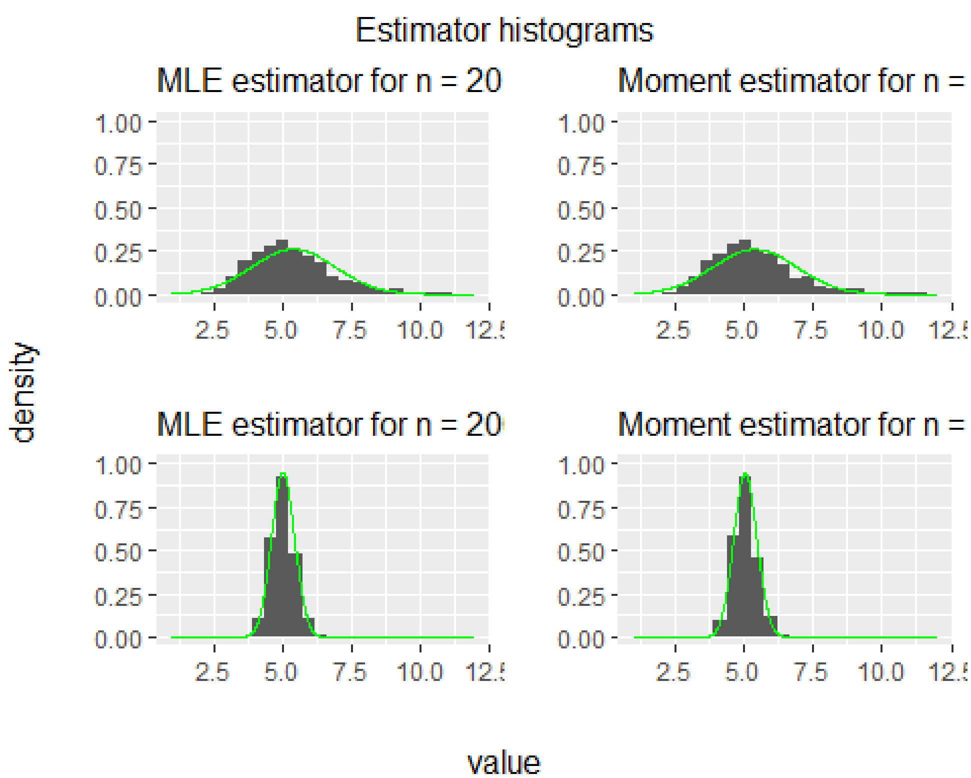
Plot histograms for MLE and moment estimator for n=20 and n=200

```
p1 = ggplot(estimators_20,  aes(x=mle)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  stat_function(fun = dnorm,
                args = list(mean = mean(estimators_20[["mle"]]),
                            sd = sd(estimators_20[["mle"]])), n=1000,
color="green") +
  xlim(1, 12) + ylim(0, 1) +
  labs(title="MLE estimator for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

p2 = ggplot(estimators_20,  aes(x=mom)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  stat_function(fun = dnorm,
                args = list(mean = mean(estimators_20[["mom"]]),
                            sd = sd(estimators_20[["mom"]])), n=1000,
color="green") +
  xlim(1, 12) + ylim(0, 1) +
  labs(title="Moment estimator for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

p3 = ggplot(estimators_200,  aes(x=mle)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  stat_function(fun = dnorm,
                args = list(mean = mean(estimators_200[["mle"]]),
                            sd = sd(estimators_200[["mle"]])), n=1000,
color="green") +
  xlim(1, 12) + ylim(0, 1) +
  labs(title="MLE estimator for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
```

```
p4 = ggplot(estimators_200,  aes(x=mom)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  stat_function(fun = dnorm,
                args = list(mean = mean(estimators_200[["mom"]]),
                            sd = sd(estimators_200[["mom"]])), n=1000,
color="green") +
  xlim(1, 12)  + ylim(0, 1) +
  labs(title="Moment estimator for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
```

Arrange histograms in a grid

```
grid.arrange(p1, p2, p3, p4, ncol=2,
             left="density", bottom="value", top="Estimator histograms")
```



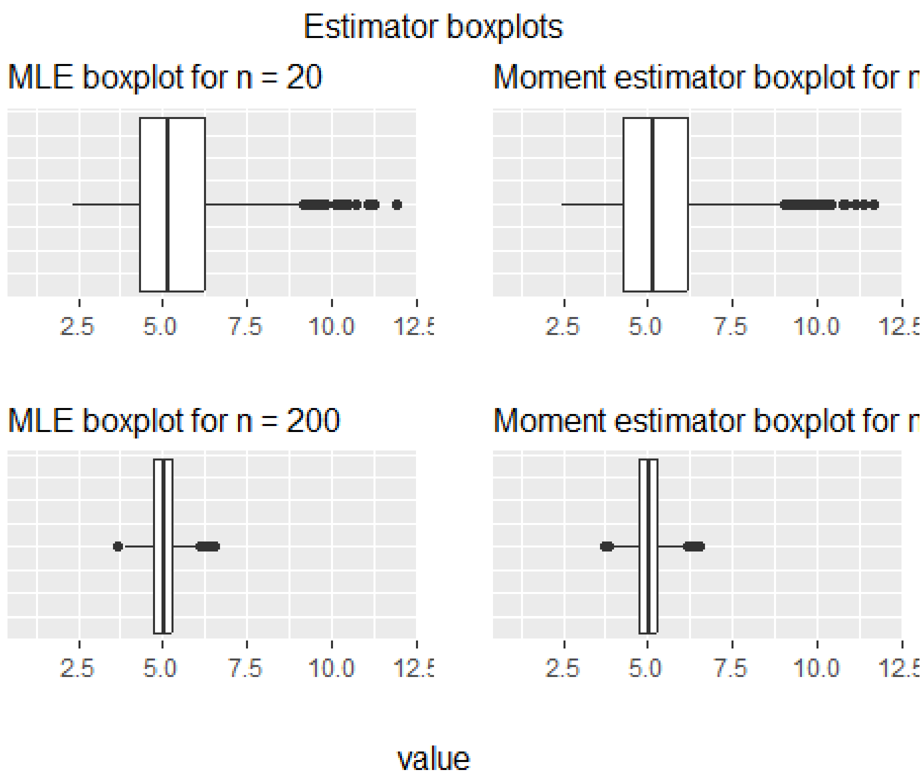
Boxplots for MLE and moment estimator for n=20 and n=200

```
p1 = ggplot(estimators_20,  aes(x=mle)) + geom_boxplot() +
  xlim(1, 12) +
  labs(title="MLE boxplot for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12),
        axis.text.y = element_blank(), axis.ticks.y = element_blank())
p2 = ggplot(estimators_20,  aes(x=mom)) + geom_boxplot() +
  xlim(1, 12) +
  labs(title="Moment estimator boxplot for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12),
        axis.text.y = element_blank(), axis.ticks.y = element_blank())
```

```r
p3 = ggplot(estimators_200,  aes(x=mle)) + geom_boxplot() +
  xlim(1, 12) +
  labs(title="MLE boxplot for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12),
        axis.text.y = element_blank(), axis.ticks.y = element_blank())
p4 = ggplot(estimators_200,  aes(x=mom)) + geom_boxplot() +
  xlim(1, 12) +
  labs(title="Moment estimator boxplot for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12),
        axis.text.y = element_blank(), axis.ticks.y = element_blank())
```

Arrange boxplots in a grid

```r
grid.arrange(p1, p2, p3, p4, ncol=2,
             bottom="value", top="Estimator boxplots")
```



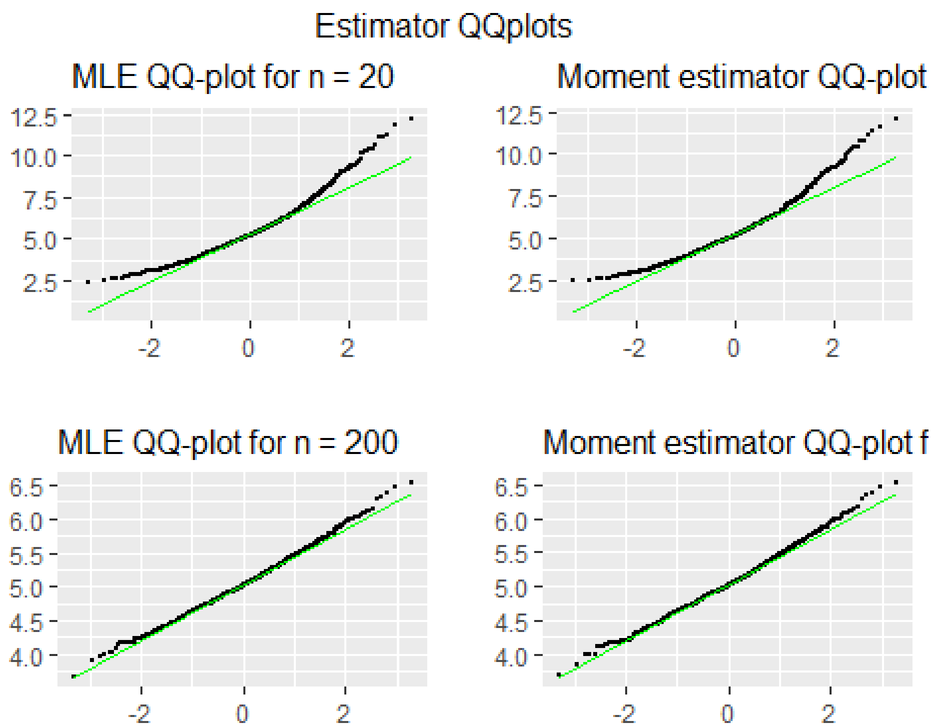
QQ-plots for MLE and moment estimator for n=20 and n=200

```r
par(mfrow=c(2,2))
p1 = ggplot(estimators_20,  aes(sample=mle)) + geom_qq(size=0.1) +
  stat_qq_line(color="green") +
  labs(title="MLE QQ-plot for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
p2 = ggplot(estimators_20,  aes(sample=mom)) + geom_qq(size=0.1) +
  stat_qq_line(color="green") +
  labs(title="Moment estimator QQ-plot for n = 20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
p3 = ggplot(estimators_200,  aes(sample=mle)) + geom_qq(size=0.1) +
```

```
  stat_qq_line(color="green") +
  labs(title="MLE QQ-plot for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
p4 = ggplot(estimators_200,  aes(sample=mom)) + geom_qq(size=0.1) +
  stat_qq_line(color="green") +
  labs(title="Moment estimator QQ-plot for n = 200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))
```

Arrange QQ-plots in a grid

```
grid.arrange(p1, p2, p3, p4, ncol=2,
             top="Estimator QQplots")
```



ii) estimate the bias, the variance and the mean-squared error of both estimators and construct approximate 95% confidence intervals for these parameters. In case of MLE compare the values of these parameters to the values provided by the asymptotic distribution of a_{MLE}.

Bias, Variance, and MSE analysis for n=20 and n=200 Bias analysis

```
cat("**Bias For n = 20: **\n\n")

## **Bias For n = 20: **

int_b20_mle = calculate_conf_int_bias(estimators_20[["mle"]], 5, 1000)
cat(str_c("Estimated value of bias for MLE:   ", round(int_b20_mle[["est"]],
3), "\n\n"))
```

```
## Estimated value of bias for MLE:   0.405

cat(str_c("Confidence intervals for MLE: (",
          round(int_b20_mle[["conf_int_lower"]], 3),  ", ",
          round(int_b20_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (0.311, 0.5)

int_b20_mom = calculate_conf_int_bias(estimators_20[["mom"]], 5, 1000)
cat(str_c("Estimated value of bias for moment estimator:   ",
round(int_b20_mom[["est"]], 3), "\n\n"))

## Estimated value of bias for moment estimator:   0.368

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_b20_mom[["conf_int_lower"]], 3),  ", ",
          round(int_b20_mom[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for moment estimator: (0.274, 0.463)

cat("**For n = 200: **\n\n")

## **For n = 200: **

int_b200_mle = calculate_conf_int_bias(estimators_200[["mle"]], 5, 1000)
cat(str_c("Estimated value of bias for MLE:   ", round(int_b200_mle[["est"]],
3), "\n\n"))

## Estimated value of bias for MLE:   0.032

cat(str_c("Confidence intervals for MLE: (",
          round(int_b200_mle[["conf_int_lower"]], 3),  ", ",
          round(int_b200_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (0.006, 0.058)

int_b200_mom = calculate_conf_int_bias(estimators_200[["mom"]], 5, 1000)
cat(str_c("Estimated value of bias for moment estimator:   ",
round(int_b200_mom[["est"]], 3), "\n\n"))

## Estimated value of bias for moment estimator:   0.027

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_b200_mom[["conf_int_lower"]], 3),  ", ",
          round(int_b200_mom[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for moment estimator: (0.001, 0.054)
```

Variance analysis

```
cat("**Variance For n = 20: **\n\n")

## **Variance For n = 20: **
```

```r
int_v20_mle = calculate_conf_int_var(estimators_20[["mle"]], 5, 1000)
cat(str_c("Estimated value of variance for MLE:   ",
round(int_v20_mle[["est"]], 3), "\n\n"))

## Estimated value of variance for MLE:   2.329

cat(str_c("Confidence intervals for MLE: (",
          round(int_v20_mle[["conf_int_lower"]], 3),  ", ",
          round(int_v20_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (2.137, 2.547)

int_v20_mom = calculate_conf_int_var(estimators_20[["mom"]], 5, 1000)
cat(str_c("Estimated value of variance for moment estimator:   ",
round(int_v20_mom[["est"]], 3), "\n\n"))

## Estimated value of variance for moment estimator:   2.347

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_v20_mom[["conf_int_lower"]], 3),  ", ",
          round(int_v20_mom[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for moment estimator: (2.154, 2.567)

cat("**For n = 200: **\n\n")

## **For n = 200: **

int_v200_mle = calculate_conf_int_var(estimators_200[["mle"]], 5, 1000)
cat(str_c("Estimated value of variance for MLE:   ",
round(int_v200_mle[["est"]], 3), "\n\n"))

## Estimated value of variance for MLE:   0.176

cat(str_c("Confidence intervals for MLE: (",
          round(int_v200_mle[["conf_int_lower"]], 3),  ", ",
          round(int_v200_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (0.161, 0.192)

int_v200_mom = calculate_conf_int_var(estimators_200[["mom"]], 5, 1000)
cat(str_c("Estimated value of variance for moment estimator:   ",
round(int_v200_mom[["est"]], 3), "\n\n"))

## Estimated value of variance for moment estimator:   0.178

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_v200_mom[["conf_int_lower"]], 3),  ", ",
          round(int_v200_mom[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for moment estimator: (0.164, 0.195)
```

MSE analysis

```r
cat("**MSE For n = 20: **\n\n")

## **MSE For n = 20: **

int_m20_mle = calculate_conf_int_mse(estimators_20[["mle"]], 5, 1000)
cat(str_c("Estimated value of MSE for MLE:   ", round(int_m20_mle[["est"]],
3), "\n\n"))

## Estimated value of MSE for MLE:   2.491

cat(str_c("Confidence intervals for MLE: (",
          round(int_m20_mle[["conf_int_lower"]], 3),  ", ",
          round(int_m20_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (2.181, 2.801)

int_m20_mom = calculate_conf_int_mse(estimators_20[["mom"]], 5, 1000)
cat(str_c("Estimated value of MSE for moment estimator:   ",
round(int_m20_mom[["est"]], 3), "\n\n"))

## Estimated value of MSE for moment estimator:   2.481

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_m20_mom[["conf_int_lower"]], 3),  ", ",
          round(int_m20_mom[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for moment estimator: (2.176, 2.785)

cat("**For n = 200: **\n\n")

## **For n = 200: **

int_m200_mle = calculate_conf_int_mse(estimators_200[["mle"]], 5, 1000)
cat(str_c("Estimated value of MSE for MLE:   ", round(int_m200_mle[["est"]],
3), "\n\n"))

## Estimated value of MSE for MLE:   0.177

cat(str_c("Confidence intervals for MLE: (",
          round(int_m200_mle[["conf_int_lower"]], 3),  ", ",
          round(int_m200_mle[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals for MLE: (0.16, 0.193)

int_m200_mom = calculate_conf_int_mse(estimators_200[["mom"]], 5, 1000)
cat(str_c("Estimated value of MSE for moment estimator:   ",
round(int_m200_mom[["est"]], 3), "\n\n"))

## Estimated value of MSE for moment estimator:   0.179

cat(str_c("Confidence intervals for moment estimator: (",
          round(int_m200_mom[["conf_int_lower"]], 3),  ", ",
          round(int_m200_mom[["conf_int_upper"]], 3), ")\n\n"))
```

```
## Confidence intervals for moment estimator: (0.162, 0.195)
```

Comments:

As we would expect, the bias converges to zero with n. For the bigger sample size, both estimators have much better results.

Both MLE and moment estimator have very similar results. For both sample sizes, the MLE has slightly greater bias and smaller variance. To combine those insights, we can look at the MSE. As we can see, the MLE performs a little better. Both estimators have MSE greater then expected for n=20 and MSE rougly equal to expected value for n=200.

Resuming: for most cases, MLE is the best choice. If we want to ensure the smallest bias, we can decide to use the moment estimator. Both estimator behave very similar, so the moment estimator is a good choice, when the MLE can by analitycaly derived.

Question 2:

Let X1, . . . , Xn be the simple random sample from the distribution with the density f(x, λ) = $\lambda e^{-}\lambda x$, for x > 0, λ > 0. Find the uniformly most powerful test at the level α = 0.05 for testing the hypothesis H0 : λ = 5 against H1 : λ = 3.

Option C:

Provide the formula for the p-value for a given random sample. For n = 20 generate one random sample from H0 and one random sample from H1 and the respective p-values. What conclusions can be drawn based on the p-values?

Load necessary libraries

```r
library(ggplot2)
library(stringr)
library(gridExtra)

# Set seed for reproducibility
set.seed(1998)
```

Function to provide estimators

```r
calculate_p_values = function(n, lambda) {
  X = rexp(n, lambda)
  T_stat = sum(X)
  p_value = 1 - pgamma(T_stat, n, 5)
  return(p_value)
}
```

Function to calculate confidence intervals

```r
calculate_conf_int = function(p, alph) {
  p_est = mean(p < alph)
  return(list(est = p_est,
```

```
               conf_int_lower = p_est - qnorm(1 - alph / 2) * sqrt(p_est * (1
- p_est) / length(p)),
               conf_int_upper = p_est + qnorm(1 - alph / 2) * sqrt(p_est * (1
- p_est) / length(p))))
}

h0_p_value = calculate_p_values(20, 5)
h1_p_value = calculate_p_values(20, 3)

cat("p_value for H0: ", h0_p_value, "p_value for H1: ", h1_p_value)

## p_value for H0:  0.05717863 p_value for H1:  3.48551e-05
```

Comments: According to the results, it can be said that p_value for H0 is closer to alpha

Option E-F-G:

Generate 1000 samples of the size n = 20, 200 from H0, H1 and calculate respective p-values and compare them. i) Compare the distribution of these p-values to the distribution derived in d): draw a histogram and a respective q-q plot.

ii) Use these simulations to construct the 95% condence interval for the type I error of the test.

iii) Compare the distribution of p-values under H0 and under H1.

iv) Use these simulations to construct the 95% condence interval for the power of the test. Compare with the theoretically calculated power.

Function for simulation

```
run_simulation = function(lambda, n, m) {
  return(replicate(m, calculate_p_values(n, lambda)))
}
```

Run simulations for both null and alternative hypotheses

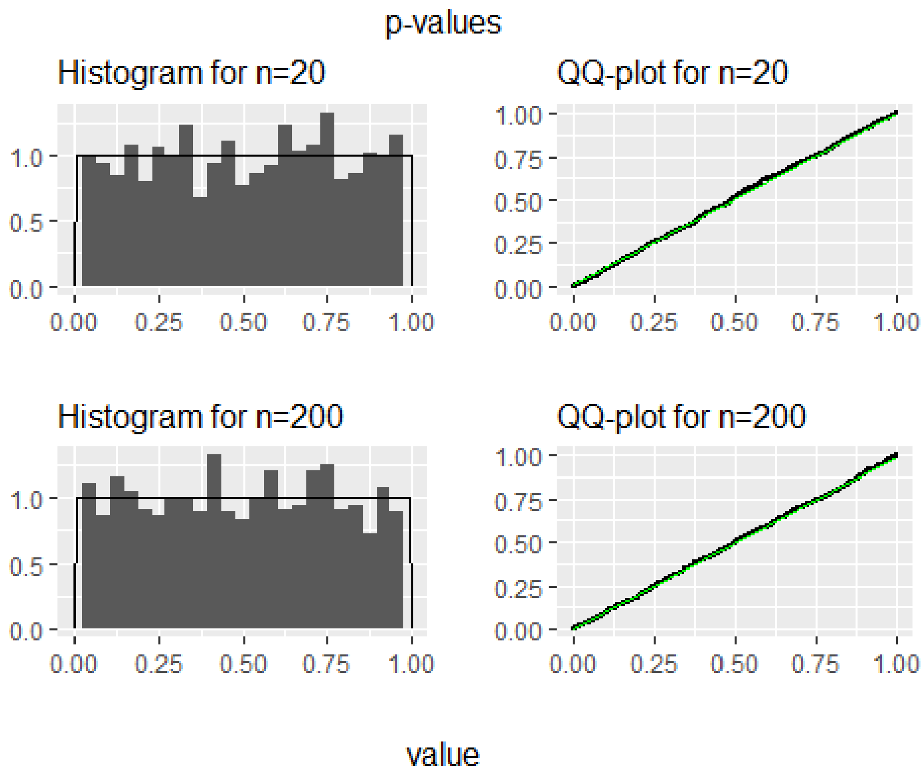
```
sim_h0_20 = run_simulation(5, 20, 1000)
sim_h0_200 = run_simulation(5, 200, 1000)
sim_h1_20 = run_simulation(3, 20, 1000)
sim_h1_200 = run_simulation(3, 200, 1000)
```

Convert simulation results to data frames

```
sim_h0_20_df = as.data.frame.numeric(t(t(sim_h0_20)))
colnames(sim_h0_20_df) = c("p_value")

sim_h0_200_df = as.data.frame.numeric(t(t(sim_h0_200)))
colnames(sim_h0_200_df) = c("p_value")
```

Plot histograms and QQ-plots for null hypothesis (n=20 and n=200)

```
p1 = ggplot(sim_h0_20_df,  aes(x=p_value)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  labs(title="Histogram for n=20") + xlab("") + ylab("") +
  stat_function(fun = dunif, args = list(min =
min(sim_h0_20_df[["p_value"]]),
                                         max =
max(sim_h0_20_df[["p_value"]])), n=1000) +
  xlim(0, 1) +
  theme(plot.title = element_text(size=12))

p2 = ggplot(sim_h0_20_df,  aes(sample=p_value)) +
  geom_qq(distribution = stats::qunif, size=0.5) +
  labs(title="QQ-plot for n=20") + xlab("") + ylab("") +
  stat_qq_line(distribution = stats::qunif, color="green") +
  theme(plot.title = element_text(size=12))

p3 = ggplot(sim_h0_200_df,  aes(x=p_value)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  labs(title="Histogram for n=200") + xlab("") + ylab("") +
  stat_function(fun = dunif, args = list(min =
min(sim_h0_200_df[["p_value"]]),
                                         max =
max(sim_h0_200_df[["p_value"]])), n=1000) +
  xlim(0, 1) +
  theme(plot.title = element_text(size=12))

p4 = ggplot(sim_h0_200_df,  aes(sample=p_value)) +
  geom_qq(distribution = stats::qunif, size=0.5) +
  labs(title="QQ-plot for n=200") + xlab("") + ylab("") +
  stat_qq_line(distribution = stats::qunif, color="green") +
  theme(plot.title = element_text(size=12))

grid.arrange(p1, p2, p3, p4, ncol=2,
             bottom="value", top="p-values")
```



Confidence interval analysis for null hypothesis

```
cat("**For n = 20: **\n\n")

## **For n = 20: **

int_b20 = calculate_conf_int(sim_h0_20, 0.05)
cat(str_c("Estimated value of Type I Error:   ", int_b20$est, "\n\n"))

## Estimated value of Type I Error:   0.06

cat(str_c("Confidence intervals: (",
          round(int_b20[["conf_int_lower"]], 3),  ", ",
          round(int_b20[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals: (0.045, 0.075)

cat("**For n = 200: **\n\n")

## **For n = 200: **

int_b200 = calculate_conf_int(sim_h0_200, 0.05)
cat(str_c("Estimated value of Type I Error:   ", int_b200$est, "\n\n"))

## Estimated value of Type I Error:   0.055

cat(str_c("Confidence intervals: (",
          round(int_b200[["conf_int_lower"]], 3),  ", ",
          round(int_b200[["conf_int_upper"]], 3), ")\n\n"))
```

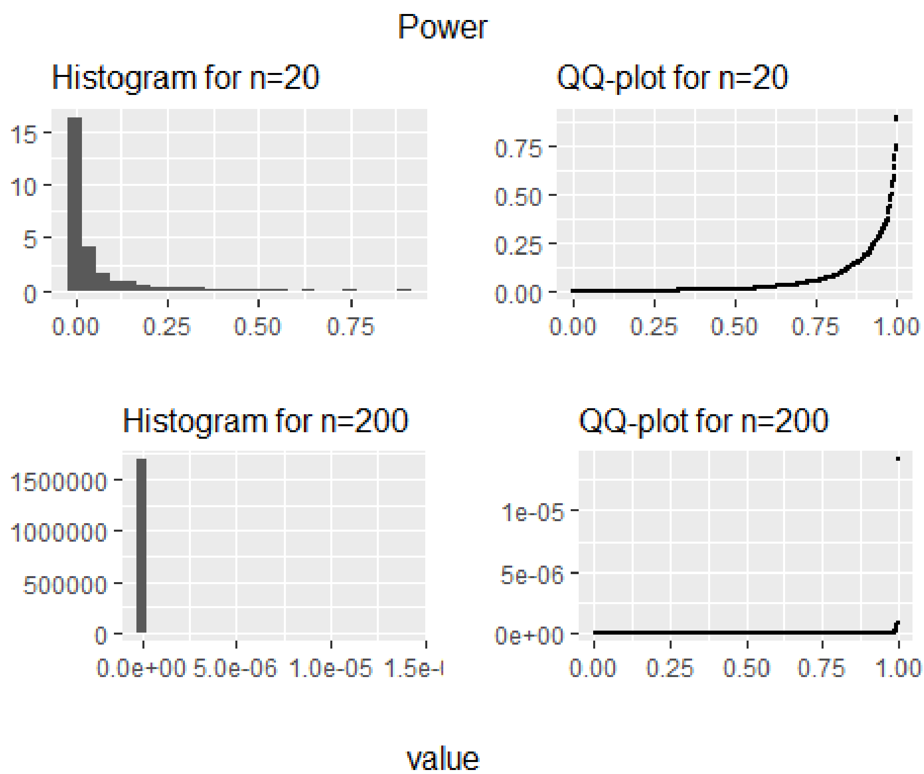
```
## Confidence intervals: (0.041, 0.069)
```

Power analysis for alternative hypothesis

```r
sim_h1_20_df = as.data.frame.numeric(t(t(sim_h1_20)))
colnames(sim_h1_20_df) = c("power")

sim_h1_200_df = as.data.frame.numeric(t(t(sim_h1_200)))
colnames(sim_h1_200_df) = c("power")
```

Plot histograms and QQ-plots for alternative hypothesis (n=20 and n=200)

```r
p1 = ggplot(sim_h1_20_df,  aes(x=power)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  labs(title="Histogram for n=20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

p2 = ggplot(sim_h1_20_df, aes(sample=power)) +
  geom_qq(distribution = stats::qunif, size=0.5) +
  labs(title="QQ-plot for n=20") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

p3 = ggplot(sim_h1_200_df,  aes(x=power)) +
  geom_histogram(aes(y=..density..), bins=25)  +
  labs(title="Histogram for n=200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

p4 = ggplot(sim_h1_200_df,  aes(sample=power)) +
  geom_qq(distribution = stats::qunif, size=0.5) +
  labs(title="QQ-plot for n=200") + xlab("") + ylab("") +
  theme(plot.title = element_text(size=12))

grid.arrange(p1, p2, p3, p4, ncol=2,
             bottom="value", top="Power")
```



Confidence interval analysis for alternative hypothesis

```
cat("**For n = 20: **\n\n")

## **For n = 20: **

int_b20 = calculate_conf_int(sim_h1_20, 0.05)
cat(str_c("Estimated value of power:   ", int_b20$est, "\n\n"))

## Estimated value of power:   0.75

cat(str_c("Confidence intervals: (",
          round(int_b20[["conf_int_lower"]], 3),  ", ",
          round(int_b20[["conf_int_upper"]], 3), ")\n\n"))

## Confidence intervals: (0.723, 0.777)

cat("**For n = 200: **\n\n")

## **For n = 200: **

int_b200 = calculate_conf_int(sim_h1_200, 0.05)
cat(str_c("Estimated value of power:   ", int_b200$est, "\n\n"))

## Estimated value of power:   1

cat(str_c("Confidence intervals: (",
          round(int_b200[["conf_int_lower"]], 3),  ", ",
          round(int_b200[["conf_int_upper"]], 3), ")\n\n"))
```

```
## Confidence intervals: (1, 1)
```

Comments:

The results of the simulations match the theoretical expectations. The p-values are uniformally distributed under H_0. The p-values under H_0 are close to α. The power is not uniformly distributed under H_1, the mass is concentrated around 0. The value of power is close to the theoretically derived values. The bigger sample is considered, the better tests one can make.

1-) $X_1, \dots, X_n \overset{iid}{\sim} f(x,\alpha) = (\alpha+1)\, x^{\alpha}, \quad x \in (0,1), \ \alpha > -1$

$f(x,\alpha) \to$ pdf of $\beta(\alpha+1, 1)$

a-) $\hat{\alpha}_{MLE} = ?$

$$\ell(\alpha, \underline{X}) = \sum_{i=1}^{n} \log(f(x_i,\alpha)) = \sum_{i=1}^{n} \log((\alpha+1)x_i^{\alpha})$$

$$= \sum_{i=1}^{n} \log(\alpha+1) + \alpha \log(x_i) = n\log(\alpha+1) + \alpha \sum_{i=1}^{n} \log x_i$$

$$\frac{\partial \ell(\alpha, \underline{X})}{\partial \alpha} = \frac{n}{\alpha+1} + \sum_{i=1}^{n} \log x_i = 0$$

$$\frac{\alpha+1}{n} = \frac{-1}{\sum_{i=1}^{n} \log x_i} \implies \boxed{\hat{\alpha}_{MLE} = -\frac{n}{\sum_{i=1}^{n} \log x_i} - 1}$$

$\ell(\hat{\alpha}_{MLE}, \underline{X}) = \max$ of function $\ell(\alpha, \underline{X})$:

$$\frac{\partial^2 \ell(\hat{\alpha}_{MLE}, \underline{X})}{\partial \alpha^2} = \frac{-n}{\alpha+1} < 0$$

b-) Fisher Information $(I) = ?$

$$I(\alpha) = E\left[\frac{\partial \log f}{\partial \alpha}\right]^2 = -E\left[\frac{\partial^2 \log f(x,\alpha)}{\partial \alpha^2}\right]$$

$$= -E\left[\frac{-1}{(\alpha+1)^2}\right] = \frac{1}{(\alpha+1)^2}$$

We know that: $\sqrt{n}(\hat{\alpha}_{MLE} - \alpha) \xrightarrow{\infty} N\left(0, \frac{1}{I(\alpha)} = (\alpha+1)^2\right)$

$$\implies \frac{\sqrt{n}(\hat{\alpha}_{MLE} - \alpha)}{\sqrt{n}} + \alpha \xrightarrow{\infty} N\left(\alpha, \frac{(\alpha+1)^2}{n}\right)$$

$$Var[\hat{\alpha}_{MLE}] = Var\left[\frac{\sqrt{n}(\hat{\alpha}_{MLE} - \alpha)}{\sqrt{n}} + \underset{const}{\alpha}\right] = \frac{1}{n} Var\left[\sqrt{n}(\hat{\alpha}_{MLE} - \alpha)\right] = \frac{(\alpha+1)^2}{n}$$

$$E[\hat{\alpha}_{MLE}] = E\left[\frac{\sqrt{n}(\hat{\alpha}_{MLE} - \alpha)}{\sqrt{n}} + \alpha\right] = \frac{1}{n} E\left[\sqrt{n}(\hat{\alpha}_{MLE} - \alpha)\right] + E[\alpha] = 0 + \alpha = \alpha$$

We have $\hat{\alpha}_{MLE} \overset{asymptotically}{\sim} N\left(\alpha, \frac{(\alpha+1)^2}{n}\right)$

b-2-) $MSE(\hat{\alpha}_{MLE}) = \mathbb{E}_\alpha \left[(\hat{\alpha}_{MLE} - \alpha)^2 \right] = \boxed{Var(\hat{\alpha}_{MLE}) = \frac{(\alpha+1)^2}{n}}$

c-) Moment Estimator $\hat{\alpha}_{MOM} = ?$

$X_1, \dots, X_n \sim f(x,\alpha) = (\alpha+1)x^\alpha$, $x \in (0,1)$, $\alpha > -1$

$$\mu_1 = EX_1 = \int_0^1 x\, f(x,\alpha)\,dx = \int_0^1 (\alpha+1)x^{\alpha+1}\,dx = \frac{\alpha+1}{\alpha+2}$$

$$(\alpha+2)\mu_1 = \alpha+1$$
$$\alpha\mu_1 - \alpha = -2\mu_1 + 1$$
$$\alpha = \frac{1-2\mu_1}{\mu_1 - 1}$$
$$\boxed{\hat{\alpha}_{MOM} = \frac{1-2\bar{X}}{\bar{X}-1}}$$

2-) $X_1, \dots, X_n \overset{i.i.d}{\sim} f(x,\alpha) = \lambda e^{-\lambda x}$, $x>0$, $\lambda > 0$, $x_i > 0$

$H_0: \lambda = 5$, $H_1: \lambda = 3$, $\alpha = 0.05$

a-) Neyman Pearson:

$$k < \frac{\prod_{i=1}^n f_1(x_i)}{\prod_{i=1}^n f_0(x_i)} = \frac{\prod_{i=1}^n 3e^{-3x_i}}{\prod_{i=1}^n 5e^{-5x_i}} = \frac{3^n e^{-3n\sum x_i}}{5^n e^{-5n\sum x_i}}$$

$$= \left(\frac{3}{5}\right)^n \exp\{-3\sum x_i + 5\sum x_i\} = \left(\frac{3}{5}\right)^n \exp\{2\sum x_i\}$$

$$C_1 := \frac{1}{2}\log\left[\left(\frac{5}{3}\right)^n k\right] < \sum x_i \quad , \quad \text{Critical area: } C = \{X : \sum_{i=1}^n x_i > C_1\}$$

$$0.05 = \alpha = P_0(X \in C) = P\{\sum_{i=1}^n x_i > C_1\}$$

$$\Downarrow$$

$$C_1 = F^{-1}_{\Gamma(n, \frac{1}{\lambda} = \frac{1}{5})}(0.95)$$

$X_i \sim exp(\lambda) \overset{D}{=} \Gamma(1, \frac{1}{\lambda})$

$\sum x_i \sim \Gamma(n, \frac{1}{\lambda})$

$\Gamma(n, \frac{1}{\lambda}) = \frac{\lambda^n}{\Gamma(n)} x^{n-1} e^{-\lambda x}$

b-) Power = ?

$$\gamma = P_1(\underline{X} \in C) = P_1\Big(\underbrace{\sum_{i=1}^{n} X_i}_{\sim \Gamma(n, \cancel{\frac{1}{\lambda}} = \frac{1}{3})} > C_1\Big) = 1 - F_{\Gamma(n,\frac{1}{3})}\left(F^{-1}_{\Gamma(n,\frac{1}{5})}(0.95)\right)$$

c-) p-value = ?

$$\text{p-value} = P_0\Big(\underbrace{G}_{\sim \Gamma(n,\frac{1}{5})} > \sum X_i\Big) = 1 - F_{\Gamma(n,1/5)}\left(\sum_{i=1}^{n} X_i\right)$$

d-) When data come from H_0, it means that null hypothesis is true. Since we want the probability of rejecting null hypothesis to be alpha (in our case 0.05). We reject H_0 when p-value is lower than alpha. This is only possible if p-value comes from a uniform distribution. So, the distribution of an invertible CDF of a random variable is

$$\Rightarrow U[0,1]$$