Nauka dla społeczeństwa

Katedra i Zakład Informatyki i Statystyki

dr Justyna Marcinkowska

data przygotowania: 2024-05-05

Zalecenia dotyczące warsztatów cz. 4. – kody RShiny c.d.:

*Praktyka dla statystyka*

*BADANIA Z OBSZARU ZDROWIA*

*RShiny*

[R Shiny](https://raw.githubusercontent.com/rstudio/cheatsheets/main/shiny.pdf) to pakiet języka R do tworzenia interaktywnych aplikacji webowych. Aplikacje Shiny mogą być wykorzystywane do tego, by własne procedury statystyczne udostępniać  w przyjaznej formie badaczom nie posiadającym umiejętności programistycznych w języku R.

**Podczas warsztatów uczestnicy nauczą się:**

1. Tworzyć proste aplikacje Shiny
2. Dodawać elementy interaktywne do aplikacji
3. Wizualizować dane w aplikacji

# zad.6

#"Warsztaty z RShiny PRAKTYKA DLA STATYSTYKA"
#cz.II
#zadanie 6:
# tabelka i wykres w tabsetPanel:
# interaktywny opis w tabelce + check + zmienna grupująca

# tu library

 sodium <- read\_excel("../sodium.xlsx", sheet = "data")
 sub\_sodium<- sodium %>%
 filter(hospitalization==1)

 sub\_sodium$medical\_transport <- factor(sub\_sodium$medical\_transport,
 levels = c(0, 1), labels = c("nie", "tak"))

 pal <- pal\_jco() # przypisanie palety do zmiennej

 ui <- fluidPage(

 # Application title
 titlePanel("Pierwszy wykres interaktywny"),
 # Sidebar with a slider input for number of bins
 sidebarLayout(
 sidebarPanel(
 wellPanel(style = "background: darkgray",
 selectInput("zmienna", "Wybierz zmienną:",
 choices = c("Wiek" = "age",
 "Stężenie sodu" = "sodium\_concentration",
 "Godziny pobytu na SOR" = "Stay\_in\_the\_Emergency\_Department"),
 selected = "age"),
 selectInput("zmienna\_gr", "Wybierz grupującą:",
 choices = c("Płeć" = "sex",
 "Oddział szpitalny" = "department",
 "Czy dowieziony transportem medycznym?" = "medical\_transport"),
 selected = "sex")
 )
 ),
 mainPanel(
 tabsetPanel(id="mojTabset",
 tabPanel("Wykresy",
 # tu będzie definicja plotOutput:
 wellPanel(style = "background: slateblue",plotOutput("wykres"))

 ),
 tabPanel("Obliczenia",
 # tu będzie definicja tableOutput:
 wellPanel(style = sprintf("color: white; background-color: %s;", pal(10)[8]),
 tableOutput('tab1')
 )
 )
 )

 )))

 server <- function(input, output) {

 output$tab1 <- renderTable({

 sub\_sodium %>%
 group\_by(!!sym(input$zmienna\_gr)) %>%
 summarise(N=n(),
 "średnia" = mean(!!sym(input$zmienna)),
 "sd" = sd(!!sym(input$zmienna)),
 "mediana" = median(!!sym(input$zmienna)),
 'kwartyl dolny'= quantile(!!sym(input$zmienna),0.25),
 'kwartyl górny'= quantile(!!sym(input$zmienna),0.75)
 )

 })

 output$wykres<-renderPlot({
 sub\_sodium %>%
 ggplot( aes(x=!!sym(input$zmienna\_gr),y=!!sym(input$zmienna))) +
 geom\_violin(aes(fill = !!sym(input$zmienna\_gr)))+
 scale\_fill\_jco()+
 theme\_classic()
 })
 }
 #Run the application
 shinyApp(ui = ui, server = server)

# zad.7 - projekt

#"Warsztaty z RShiny PRAKTYKA DLA STATYSTYKA"

#cz.II
#zadanie 7 - projekt aplikacji interaktywnej - rozbudowujemy kod

# sprawdź aplikację RShiny pod adresem:
# https://kpkuwz-justyna-marcinkowska.shinyapps.io/warsztaty\_NDS/
# i zainspiruj się lub zrealizuj własne pomysły :)
#Propozycje:
# dodanie nowych grupujących: grupa wieku, grupa stężenia sodu,
# przypisanie dla wykresu jitter size,fill i color

#poziomy sodu: normy:
#breaks = c(0, 129, 135, 144,150,170),
#labels = c("very low values",
 # "below standard",
 # "standard",
 # "above normal",
 # "very high values")

library(shiny)
library(readxl)
library(dplyr)
library(ggplot2)
library(ggsci)
library(plotly)

 sodium <- read\_excel("../sodium.xlsx", sheet = "data")
 sodium$medical\_transport <- factor(sodium$medical\_transport,
 levels = c(0, 1), labels = c("nie", "tak"))

 sub\_sodium<- sodium %>%
 filter(hospitalization==1)
 pal <- pal\_jco() # przypisanie palety do zmiennej

 ui <- fluidPage(

 div(style = "display: block; width: 100%;",
 div(style = "display: flex; align-items: center;",
 titlePanel(HTML("Warsztaty z RShiny <br> PRAKTYKA DLA STATYSTYKA")),
 img(src = "NDS.png", height = "100px", style = "margin-left: 20px;")
 )
 ),

 sidebarLayout(

 sidebarPanel(
 fluidPage(
 style = sprintf("background-color: %s;", pal(10)[3]),
 checkboxInput('czy\_hosp', 'Pacjenci hospitalizowani', value = TRUE),
 varSelectInput("variable", "Wybierz zmienną:",
 sub\_sodium[, c("age", "sodium\_concentration", "Stay\_in\_the\_Emergency\_Department")],
 selected = "age"),
 varSelectInput("gr\_variable", "Wybierz grupującą:",
 sub\_sodium[, c("department", "sex")],
 selected = "department"),
 downloadButton("downloadPlot", "Pobierz wykres"),p(""),p("")
 )
 ),
 mainPanel(
 tabsetPanel(id="mojTabset",
 tabPanel("Wykresy",
 wellPanel(style = sprintf("background-color: %s;", pal(10)[5]),
 radioButtons("czy\_box", "Typ wykresu: ",
 choices = c("box plot" = "b", "violin" = "v", "jitter" = "j"),
 inline = TRUE),
 plotlyOutput("data\_plotly") )

 ),
 tabPanel("Obliczenia",
 wellPanel(style = sprintf("color: white; background-color: %s;", pal(10)[8]),
 tableOutput('tab1'),
 plotOutput("data\_plot")
 )
 )
 )
 )
 )
 )

 server <- function(input, output) {
 # ----
 hospitalizacja=T
 p<-ggplot()
 pl<-ggplot()

 output$tab1 <- renderTable({
 hospitalizacja=input$czy\_hosp
 df\_sodium <- if(hospitalizacja) sub\_sodium else sodium

 df\_sodium %>% group\_by(!!input$gr\_variable) %>%
 summarise(N=n(),
 średnia = mean(!!input$variable),
 sd = sd(!!input$variable),
 mediana = median(!!input$variable),
 'kwartyl dolny'= quantile(!!input$variable,0.25),
 'kwartyl górny'= quantile(!!input$variable,0.75)
 )

 })

 p <- reactive({
 hospitalizacja=input$czy\_hosp
 df\_sodium <- if(hospitalizacja) sub\_sodium else sodium

 # Obliczanie optymalnej szerokości koszyka zgodnie z regułą Freedmana-Diaconisa
 IQR <- IQR(df\_sodium[[input$variable]]) # Międzykwartylowy rozstęp danych
 n <- length(df\_sodium[[input$variable]]) # Liczba punktów danych
 binwidth\_fd <- 2 \* IQR / (n^(1/3))
 # Obliczenie liczby przedziałów
 bins <- ceiling(sqrt(n))

 df\_sodium %>%
 ggplot( aes(x=!!input$variable))+
 geom\_histogram(bins=bins, binwidth = binwidth\_fd, aes(y=after\_stat(density)),alpha=0.8,color="black")+
 geom\_density(alpha=0.5,aes(fill=!!input$gr\_variable))+
 labs(y="gęstość")+
 scale\_fill\_jco()+
 scale\_color\_jco() + # Dla color w geom\_density
 theme\_classic()

 })

 pl <- reactive({
 hospitalizacja=input$czy\_hosp
 df\_sodium <- if(hospitalizacja) sub\_sodium else sodium

 if(input$czy\_box=='b'){
 df\_sodium %>%
 ggplot( aes(x=!!input$gr\_variable,y=!!input$variable)) +
 geom\_boxplot(aes(fill = !!input$gr\_variable))+
 scale\_fill\_jco()+
 theme\_classic()
 }else if(input$czy\_box=="j"){
 df\_sodium %>%
 ggplot( aes(x=!!input$gr\_variable,y=!!input$variable)) +
 geom\_jitter(aes(fill = !!input$gr\_variable))+
 scale\_fill\_jco()+
 theme\_classic()
 }else{
 df\_sodium %>%
 ggplot( aes(x=!!input$gr\_variable,y=!!input$variable)) +
 geom\_violin(aes(fill = !!input$gr\_variable))+
 scale\_fill\_jco()+
 theme\_classic()
 }

 })

 output$data\_plotly <- renderPlotly({ ggplotly(pl())})
 output$data\_plot <- renderPlot({ p()})

 # Obsługa pobierania wykresu
 output$downloadPlot <- downloadHandler(
 filename = function() {

 rodzaj <- if (input$mojTabset == "Obliczenia") "histogram"
 else if(input$czy\_box == "b") "box plot" else if(input$czy\_box == "v") "violin" else "jitter"
 nazwa\_png <- paste0(input$variable, ' w grupach ', input$gr\_variable, " wykres ", rodzaj)

 paste(nazwa\_png, Sys.Date(), ".png", sep="")
 },
 content = function(file) {
 p\_wykres <- if(input$mojTabset =="Obliczenia") p() else pl()
 # Zapis wykresu do pliku
 ggsave(file, plot = p\_wykres, width = 10, height = 6, dpi = 300)
 }
 )
 }
 #Run the application
 shinyApp(ui = ui, server = server)