Webowy interfejs - Shiny

Przygotowanie danych

1. Uruchom RStudio.
2. Ustaw swój Working Directory używając polecenia setwd().

```
setwd("F:/inazwisko")
```

3. Wykorzystaj dane, które zostały przygotowane w podobny sposób jak na poprzednim laboratorium. Dane pochodzą z raportów miesięcznych z 2015 roku ze stacji Kraków-Kurdwanów (źródło: http://monitoring.krakow.pios.gov.pl/). Pobierz, rozpakuj, a następnie wczytaj dane z pliku.

```
download.file("http://home.agh.edu.pl/~mmd/_media/dydaktyka/adp/dane-pomiarowe-dla-stacji-krakow-kurdwanow.zip", "dane-pomiarowe-dla-stacji-krakow-kurdwanow.zip")
```

```
unzip("dane-pomiarowe-dla-stacji-krakow-kurdwanow.zip")
```

```
data <- dget("./dane-pomiarowe-dla-stacji-krakow-kurdwanow")
```

Webowy interfejs

1. Wczytaj pakiet shiny. Jeśli pakiet nie może zostać załadowany przed wczytaniem użyj polecenia: install.packages("shiny")

```
library(shiny)
```

2. Sprawdź jaka wartość zostanie zapisana jeśli zostanie wywołane poniższe polecenie. Zwróć uwagę, że funkcja generuje kod HTML.

```
ui <- fluidPage("test")
```

```
ui
```

3. Wyświetl wszystkie możliwe tagi html.

```
names(tags)
```

4. Stwórz szablon pliku w Working Directory: app.R, który będzie uruchamiany jako docelowa aplikacja. Jako argument funkcji wprowadź tytuł h1 "test"

```
library(shiny)


ui <- fluidPage(
      tags$h1("test")
)
server <- function(input, output) {
}
shinyApp(ui=ui, server=server)
```

5. Uruchom stworzony program za pomocą poniższego polecenia. Aplikacja webowa zostanie automatycznie wyświetlana. Wyświetlenie aplikacji będzie również możliwe poprzez wpisanie w przeglądarce adresu: http://127.0.0.1:6502/

```
shiny::runApp()
```

6. Zamknij aplikacje klikając w konsoli ESC.
7. Stwórz aplikacje, która umożliwi wyświetlenie wybranego histogramu ("SO2" "NO2" "NOx" "NO" "O3" "O38h" "PM10" "PM25"). Wybór powinien być możliwi w formie select'a. Wprowadź w funkcji fluidPage() poniższe argumenty. Rozdziel argumenty przecinkami.
    a. sliderInput (przykładowy element *input)
    b. plotOutput (przykładowy element *output)
    c. renderPlot (przykładowy element *render)

```
library(shiny)


ui <- fluidPage(

      selectInput(inputId = 'mySelectInput',
                  label='Narysuj histogram dla danych',
                  choices = names(data)[1:8], selected = "PM25"),

      plotOutput(outputId = 'myPlotOutput')

)

server <- function(input, output) {

      output$myPlotOutput <- renderPlot({
            hist(data[[input$mySelectInput]])
      })

}

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

8. Ponownie uruchom aplikacje i przeanalizuj uzyskany efekt. Zwróć uwagę, że zmienna input$mySelectInput jest reaktywna, tzn. jakakolwiek jej zmiana powoduje ponowne uruchomienie funkcji renderPlot, która modyfikuje wartość output$myPlotOutput. Zwróć uwagę na to, że zmienne rekatywne mogą być używane wyłącznie w reaktywnych funkcjach (np. *Render). Po przeanalizowaniu aplikacji zamknij aplikacje klikając w konsoli ESC.
9. Dodaj pole tekstowe, w którym będzie można zdefiniować tytuł. Użyj funkcji isolate(), aby tytuł nie był zmieniany automatycznie. Zwróć uwagę, że tytuł zostanie zmieniony dopiero po zmienieniu danych, które mają zostać przedstawione na histogramie.

```
library(shiny)


ui <- fluidPage(

      textInput(inputId = 'myTitle', label = 'Tytuł wykresu',
                value = 'Histogram'),
```

```
      selectInput(inputId = 'mySelectInput',
                  label='Narysuj histogram dla danych',
                  choices = names(data)[1:8], selected = "PM25"),

      plotOutput(outputId = 'myPlotOutput')

)

server <- function(input, output) {

      output$myPlotOutput <- renderPlot({

            hist(data[[input$mySelectInput]],
                 main=isolate(input$myTitle))

      })

}

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

10. Dodaj przycisk ‘Zmień tytuł’. Zmodyfikuj kod tak, aby dopiero po jego naciśnięciu tytuł wykresu zmienił się na tekst wprowadzony w polu tekstowym.

```
library(shiny)


ui <- fluidPage(

      textInput(inputId = 'myTitle', label = 'Tytuł wykresu',
                value = 'Histogram'),

      actionButton(inputId = 'change', label = 'Zmień tytuł'),

      selectInput(inputId = 'mySelectInput',
                  label='Narysuj histogram dla danych',
                  choices = names(data)[1:8], selected = "PM25"),

      plotOutput(outputId = 'myPlotOutput')

)

server <- function(input, output) {

      title <- eventReactive(input$change, {input$myTitle},
                             ignoreNULL = FALSE)

      output$myPlotOutput <- renderPlot({

            hist(data[[input$mySelectInput]], main=title())

      })

}

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

11. Dodaj przycisk umożliwiający zapis wykresu.

```
library(shiny)


ui <- fluidPage(

      textInput(inputId = 'myTitle', label = 'Tytuł wykresu',
                value = 'Histogram'),

      selectInput(inputId = 'mySelectInput',
                  label='Narysuj histogram dla danych',
                  choices = names(data)[1:8], selected = "PM25"),

      plotOutput(outputId = 'myPlotOutput'),

      actionButton(inputId = 'save', label="Zapisz")

)

server <- function(input, output) {

      output$myPlotOutput <- renderPlot({

            hist(data[[input$mySelectInput]], main=input$myTitle)

      })

      observeEvent(input$save, {

            hist(data[[input$mySelectInput]],
                  main=isolate(input$myTitle))

            dev.copy(pdf, "shiny.pdf")

            dev.off()

      })

}

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

12. Stwórz aplikacje, która umożliwi wyznaczenie linii regresji dla wybranej pary danych spośród: "SO2" "NO2" "NOx" "NO" "O3" "O38h" "PM10" "PM25". Do obliczenia modelu wykorzystaj zmienną reaktywną. Wprowadź w aplikacji następujące elementy:
    a. sliderInput – wybór parametru wyświetlanego na osi x
    b. sliderInput – wybór parametru wyświetlanego na osi y
    c. textOutput – wyświetlenie współczynników a oraz b
    d. plotOutput – wykres z linią regresji

```
library(shiny)


ui <- fluidPage(

      selectInput(inputId = 'mySelectInput1', label='Oś x',
                  choices = names(data)[1:8],
                  selected = names(data)[1]),
```

```
      selectInput(inputId = 'mySelectInput2', label='Oś y',
                  choices = names(data)[1:8],
                  selected = names(data)[2]),

      textOutput(outputId = 'model'),

      plotOutput(outputId = 'myPlotOutput')

)

server <- function(input, output) {

      model <- reactive({

            lm(data[[input$mySelectInput2]] ~
               data[[input$mySelectInput1]])

      })

      output$model <- renderText({

            paste("y = ", round(model()$coefficients[2],2),
                  " * x + ", round(model()$coefficients[1],2))

      })

      output$myPlotOutput <- renderPlot({

            plot(data[[input$mySelectInput2]] ~
                 data[[input$mySelectInput1]])

            abline(model(), col="red", lwd=5)

      })

}

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

Lista funkcji, która może zostać użyta do stworzenia aplikacji:

```
sliderInput(), actionButton(), submitButton(), checkboxInput(),
checkboxGroupInput(), dateInput(), dateRangeInput(), fileInput(),
numericInput(), passwordInput(), radioButtons(), selectInput(),
sliderInput(), textInput(), dataTableOutput(), htmlOutput(),
imageOutput(), plotOutput(), tableOutput(), textOutput(),
uiOutput(), verbatimTextOutput(),renderDataTable(), renderImage(),
renderPlot(), renderPrint(), renderTable(), renderText(), renderUI()
```