SEC-1-SA1-GROUP-1-SIGUE,-JP, QUIJANO, JP-SA1

Sigue, John Patrick A., Quijano, Julian Philip S.

2025-03-06

A company has three factories producing a product. Factory 1 produces x_1 of the product, factory 2 produces x_2, and factory 3 produces x_3, where $\sum_{i=1}^{3} x_i = 1$. The defective rates of the products are y_1, y_2, and y_3 respectively, where $\sum_{i=1}^{3} y_i = 0.12$ Write a program (user input for x_i and y_i) to calculate the probability that a randomly selected product is defective.

```
library(shiny)
```

```
## Warning: package ’shiny’ was built under R version 4.4.3
```

```
ui <- fluidPage(
  titlePanel("Defective Product Probability Calculator"),

  sidebarLayout(
    sidebarPanel(
      h3("Production Proportions"),
      numericInput("f1", "Factory 1 Production (%):", value = 33, min = 0, max = 100),
      numericInput("f2", "Factory 2 Production (%):", value = 33, min = 0, max = 100),
      numericInput("f3", "Factory 3 Production (%):", value = 34, min = 0, max = 100),

      h3("Defect Rates"),
      numericInput("d1", "Defect Rate for Factory 1 (%):", value = 4, min = 0, max = 100),
      numericInput("d2", "Defect Rate for Factory 2 (%):", value = 4, min = 0, max = 100),
      numericInput("d3", "Defect Rate for Factory 3 (%):", value = 4, min = 0, max = 100),

      actionButton("calculate", "Calculate Probability")
    ),

    mainPanel(
      h3("Results"),
      verbatimTextOutput("result")
    )
  )
)

server <- function(input, output) {
  observeEvent(input$calculate, {
    f1 <- input$f1 / 100
    f2 <- input$f2 / 100
    f3 <- input$f3 / 100

    d1 <- input$d1 / 100
    d2 <- input$d2 / 100
    d3 <- input$d3 / 100

    if (f1 + f2 + f3 == 1 && d1 + d2 + d3 == 0.12) {
      prob_def <- round(sum(c(f1, f2, f3) * c(d1, d2, d3)), 4)
      output$result <- renderText({
        paste("The probability of selecting a defective product is", prob_def * 100, "%")
      })
    } else {
      output$result <- renderText({
        "Error: Production proportions must sum to 100% and defect rates must sum to 12%."
      })
    }
  })
}

shinyApp(ui = ui, server = server)
```

With your own computing experience, develop a front end to R that allows the user to:

Input the values of a univariate discrete random variable and the associated probabilities and to obtain the mean and variance

Input the values of a bivariate discrete random variable and the associated probabilities and to obtain the marginal and conditional distributions

```
library(shiny)

ui <- fluidPage(
  titlePanel("Discrete Random Variable Calculator"),

  sidebarLayout(
    sidebarPanel(
      tabsetPanel(
        tabPanel("Univariate",
                 numericInput("num_out_uni", "Enter number of outcomes:", value = 1, min = 1),
                 actionButton("submit_uni", "Submit"),
                 uiOutput("uni_outcomes"),
                 uiOutput("uni_probabilities"),
                 verbatimTextOutput("uni_results")
        ),
        tabPanel("Bivariate",
                 numericInput("num_out1_bi", "Enter number of outcomes for variable 1:", value = 1, min
                 numericInput("num_out2_bi", "Enter number of outcomes for variable 2:", value = 1, min
                 actionButton("submit_bi", "Submit"),
                 uiOutput("bi_outcomes"),
                 uiOutput("bi_probabilities"),
                 verbatimTextOutput("bi_results")
        )
      )
    ),

    mainPanel(
      h3("Results"),
      verbatimTextOutput("results")
    )
  )
)

server <- function(input, output, session) {

  # Univariate
  output$uni_outcomes <- renderUI({
    req(input$num_out_uni)
    lapply(1:input$num_out_uni, function(i) {
      numericInput(paste0("outcome_uni_", i), paste("Enter value for outcome", i), value = 0)
    })
  })

  output$uni_probabilities <- renderUI({
    req(input$num_out_uni)
    lapply(1:input$num_out_uni, function(i) {
      numericInput(paste0("prob_uni_", i), paste("Enter probability for outcome", i, "(%)"), value = 0)
    })
  })

  observeEvent(input$submit_uni, {
    outcomes <- sapply(1:input$num_out_uni, function(i) input[[paste0("outcome_uni_", i)]])
    probabilities <- sapply(1:input$num_out_uni, function(i) input[[paste0("prob_uni_", i)]]) / 100

    mean_out <- sum(outcomes * probabilities)
    var_out <- sum((outcomes - mean_out)^2 * probabilities)

    output$uni_results <- renderPrint({
      cat("Mean:", mean_out, "\nVariance:", var_out)
    })
  })

  # Bivariate
  output$bi_outcomes <- renderUI({
    req(input$num_out1_bi, input$num_out2_bi)
    lapply(1:input$num_out1_bi, function(i) {
      lapply(1:input$num_out2_bi, function(j) {
        numericInput(paste0("outcome_bi_", i, "_", j), paste("Enter value for outcome", i, "and variabl
      })
    })
  })

  output$bi_probabilities <- renderUI({
    req(input$num_out1_bi, input$num_out2_bi)
    lapply(1:input$num_out1_bi, function(i) {
      lapply(1:input$num_out2_bi, function(j) {
        numericInput(paste0("prob_bi_", i, "_", j), paste("Enter probability for outcome", i, "and varia
      })
    })
  })

  observeEvent(input$submit_bi, {
    outcomes <- sapply(1:input$num_out1_bi, function(i) {
      sapply(1:input$num_out2_bi, function(j) input[[paste0("outcome_bi_", i, "_", j)]])
    })

    probabilities <- sapply(1:input$num_out1_bi, function(i) {
      sapply(1:input$num_out2_bi, function(j) input[[paste0("prob_bi_", i, "_", j)]]) / 100
    })

    expected_value <- sum(outcomes * probabilities)

    marginal_var1 <- rowSums(probabilities)
    marginal_var2 <- colSums(probabilities)

    conditional_var1_given_var2 <- probabilities / marginal_var2
    conditional_var2_given_var1 <- t(probabilities) / marginal_var1

    output$bi_results <- renderPrint({
      cat("Expected Value:", expected_value, "\n")
      cat("Marginal Distribution for Variable 1:\n")
      print(marginal_var1)
      cat("Marginal Distribution for Variable 2:\n")
      print(marginal_var2)
      cat("Conditional Distribution of Variable 1 given Variable 2:\n")
      print(conditional_var1_given_var2)
      cat("Conditional Distribution of Variable 2 given Variable 1:\n")
      print(conditional_var2_given_var1)
    })
  })
}

shinyApp(ui = ui, server = server)
```

By generating 10,000 searches in R, carry out a simulation experiment for a search engine going through a list of sites for a given key phrase, until the key phrase is found. You should allow your program to input the probability p that any site will contain the key phrase.

Plot the simulated pdf and calculate its mean and variance, and

Obtain the simulated conditional distribution of searches when three searches have been carried out without success. Calculate its mean and variance, and satisfy yourself that they are equivalent to the simulated distribution of the complete set.

As test data assume each site has a 60% chance of containing the key phrase. To satisfy yourself that the Markov memoryless property holds, obtain estimates of

(a) $P(X = 4|X > 3)$ and $P(X = 1)$

(b) $P(X = 5|X > 3)$ and $P(X = 2)$

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
set.seed(123)
n_simulations <- 10000
p <- 0.6

searches <- numeric(n_simulations)
for (i in 1:n_simulations) {
    searches[i] <- rgeom(1, p) + 1
}

searches_df <- data.frame(searches = searches)
ggplot(searches_df, aes(x = searches)) + geom_bar(aes(y = after_stat(count)/sum(after_stat(count))),
    fill = "blue", alpha = 0.7) + labs(title = "Simulated PDF of Searches Until Key Phrase Found",
    x = "Number of Searches", y = "Probability") + theme_minimal()

mean_searches <- mean(searches)
var_searches <- var(searches)

cat("Mean of searches:", mean_searches, "\n")
```

```
## Mean of searches: 1.6612
```

```r
cat("Variance of searches:", var_searches, "\n")
```

```
## Variance of searches: 1.104725
```

```r
conditional_searches <- searches[searches > 3] - 3

mean_conditional <- mean(conditional_searches)
var_conditional <- var(conditional_searches)

cat("Mean of conditional searches:", mean_conditional, "\n")
```

```
## Mean of conditional searches: 1.643533
```

```r
cat("Variance of conditional searches:", var_conditional, "\n")
```

```
## Variance of conditional searches: 1.16499
```

```r
cat("Are the means equivalent?", all.equal(mean_searches, mean_conditional +
    3), "\n")
```

```
## Are the means equivalent? Mean relative difference: 1.795288
```

```r
cat("Are the variances equivalent?", all.equal(var_searches, var_conditional),
    "\n")
```

```
## Are the variances equivalent? Mean relative difference: 0.05455174
```

```r
# P(X = 4 | X > 3) and P(X = 1)
p_x_4_given_x_gt_3 <- mean(searches == 4 & searches > 3)/mean(searches > 3)
p_x_1 <- mean(searches == 1)

cat("P(X = 4 | X > 3):", p_x_4_given_x_gt_3, "\n")
```

```
## P(X = 4 | X > 3): 0.6182965
```

```r
cat("P(X = 1):", p_x_1, "\n")
```

```
## P(X = 1): 0.6045
```

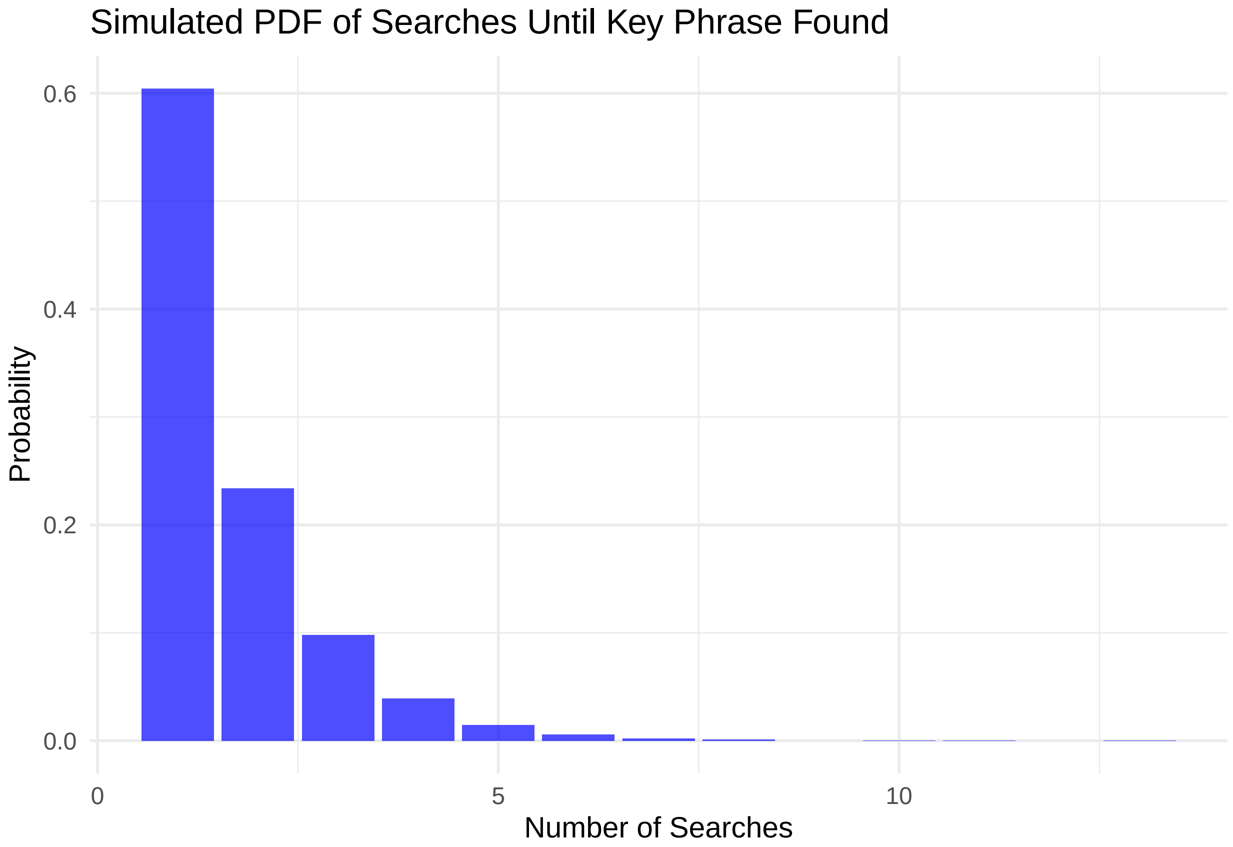
```r
# P(X = 5 | X > 3) and P(X = 2)
p_x_5_given_x_gt_3 <- mean(searches == 5 & searches > 3)/mean(searches > 3)
p_x_2 <- mean(searches == 2)

cat("P(X = 5 | X > 3):", p_x_5_given_x_gt_3, "\n")
```

```
## P(X = 5 | X > 3): 0.2287066
```

```r
cat("P(X = 2):", p_x_2, "\n")
```

```
## P(X = 2): 0.234
```



Github Link: https://github.com/SylTana/APM1110-QUIJANO-JULIAN_PHILIP/tree/main/SA1