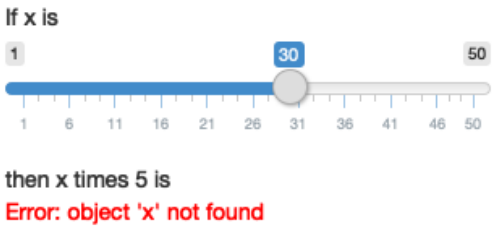
Ćwiczenia 12 (Wizualizacja i przetwarzanie danych)

1. Zaprojektowano aplikację, która pozwala użytkownikowi wybrać liczbę pomiędzy 1 a 50 i wyświetlić wynik iloczynu tej liczby i 5.

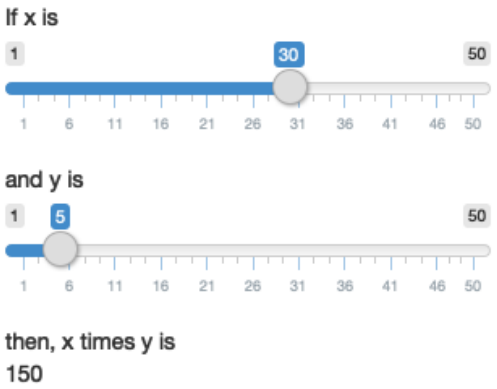
```
ui <- fluidPage(
  sliderInput("x", label = "If x is", min = 1, max = 50, value = 30),
  "then x times 5 is",
  textOutput("product")
)

server <- function(input, output, session) {
  output$product <- renderText({
    x * 5
  })
}

shinyApp(ui, server)
```

Niestety nie działa.



Popraw aplikację tak aby działała.

2. Rozbuduj aplikację z poprzedniego ćwiczenia tak, aby pozwala użytkownikowi ustawić wartość mnożnika.



3. Kolejna aplikacja dodaje kolejne funkcjonalności. Popraw ją tak (zredukuj ilość zduplikowanego kodu), aby wykorzystać funkcje reaktywne.

```
ui <- fluidPage(
  sliderInput("x", "If x is", min = 1, max = 50, value = 30),
  sliderInput("y", "and y is", min = 1, max = 50, value = 5),
  "then, (x * y) is", textOutput("product"),
  "and, (x * y) + 5 is", textOutput("product_plus5"),
  "and (x * y) + 10 is", textOutput("product_plus10")
)

server <- function(input, output, session) {
  output$product <- renderText({
    product <- input$x * input$y
    product
  })
  output$product_plus5 <- renderText({
    product <- input$x * input$y
    product + 5
  })
  output$product_plus10 <- renderText({
    product <- input$x * input$y
    product + 10
  })
}

shinyApp(ui, server)
```

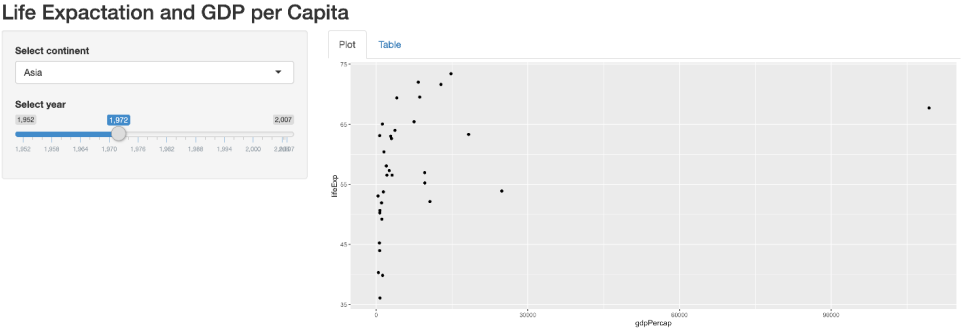
4. Poniższa aplikacja pozwala wybrać zbiór danych z pakietu `ggplot2`, a następnie wyświetla podsumowanie danych i rysuje wykresy. Są w niej jednak trzy błędy. Wskaż je i popraw.

```
datasets <- c("economics", "faithfuld", "seals")

ui <- fluidPage(
  selectInput("dataset", "Dataset", choices = datasets),
  verbatimTextOutput("summary"),
  tableOutput("plot")
)

server <- function(input, output, session) {
  dataset <- reactive({
    get(input$dataset, "package:ggplot2")
```

```
  })
  output$summmry <- renderPrint({
    summary(dataset())
  })
  output$plot <- renderPlot({
    plot(dataset)
  }, res = 96)
}

shinyApp(ui, server)
```

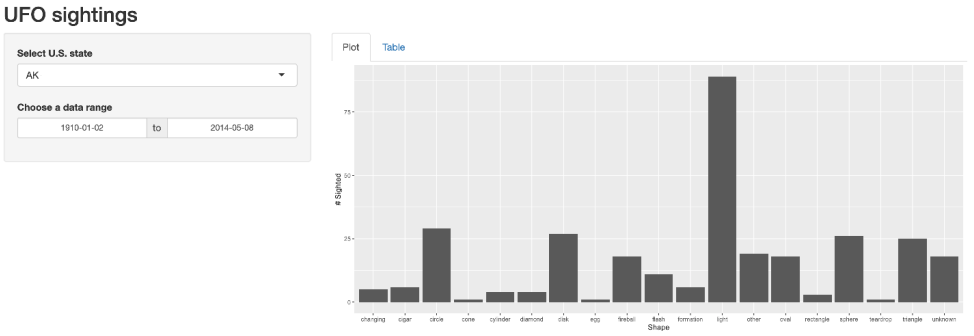
5. (S) Zbuduj aplikację shiny, która pozwala wpisać swoje imię i nazwisko oraz wybrać powitanie (Hello/Bonjour) i zwraca „Hello, Tomasz", gdy użytkownikiem jest Tomasz. Twoja ostateczna aplikacja powinna wizualnie przypominać zrzut ekranu poniżej.

6. (S) Dla zbioru danych *gapminder* z biblioteki `gapminder` zbuduj aplikację shiny, która wizualnie powinna przypominać zrzut ekranu poniżej. Powinna istnieć możliwość wyboru kontynentu i roku. Dla wybranych wartości należy przygotować wykres i tabelę na osobnych zakładkach.



7. Narodowe Centrum Raportowania UFO (NUFORC) zbierało dane o zdarzeniach UFO w ciągu ostatniego stulecia (zbiór danych: *usaUFOsightings.csv*).

Zbuduj aplikację, która pozwoli użytkownikom wybrać stan USA i okres, w którym miały miejsce zdarzenia. Wykres powinien pokazywać liczbę zdarzeń dla wybranego stanu i okresu. Tabela powinna pokazywać, dla wybranego stanu i okresu czasu, liczbę zaobserwowanych obserwacji oraz średnią, medianę, minimalny i maksymalny czas trwania (w sekundach) wydarzenia. Twoja ostateczna aplikacja powinna wizualnie przypominać zrzutu ekranu poniżej.



8. Zbuduj aplikację do wizualizacji Centralnego Twierdzenia Granicznego. Próbki powinny być losowane z rozkładu jednostajnego na odcinku [0,1]. Użytkownik powinien móc zmienić liczbę próbek (od 1 do 100, domyślnie 2) oraz liczbę słupków w histogramie (od 5 do 50, domyślnie 20). Liczba doświadczeń (liczenia średnich) powinna być ustalona na stałe (10 000). Ustaw szablon na *darkly*. Zabezpiecz aplikację przed wpisaniem ujemnych wartości dla liczby próbek i liczby słupków (biblioteka `shinyFeedback`). Dopasuj wygląd aplikacji do zrzutu ekranu poniżej.