MATH 185, HW1

Victor Pekkari, epekkari@ucsd.edu

January 23, 2025

Problem 1

(a)

$$E(X) = 0.3 \cdot 1 + 0.5 \cdot 2 + 0.15 \cdot 3 + 0.05 \cdot 4 = 1.95$$

Answer: **1.95**

(b)

Answer: **80%**

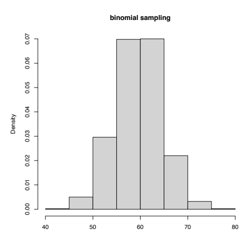
Problem 2

Approximating a Binomial distribution as a Normal-distribution works well when (*) holds

$$0 < p - 3 \cdot \sqrt{\frac{p(1-p)}{n}} < p + 3 \cdot \sqrt{\frac{p(1-p)}{n}} \tag{*}$$

(a)

$$p = 0.6, n = 100$$
$$0 < 0.453... < 0.747... < 1$$

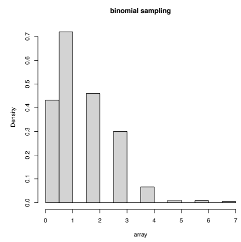


Answer: It look like a normal distribution and makes sense since **(*)** holds

(b)

$$p = 0.015, n = 100$$

$$0 > -0.0215... \qquad \text{(this means (*) doesn't hold)}$$



Answer: It doesn't look like a normal distribution and makes sense since **(*)** doesn't hold

Problem 3

(a) Comment:

My result was **0.949** which is very close to the nominal coverage of 95%. The value 0.949 represents the proportions of simulations in which the variance for the sample was in the confidence interval.

The confidence interval is an interval that contains the parameter being estimated x% of the time. We chose a confidence level of 95% which means the variance of our samples should be within our confidence interval 95% of the times. **The nominal coverage** is the theoretical (intended) confidence interval which is specified before the analysis. The confidence level of 95% in our example gives us a nominal coverage of 95%, which means that in theory 95% of the confidence intervals constructed from repeated random samplings should contain the true parameter.

(b)

(N(1,1), n=10) - Sampling: 95%
(exp(1), n=100) - Sampling: 68%
(exp(1), n=10) - Sampling: 76%

Conclusion: This makes sense, since increasing the n, decreases the width of the confidence interval, which makes a higher percentage of the samples fall within the confidence interval.

One thing I don't really understand is that I thought the normalized exponential distribution should converge closer to the normalized normal distribution as the sample size increased, and therefore behave more like it, but from the results it seems like the exponential with fewer samples behaves more like the normalized normal distribution.

(c)

Probability of type I error is **5.3%**, which is close to 5% which it should theoretically be.

(d)

Type I error is rejected for samplings ___, with probability ___

(N(1,1), n=100) - Sampling: 5.4%
(exp(1), n=100) - Sampling: 5%
(exp(1), n=10) - Sampling: 5.1%

Conclusion: We only wanna reject the correct distribution 5% of the time, so rejecting the normal and the exponential distribution around 5% of the time is very reasonable.

1 Code for the homework:

Listing 1: R Code

```

#1c)
simulation <- function(n){
  return(sample(1:4, size=n, prob = c(0.3, 0.5, 0.15, 0.05), replace
     =TRUE))
}
#1d)
estimate_mean <- function(n) {
  return(mean((simulation(n))))
}
#1e)
estimate_prob <- function(n, p){
  array <- simulation(n)
  filtered <- array[array <= p]
  return(length(filtered) / n)
}
#---------------------------

#2a)
plot_binom <- function(n, p, file_name){
  array <- rbinom(1000, n, p)
  pdf(file_name)
  hist(array, freq = FALSE, main = "binomial sampling",
       plot = TRUE, border = "black")
```

```
  dev.off()
}
#---------------------------

#3a)
hypothesis_test <- function(){
  n <- 100       # Sample size
  mu <- 1        # Mean
  sigma2 <- 1    # Variance
  alpha <- 0.05 # Significance level
  sim_count <- 1000 # Number of simulations
  coverage_count <- 0
  for (i in 1:sim_count){
    #sampling <- rnorm(n, mu, sigma2) #q3a,b,c)
    sampling <- rexp(n, 1)
    s2 <- var(sampling)
    #Chi-squared critical values
    lower_bound <- (n - 1) * s2 / qchisq(1 - alpha / 2, df = n - 1)
    upper_bound <- (n - 1) * s2 / qchisq(alpha / 2, df = n - 1)

    if (sigma2 >= lower_bound && sigma2 <= upper_bound) {
      coverage_count <- coverage_count + 1
    }
  }
  result <- (coverage_count / sim_count)
  print(result)
  return(result)
}

perform_nulltest <- function(sample){
  n <- length(sample)
  s2 <- var(sample)  # Sample variance
  sigma2_0 <- 1
  alpha <- 0.05
  test_result <- (n - 1) * s2 / sigma2_0
  lower <- qchisq(alpha / 2, df = n - 1)
  upper <- qchisq(1 - (alpha / 2), df = n - 1)
  return(lower > test_result || test_result > upper)
}

null_test <- function() {
  num_simulations <- 100000
  type1_error_count <- 0
  sigma2 <- 1
  mu <- 1
  n <- 100
  mean <- 1
```

```
  for (i in 1:num_simulations) {
    sample <- rnorm(n, mean = mu, sd = sqrt(sigma2))
    #sample <- rexp(n, 1)
    mean <- mean + var(sample)
    #print(paste("sd: ", sd(sample)))
    if (perform_nulltest(sample)) {
      type1_error_count <- type1_error_count + 1
    }
  }
  print(mean / num_simulations)
  print(type1_error_count / num_simulations)
}
#3b, N(1,1) -> 0.946
#3b n=100 rexp(100,1) -> 0.681
#3b n=10 rexp(10,1) -> 0.758
# Parameters
#hypothesis_test()

null_test()

#print(plot_binom(100, 0.015, "binomial_2b.pdf"))
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3d) rexp(100,1)"
[1] 0.324
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3d) rexp(10,1)"
[1] 0.232
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3a)"
[1] 0.946
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3b) rnorm(10, 1, 1)"
[1] 0.953
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3b) rexp(10, 1, 1)"
[1] 0.777
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3b) rexp(100, 1, 1)"
[1] 0.688
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "1e)"
[1] 0.783
```

```
[1] 2
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "1d)"
[1] 1.75
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "1c)"
[1] 2 2 1 4
> 
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3c)"
[1] 0.05
```

```
> source("/Users/victorpekkari/Desktop/comp_stats/hw1/hw1.r", encoding = "UTF-8")
[1] "3d) rnorm(10,1,1)"
[1] 0.042
```