Bitcoin

Magdalena Mazur
Agata Rogowska
Zuzanna Różak
Aleksandra Siepiela

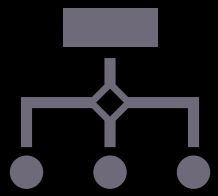
- Analiza wartości Bitcoina na przestrzeni lat
- Badanie danych giełdowych poszczególnych okresach czasu

Przykładowy fragment tabeli danych

	Timestamp	Open	High	Low	Close	Volume_.BTC.	Volume_.Currency.	Weighted_Price
1690567	1427140200	262.86	262.86	262.85	262.85	3.99314076	1049.617049	262.8550
1690568	1427140260	262.65	262.83	262.40	262.40	14.63925000	3845.605167	262.6914
1690569	1427140320	262.68	262.68	262.65	262.65	0.11900000	31.256730	262.6616
1690570	1427140380	262.59	262.59	262.59	262.59	0.58791079	154.379494	262.5900
1690571	1427140440	262.58	262.74	262.24	262.68	4.01710000	1054.796778	262.5767
1690572	1427140500	262.59	262.59	262.54	262.55	9.59620000	2519.514158	262.5533
1690573	1427140560	262.55	262.55	262.18	262.18	16.82765981	4412.186723	262.1985
1690574	1427140620	262.06	262.18	262.00	262.18	41.57399416	10892.634632	262.0060
1690575	1427140680	262.18	262.18	262.18	262.18	0.47870155	125.505972	262.1800
1690576	1427140740	262.18	262.18	262.18	262.18	1.14620015	300.510755	262.1800
1690577	1427140800	262.18	262.47	262.18	262.47	1.75151575	459.240297	262.1959
1690578	1427140860	262.47	262.55	262.47	262.55	9.31200000	2444.637441	262.5255
1690579	1427140920	262.47	262.53	262.47	262.51	3.38670969	889.054973	262.5129

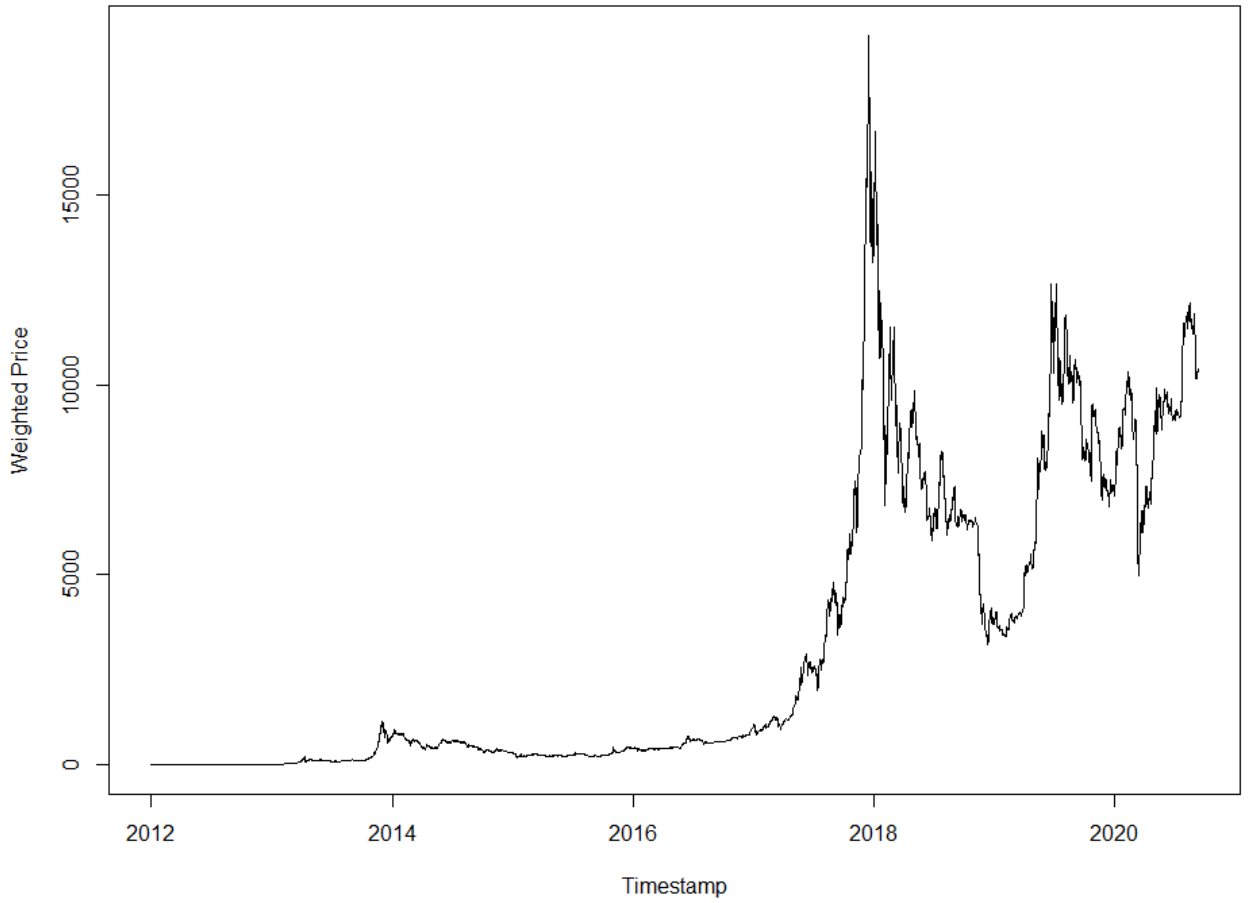
Źródło: yahoo.com

Format: .csv



Przekształcenia: dopasowanie nazw kolumn do funkcjonalności, rozwiązanie problemu brakujących danych

Średnia dzienna cena Bitcoina na przestrzeni lat



Funkcjonalności aplikacji:

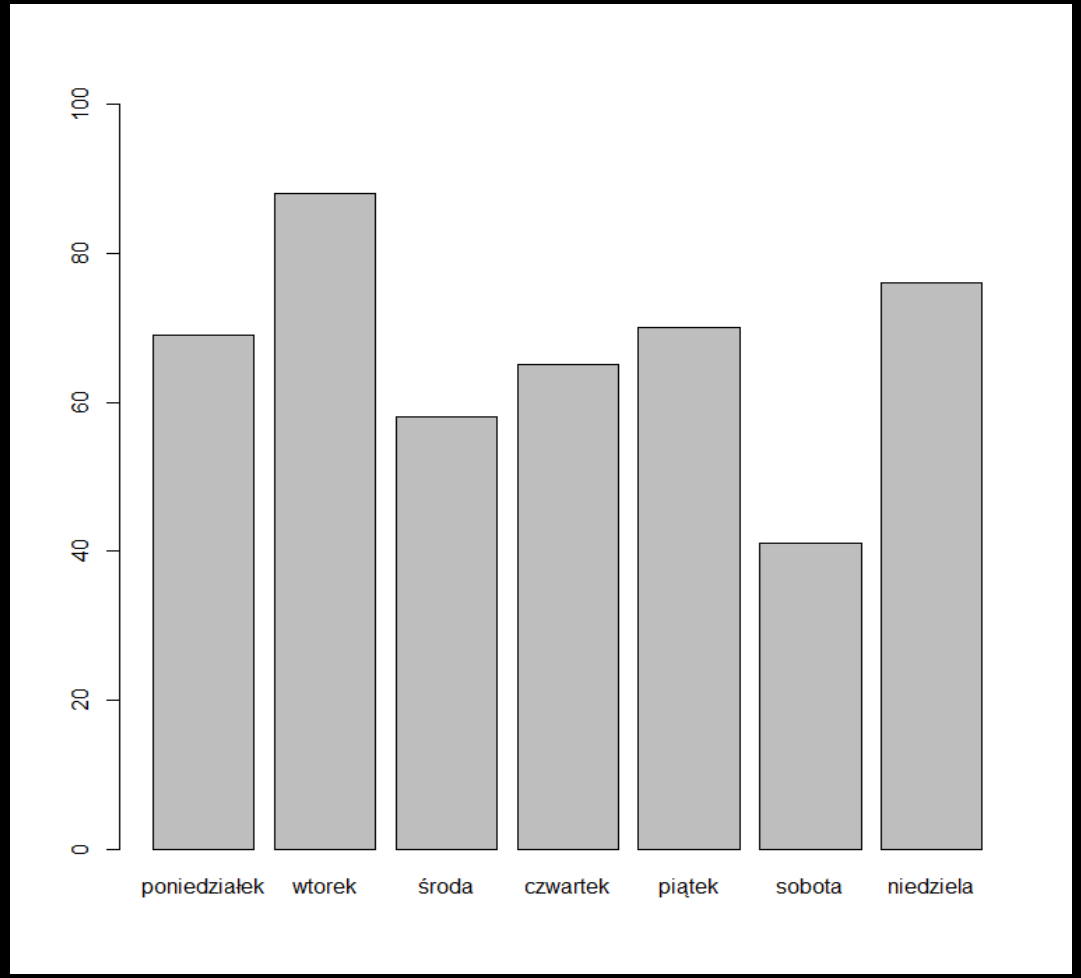
- Analiza ceny otwarcia, zamknięcia, ceny maksymalnej i minimalnej, wolumenu w wybranym zakresie czasowym
- Wizualizacja wskaźników giełdowych na wybranym przedziale czasu
- Prezentacja danych historycznych
- Wizualizacja danych użytkownika na wybranym przedziale czasu

Implementacja funkcjonalności kodu

```
# statystyki opisowe z danego dnia (input - konkretny dzien) ----
input <- "2019-01-01"
d[YMD == as.Date(input), .(Min = min(Weighted_Price, na.rm = TRUE),
                           Q1 = quantile(Weighted_Price, 0.25, na.rm = TRUE),
                           Mean = mean(Weighted_Price, na.rm = TRUE),
                           Median = median(Weighted_Price, na.rm = TRUE),
                           Q3 = quantile(Weighted_Price, 0.75, na.rm = TRUE),
                           Max = max(Weighted_Price, na.rm = TRUE))]
boxplot(d[YMD == as.Date(input), Weighted_Price])
plot(d[YMD == as.Date(input), Volume_BTC])
```

Implementacja funkcjonalności kodu

```
# najlepszy dzień na zakupy ----
dow <- function(x) format(as.Date(x), "%A")
d[, DayOfWeek := dow(d$Date)] # wday
d[, MinWP := min(Weighted_Price), by = .(Week, Year)]
day <- unique(d[MinWP == Weighted_Price, DayOfWeek,
                by= .(Week, Year)])[, DayOfWeek]
day <- factor(day,
              levels = c('poniedziałek', 'wtorek',
                         'środa', 'czwartek', 'piątek',
                         'sobota', 'niedziela'))
barplot(table(day), ylim = c(0, 100))
```



Plan projektu

- Pobieranie danych z bazy danych
- Wstępna eksploracja danych
- Wybór najlepszej strategii dotyczącej braków danych
- Wizualizacja danych
- Stworzenie aplikacji w Shiny

Dziękujemy za uwagę!

PREZENTACJA KOŃCOWA

Magdalena Mazur
Agata Rogowska
Zuzanna Różak
Aleksandra Siepiela

WCZYTANIE DANYCH DOMYŚLNYCH

```
getSymbols("BTC-USD", src = "yahoo")
dates <- index(`BTC-USD`)
d.day <- as.data.frame(`BTC-USD`)
d.day <- as.data.table(cbind(dates, d.day))
names(d.day) <- c('Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adjusted')
d.day <- na.omit(d.day)
```

WCZYTANIE DANYCH UŻYTKOWNIKA

```
UserData <- eventReactive(input$upload,{
    inFile <- input$upload
    ext <- tools::file_ext(input$upload$name)
    switch(ext,
           csv = vroom::vroom(input$upload$datapath, delim = ","),
           validate("Invalid file. Please upload a .csv file.")
    )
    if (is.null(inFile)){
        returnValue()
    }
    else{
        d <- read.csv(inFile$datapath)
        d <- as.data.table(d)
        d[, Date := as.Date(Date)]
        d
    }
})
```

```
changeColor <- function(d, price){
  diffrence <- abs(d[which.max(as.Date(Date)), round(get(price), 2)] - data[which.max(as.Date(Date)) - 1, round(get(price), 2)])
  if(d[which.max(as.Date(Date)), round(get(price), 2)] > d[which.max(as.Date(Date)) - 1, round(get(price), 2)]){
    infoBox(
      title = paste(price, " price"),
      value = tags$p(d[which.max(as.Date(Date)), round(get(price), 2)], "$", style = "font-size: 150%;"),
      icon = icon("arrow-up"),
      subtitle = tags$a(icon("arrow-up"), paste(round((diffrence/d[which.max(as.Date(Date)) - 1, round(get(price), 2)])*100,2), "%"), style = "color: white"),
      color = "green", fill = TRUE
    )
  }
  else{
    infoBox(
      title = paste(price, " price"),
      value = tags$p(d[which.max(as.Date(Date)), round(get(price), 2)], "$", style = "font-size: 150%;"),
      icon = icon("arrow-down"),
      subtitle = tags$a(icon("arrow-down"), paste(round((diffrence/d[which.max(as.Date(Date)) - 1, round(get(price), 2)])*100,2), "%"), style = "color: white"),
      color = "red", fill = TRUE
    )
  }
}
```

RENDEROWANIE INFOBOXÓW

WYZNACZANIE PRZEDZIAŁÓW CZASOWYCH

```
ZoomDataRange <- function(d, input1, input2){
  if (input1 == "3D"){
    c(as.Date(max(d$Date))-2, as.Date(max(d$Date)))
  }
  else if (input1 == "7D"){
    c(as.Date(max(d$Date))-6, as.Date(max(d$Date)))
  }
  else if (input1 == "1M"){
    c(as.Date(max(d$Date)) %m-% months(1), as.Date(max(d$Date)))
  }
  else if (input1 == "3M"){
    c(as.Date(max(d$Date)) %m-% months(3), as.Date(max(d$Date)))
  }
  else if (input1 == "6M"){
    c(as.Date(max(d$Date)) %m-% months(6), as.Date(max(d$Date)))
  }
  else if (input1 == "1Y"){
    c(as.Date(max(d$Date)) %m-% years(1), as.Date(max(d$Date)))
  }
  else if (input1 == "Range"){
    input2
  }
  else if (input1 == "All"){
    c(as.Date(min(d$Date)), as.Date(max(d$Date)))
  }
}
```

PRZYGOTOWANIE DANYCH

```
PrepareZoomDataOHLC <- function(d, data_range){
  d[Date <= data_range[2] & Date >= data_range[1], .(Date, Open, High, Low, Close)]
}
```

RYSOWANIE WYKRESU LINIOWEGO

```
PlotLineChart <- function(d, input1){
  lineChart(d[, .(Date, get(input1))], line.type = 'h', name = "Price", theme = chartTheme("white", up.col = "#f2a900"))
}
```

```r
PlotCandlestickChart <- function(d, avg, periods){
  chartSeries(d, type = "candlesticks", name = "Candlestick chart", theme = chartTheme('white', up.col = "green", dn.col = "red"))
  if (avg == 1){
    plot(addSMA(n = as.numeric(periods), overlay = TRUE, col = "blue"))
  }
  else if (avg == 2){
    plot(addWMA(n = as.numeric(periods), overlay = TRUE, col = "orange"))
  }
  else if (avg == 3){
    plot(addEMA(n = as.numeric(periods),  overlay = TRUE, col = "green"))
  }
}
```

RYSOWANIE WYKRESU ŚWIECOWEGO

```
observe({
    updateSelectInput(session = session,
                      "numOfPeriods",
                      choices = 1:as.numeric(ZoomDataRange(d.day,
                                                           input$radioButtons_Zoom,
                                                           input$dateRangeInput_Date)[2] - ZoomDataRange(d.day,
                                                                                                         input$radioButtons_Zoom,
                                                                                                         input$dateRangeInput_Date)[1]))
})
```

WYZNACZANIE LICZBY PRZEDZIAŁÓW DO WYKRESU ŚWIECOWEGO

WYZNACZANIE I RYSOWANIE STÓP ZWROTU

```
GetRateData <- function(d, type, period, data_range){
  if (period != "all"){
    periodReturn(d[Date <= data_range[2] & Date >= data_range[1], .(Date, Open, High, Low, Close)],
                 type = type,
                 period = period)
  }
  else {
    allReturns(d[Date <= data_range[2] & Date >= data_range[1], .(Date, Open, High, Low, Close)],
               type = type)
  }
}

PlotRate <- function(d){
  lineChart(d, line.type = 'h', name = "Rates of return",
            theme = chartTheme("white", up.col = "#f2a900"))
}
```

GENEROWANIE TABELI

```
GetZoomTableOHLC <- function(d){
  DT::datatable(d[, .(Date, Open, High, Low, Close)][order(Date, decreasing = TRUE)],
                options = list(paging = FALSE, searching = FALSE))
}
```

WYRÓŻNIONE FRAGMENTY KODU

```r
header <- dashboardHeader()
sidebar <- dashboardSidebar()
body <- dashboardBody()

shinyUI(
    dashboardPage(
        header,
        sidebar,
        body
    )
)
```

SHINY DASHBOARD W UŻYCIU

TWORZENIE ANIMOWANEGO LOGO

```
header <- dashboardHeader(
    title = list(span(img(src="btcLOGO.png", width = 100), span(img(src="btcGIF.gif", width = 50))))
)
```

```
output$dates <- renderUI({
    dates <- as.Date(UserData()$Date)
    minval <- min(dates)
    maxval <- max(dates)

    dateRangeInput("dateRangeInput_Date_user_data", label = "Choose date range:",
                   start = minval, end = maxval,
                   min = minval, max = maxval)
})
```

INTERAKTYWNY dateRangeInput

```
# ui ----

actionButton("info", "Info")

# server ----

observeEvent(input$info_line, {
    showModal(modalDialog("..."))
})
```

INFORMACJE O WYKRESACH

```
data_rate <- reactive({
    GetRateData(d.day,
                input$selectInput_rate_type,
                input$selectInput_rate_period,
                dateRange())
})

output$line_rate <- renderPlot({
    PlotRate(data_rate())
})
```

GENEROWANIE WYKRESU W ZALEŻNOŚCI OD WYBORU

KONIEC