Data Vizualization Documentation

Biometric Wearables

This project is a part of the Purdue data science initiative, where students are given an opportunity to work on the data-driven project, guided by an industry mentor. This Biometric Wearables project is a part of an ongoing collaboration between the Purdue and Merck industry. Following is the brief description of the project:

Purpose

The purpose of the project is to create an automated system that will capture the biometric data from wearable fitness technology using the mobile application and communicate its' result to Merck scientists.

Outcome

This automated system will be able to collect accurate and less biased data from the clinical trial patients because the digital process reduces the human error and reduce the pressure on the clinical trial patient with manual processing procedure.

Significance

This project may help to reduce the experiment biased by collecting more extensive data and using the technology. Also, this approach may help to reduce the clinical trial time and cost to ease of data collection.

Data visualization Team

This team is a part of a big team working on the above-described project, part of the Merck-Purdue data science collaboration. This team has been assigned the responsibility to design rich, interactive graphics, data visualization, and plotting method to facilitate the MERCK scientists. In addition, this team is responsible for brainstorming the data, and experiments with different visualization techniques that can help the project.

Previous Work Done on the project

The previous team that worked on the data visualization aspect of this project created a baseline shiny website that had some specific widgets created. There are two pages on the shiny website i.e., activity progress and step goals. The activity progress has a bar chart that shows the progress by comparing two biometric data points. The bar chart has an x-axis which is to show the dates and a y-axis which is to show the steps. One of the widget's features on this page was created to view a specific time period of the data. The second page is to show step goals using a pie chart.

Team Members

Following is the brief introduction of the data visualization team members:

Ahmed Ashraf Butt

He is a doctoral student at the School of Engineering Education, Purdue University. He is currently working as a research assistant on the CourseMIRROR project funded by the Institute of Education Sciences (IES). He is interested in designing educational tools and exploring their impact on enhancing students' learning experiences. Also, he is working on the Merck-Purdue data science collaboration responsible for creating an automated system that will capture the biometric data from the wearable fitness technology using the mobile application and communicate its result to Merck scientists.

Pika Seth

She is a second-year student in the College of Engineering studying Biomedical Engineering at Purdue University. She is currently involved with the Data Mine Corporate Program working on the Biometrics Wearable Project with Merck. She is involved with the data visualization aspect dealing with the programming of R Shiny. She is a part of the Red Cross Club and the Women in Engineering Program. Also, she is a part of the Sigma Delta Tau sorority.

Sharon Patta

She is a first-year student in the College of Science at Purdue University. She is studying data science. Her interests are in the fields of AI and machine learning.

Surya Suresh

He is a first-year student in the College of Science at Purdue University studying Computer Science. He is working with the Merck Data Mine group on the Data Visualization part of the project.

Team formation

The team has worked together to become more familiar with R programming and shiny application. Each member of the group explored different data visualizations and then, worked colloborately to implement a new visualization or improving the previous shiny app. Additionally, the group had several meetings to collectively present findings to one another and discuss errors and troubleshooting.

Term goals

We are going to begin working on adding more features to the old Shiny App as well as implementing our own visualizations that we think would be useful. We also need to figure out where we are getting our data from once the API is running because as of right now we are just taking it off of the CSV file that has Dr. Ward's data.

Visulization 1

In this visualization, we were able to create a visualization that adapts to the given data and creates a Data Table with all of the given data. We used the Data Tables library to accomplish this. The Data Table contains two main tabs. One tab contains all of the mean values of the given data. The other tab allows the user to navigate through all of the data points in a clean and efficient way.Specifically it allows the user to filter the data as well as list the data in any desired order. For example, the data can be listed in order from the days with the most steps to the days with the least. ## Code

I imported the libraries needed for the vizualization

```
library(shiny) #Used for Shiny Dashboard
library(ggplot2) #Needed for Plots (Not specifically this Viz)
library(reshape2) #Needed for Plots (Not specifically this Viz)
library(DT) #Used for the Data Tables used in this Vizualization
```

biometric data 'bioFinal.csv' for biometric data from 2020 Feb. to present? - Merckteam's data 'DrWard-Data.csv' for biometric data from 2016 to 2019 April - Dr. Ward's data

Used to read data into the myDF data frame

```
myDF <- read.csv("liveMerck.csv",TRUE,",")
```

active miles (stacked)

```
miles1 <- subset(myDF, select = c('Date','Lightly.Active.Miles',
                                  'Moderately.Active.Miles',
                                  'Very.Active.Miles'))
miles <- melt(miles1)
miles$Date <- as.Date(miles$Date)
```

active minutes (stacked)

```
minutes1 <- subset(myDF, select = c('Date', 'Sedentary.Minutes',
                                    'Lightly.Active.Minutes',
                                    'Fairly.Active.Minutes',
                                    'Very.Active.Minutes'))
minutes <- melt(minutes1)
minutes$Date <- as.Date(minutes$Date)
```

HR minutes (stacked)

```
hr1 <- subset(myDF, select = c('Date', 'HR.30.100.Minutes',
                               'HR.100.140.Minutes', 'HR.140.170.Minutes',
                               'HR.170.220.Minutes'))
hr <- melt(hr1)
hr$Date <- as.Date(hr$Date)
```

Finding out the collums index with numeric value in it.

```
columsIndexWithNumberValue <- unlist(lapply(myDF, is.numeric))
```

Extracting the columns with number value in it.

```
columWithNumber <- myDF[ , columsIndexWithNumberValue]
```

finding the mean of all collum's mean

```
meansOFColumWithNumber <- lapply(columWithNumber[,], mean,na.rm=TRUE)
```

Converting them back into data frame.

```
meansOFColumWithNumberDataFrame<-data.frame(matrix(unlist(meansOFColumWithNumber), nrow=length(meansOFC
```

Extracting the name of the collums names from columWithNumber.

```
colName<- data.frame(names(columWithNumber))
```

Extracting the name of the collums.

```
finalDataFrame <- data.frame(a=colName, c=meansOFColumWithNumberDataFrame)
```

changing the Collum name of the final dataframe

```
colnames(finalDataFrame)[1]<-"Variables"
colnames(finalDataFrame)[2]<-"Mean Values"
```

myDF to date format

```
myDF$Date = as.Date(myDF$Date)
```

Remove the X column from myDF

```
drops <- c("X")
myDF <- myDF[ , !(names(myDF) %in% drops)]
```

UI of the Application

```
# Uses the fluidPage and a Sidebar Layout
ui <- fluidPage(
  title = "Tester",
  titlePanel("Visualization 1"),
  sidebarLayout(
    sidebarPanel(
      # This panel contains the first Data Table
      conditionalPanel(
        #input.dataset is used in DT's to show which data is actually used
        'input.dataset === "myDF"',
        #Included a checkbox for what columns should be in the data Table
        #The pre selected columns are "Date" and "Steps"
        checkboxGroupInput("show_vars", "Columns of information to show:",
                           names(myDF), selected = list("Date", "Steps"))
      ),
      # This panel contains the second Data Table
```

```
      conditionalPanel(
        #input.dataset is used in DT's to show which data is actually used
        'input.dataset === "finalDataFrame"',
        #We can change this text to anything we want user to see
        helpText("Click the column header to sort a column.")
      ),
    ),
    mainPanel(
      # Allows the user to pick between the 2 tabs containing the Data Tables
      tabsetPanel(
        #name of the tabs
        id = 'dataset',
        #First tab named "myDF" contains Data Table from first set of data
        tabPanel("myDF", DT::dataTableOutput("mytable1")),
        #Second tab named "finalDataFrame" contains DT from second set of data
        tabPanel("finalDataFrame", DT::dataTableOutput("mytable2"))
      )
    )
  )
)
```

Server of the Application

```
server <- function(input, output) {
  # choose columns to display
  myDF2 = myDF[sample(nrow(myDF), nrow(myDF), replace = FALSE), ]
  #Used to render the actual Data Table based off any changes made by the user
  #Changes based on what variables are selected and the filter feature is
  #located on the top of the Data Table
  output$mytable1 <- DT::renderDataTable({
    DT::datatable(myDF2[, input$show_vars, drop = FALSE], rownames = FALSE, filter = 'top')
  })

  #Used to render teh Data Table and does not have many features like the first
  #Data Table. This one only contains the means (finalDataFrame)
  output$mytable2 <- DT::renderDataTable({
    DT::datatable(finalDataFrame, options = list(orderClasses = TRUE), rownames = FALSE)
  })
}
```

Runs the application

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

Visulization 2

In this visulization, we categorizes the data into three categories: Steps and Floors Climbed, Distance, and Activity Level. Each category contains bar graphs that display the data for each respective variable per month. Both the Distance and the Activity Level categories display visualizations that separate the data into different subcategories based on the intensity of the activity.

Code

```
library(shiny)
library(ggplot2)
library(reshape2)
# biometric data
# 'bioFinal.csv' for biometric data from 2020 Feb. to present? - Merckteam's data
# 'DrWardData.csv' for biometric data from 2016 to 2019 April - Dr. Ward's data

#This function reads in the CSV file that contains the data, named DrWardData.csv, and stores it in a d
myDF <- read.csv("/class/datamine/corporate/merck/Merck201920/fitbit_data/DrWardData.csv")

# The subset function selects the data from the myDF data frame that separates the miles the user has w
# The melt function then takes the columns stored in miles1 and stacks them into a single column in "mi
# The as.Date function goes into the Date column of "miles" and correctly formats the values as a date.
miles1 <- subset(myDF, select = c('Date','Lightly.Active.Miles', 'Moderately.Active.Miles', 'Very.Activ
miles <- melt(miles1)
miles$Date <- as.Date(miles$Date)

# The subset function selects the data from the myDF data frame that separates the minutes the user spe
# The melt function then takes the columns stored in minutes1 and stacks them into a single column in "
# The as.Date function goes into the Date column of "minutes" and correctly formats the values as a dat
minutes1 <- subset(myDF, select = c('Date', 'Sedentary.Minutes', 'Lightly.Active.Minutes', 'Fairly.Acti
minutes <- melt(minutes1)
minutes$Date <- as.Date(minutes$Date)

# The subset function selects the data from the myDF data frame that separates the data by the number o
# The melt function then takes the columns stored in hr1 and stacks them into a single column in "hr".
# The as.Date function goes into the Date column of "hr" and correctly formats the values as a date.
hr1 <- subset(myDF, select = c('Date', 'HR.30.100.Minutes', 'HR.100.140.Minutes', 'HR.140.170.Minutes',
hr <- melt(hr1)
hr$Date <- as.Date(hr$Date)

# The as.Date function goes into the Date column of "myDF" and correctly formats the values as a date.
# The first format function goes into the year column of "myDF" and formats the values as a year.
# The second format function goes into the month column of "myDF" and formats the values as unabbreviat
myDF$Date = as.Date(myDF$Date)
myDF$year <- format(myDF$Date,'%Y')
myDF$month <- factor(format(myDF$Date,'%B'), levels = month.name)

# Every shiny app must include a UI header beginning with fluidPage()
# fluidPage() allows the user to adjust the size of their window while preserving the dimensions of the
ui<-fluidPage(
  # The first section of code uses the navbarPage() function to create the menu at the top of the page

  # Creates a menu of selections titled "Fitbit Dashboard"
  navbarPage("Fitbit Dashboard",
             # tabPanel 1

             # The tabPanel() function creates the "How to Use" tab in the Fitbit Dashboard and include
             tabPanel("How to Use",
                      # Includes Merck logo
                      img(src = "https://assets.phenompeople.com/CareerConnectResources/MERCUS/social/1
```

```
                # Creates main title "Biometric Data Dashboard"
                h1("Biometric Data Dashboard"),
                mainPanel(
                  # Includes bolded text and link
                  strong("Refer to the website here for more information:"),
                  a("https://www.merck.com/index.html"),
                  # Smaller heading than h1 to create title "How to Use My Dashboard"
                  h3("How to Use My Dashboard"),
                  # Outputs HTML "text2" which is included in the server block
                  htmlOutput("text2")
                )
        ),

        # tabPanel 2

        # The tabPanel() function creates the "Activity Progress" tab in the Fitbit Dashboard and
        tabPanel("Activity Progress",
                 sidebarLayout(
                   # Creates title "Activity Progress" for sidebar panel
                   sidebarPanel(h3("Activity Progress"),
                                # Allows user to input date range
                                dateRangeInput("dates", ("Date range")),
                                # Allows user to select single y-axis variable for graph
                                varSelectInput("variables", ("Variable:"), myDF[c(-1, -2)]),
                                # Allows user to select variable y-axis (includes all levels of se
                                selectInput("stack", ("Variable (Stacked):"),
                                            c("N/A" = "nana", "Active Miles" = "miles", "Active mi
                   ),
                   # Creates the main panel
                   mainPanel(
                     # Includes italicized text
                     em("Select a date range and a variable!"),
                     # Includes bolded text
                     strong("Refer to the website here for more information:"),
                     # Includes link
                     a("https://www.merck.com/index.html"),
                     # Displays the bar graph with selected variable
                     plotOutput(outputId = "bioBarGraph")
                   )
                 )
        ),

        # tabPanel 3

        # The tabPanel() function creates the "Step Goals" tab in the Fitbit Dashboard and include
        tabPanel("Step Goals",
                 # Creates sidebar panel with smaller heading "10,000-Step Goal"
                 sidebarPanel(h3("10,000-Step Goal"),
                              # Allows user to input one specific date
                              dateInput("goaldate", ("Date"))
                 ),
                 # Creates the main panel
                 mainPanel(
```

```
                    # Includes bolded text
                    strong("Refer to the website here for more information:"),
                    # Includes link
                    a("https://www.merck.com/index.html"),
                    # Displays pie chart
                    plotOutput(outputId = "bioDoughnut")
                  )
          ),
          # tabPanel 4

          # The tabPanel() function creates the "Visualization 2" tab in the Fitbit Dashboard and in
          tabPanel("Visualization 2",
                   sidebarLayout(
                    # Creates title "Activity Progress" for sidebar panel
                    sidebarPanel(h3("Visualization 2"),
                    ),
                    # Creates the main panel
                    mainPanel(
                      # Includes bolded text
                      strong(HTML("Activity graphs are listed below <br/>")),
                      strong("Steps and Floors Climbed"),
                      # Displays the bar graph that shows total steps per month
                      plotOutput(outputId = "bioStep"),
                      # Displays the bar graph that shows total floors climbed per month
                      plotOutput(outputId = "bioFloors"),
                      # Includes bolded text
                      strong(HTML("Distance <br/>")),
                      # Includes italicized text
                      em("This section shows the total distance walked per month in miles, as well a
                      # Displays the bar graph that shows total miles walked per month
                      plotOutput(outputId = "bioTM"),
                      # Displays the bar graph that shows total number of lightly active miles walk
                      plotOutput(outputId = "bioLM"),
                      # Displays the bar graph that shows total number of moderately active miles w
                      plotOutput(outputId = "bioMM"),
                      # Displays the bar graph that shows total number of very active miles walked
                      plotOutput(outputId = "bioVM"),
                      # Includes bolded text
                      strong(HTML("Activity Level <br/>")),
                      # Includes italicized text
                      em("This section shows the time spent in each level of activity."),
                      # Displays the bar graph that shows the amount of time spent being lightly ac
                      plotOutput(outputId = "bioLA"),
                      # Displays the bar graph that shows the amount of time spent being fairly act
                      plotOutput(outputId = "bioFA"),
                      # Displays the bar graph that shows the amount of time spent being very activ
                      plotOutput(outputId = "bioVA"),
                    )
                   )
          )
  )
)
```

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

  # You can access the values of the widget (as a vector of Dates)
  # with input$dates,
  # e.g. output$value <- renderPrint({ input$dates })

  # text on tabPanel 1
  output$text2 <- renderUI({
    HTML(paste("", "1. Activity Progress with Bar Charts",
               "- Date Range: select a start and an end date to set a date range of the activity progres
               "- Variables: use the drop-down to select a variable of the activity progress you wish to
               "- Variables (Stacked): use the drop-down to select a variable of the activity progress y
               "", "2. Step Goals with Doughnut Charts",
               "- Date: select a date of the activity progress you wish to see",
               sep="<br/>"))
  })

  # bar charts on tabPanel 2
  output$bioBarGraph <- renderPlot({

    # counter for the start date - date range
    counter1 = 0
    for (myDate in as.character(myDF$Date)){
      counter1 = counter1 + 1
      if (myDate == input$dates[1]) {
        break
      }
    }

    # counter for the end date - date range
    counter2 = 0
    for (myDate in as.character(myDF$Date)){
      counter2 = counter2 + 1
      if (myDate == input$dates[2]) {
        break
      }
    }

    # subset of myDF with the selected date range
    mySubset <- myDF[c(counter1:counter2),]
    if (input$stack == "nana"){ # single bar chart - if stacked bar is N/A
      ggplot(data=mySubset, aes(x=Date, y=!!input$variables)) + geom_bar(stat="identity", width = .5, fi
    }else if(input$stack == "miles"){ # stacked bar chart - active miles
      ggplot(data=miles, aes(x=Date, y=value, fill=variable)) + geom_bar(stat="identity", width = .5)
    }else if(input$stack == "minutes"){ # stacked bar chart - active minutes
      ggplot(data=minutes, aes(x=Date, y=value, fill=variable)) + geom_bar(stat="identity", width = .5)
    }else if(input$stack == "hr"){ # stacked bar chart - HR minutes
      ggplot(data=hr, aes(x=Date, y=value, fill=variable)) + geom_bar(stat="identity", width = .5)
    }
  })
  # Visualization 2
  # Bar graph that shows total steps per month
  output$bioStep <- renderPlot({
```

```
  ggplot(data = myDF, aes(x = month, y= Steps))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Total Steps per Month") +
    xlab("Month") +
    ylab("Steps")
})

# Bar graph that shows total floors climbed per month
output$bioFloors <- renderPlot({
  ggplot(data = myDF, aes(x = month, y= Floors.Climbed))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Total Floors Climbed per Month") +
    xlab("Month") +
    ylab("Floors Climbed")
})

# Bar graph that shows total miles walked per month
output$bioTM <- renderPlot({
  ggplot(data = myDF, aes(x = month, y= Total.Miles))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Total Miles Walked per Month") +
    xlab("Month") +
    ylab("Miles")
})

# Bar graph that shows lightly active miles walked per month
output$bioLM <- renderPlot({
  ggplot(data = myDF, aes(x = month, y= Lightly.Active.Miles))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Lightly Active Miles Walked per Month") +
    xlab("Month") +
    ylab("Miles")
})

# Bar graph that shows moderately active miles walked per month
output$bioMM <- renderPlot({
  ggplot(data = myDF, aes(x = month, y= Moderately.Active.Miles))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Moderately Active Miles Walked per Month") +
    xlab("Month") +
    ylab("Miles")
})

# Bar graph that shows very active miles walked per month
output$bioVM <- renderPlot({
  ggplot(data = myDF, aes(x = month, y= Very.Active.Miles))+
    geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6)
    ggtitle("Very Active Miles Walked per Month") +
    xlab("Month") +
    ylab("Miles")
})

# Bar graph that shows the amount of time spent being lightly active per month
```

```
  output$bioLA <- renderPlot({
    ggplot(data = myDF, aes(x = month, y= Lightly.Active.Minutes))+
      geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6
      ggtitle("Time Spent Being Lightly Active per Month") +
      xlab("Month") +
      ylab("Minutes")
  })

  # Bar graph that shows the amount of time spent being fairly active per month
  output$bioFA <- renderPlot({
    ggplot(data = myDF, aes(x = month, y= Fairly.Active.Minutes))+
      geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6
      ggtitle("Time Spent Being Fairly Active per Month") +
      xlab("Month") +
      ylab("Minutes")
  })

  # Bar graph that shows the amount of time spent being very active per month
  output$bioVA <- renderPlot({
    ggplot(data = myDF, aes(x = month, y= Very.Active.Minutes))+
      geom_bar(stat = "identity", color = "#007a73", width = 0.5, position = position_dodge(width = 0.6
      ggtitle("Time Spent Being Very Active per Month") +
      xlab("Month") +
      ylab("Minutes")
  })


  # doughnut chart on tabPanel 3
  output$bioDoughnut <- renderPlot({

    # counter for the date - daily goals
    counter3 = 0
    for (myDate in as.character(myDF$Date)){
      counter3 = counter3 + 1
      if (myDate == input$goaldate) {
        break
      }
    }

    # doughnut bar
    # daily steps of the selected date
    steps <- myDF[c(counter3:counter3),c("Steps")]
    # get a percentage - daily steps out of 10000 steps
    data <- data.frame(
      category=c("Steps","N/A"),
      count=c(steps, 10000-steps)
    )
    data$fraction = data$count / 100
    data$ymax = cumsum(data$fraction)
    data$ymin = c(0, head(data$ymax, n=-1))
    ggplot(data, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) + geom_rect() + coord_polar(
  })
}
```

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

Critical decisions

Our group decided to use RStudio to test pieces of R code on our own local machines, but we ran into the problem of not being able to collaborate with our team in an efficient way, so we looked for an alternative method where we could all have access to update a file. We decided to create a repository on bitbucket where we all had access to the files and could update everything using git.

After setting up the colloboration, we started working to improve the previous dashboard and implementing new feature. To improve our skills in R shiny dashboard, we each implemented a different form of data visualization, including a histogram, a scatterplot, a box plot, and a pie chart, and implemented them using data from the previous year. Additionally, we looked at code from the previous year's team and tried to run their app and resolve any issues that occurred.

While implementing the visulization, we faced few difficuluties. For instance, in the viuslization 1, we have to modify the whole visualization as our initial visulization was having issue to modify itself during the user interaction. Simialarly, second visulization was interactive initially and allowed the user to choose which variable they wanted to see on the graph. However, an issue we faced with this was that when the user selected a variable, the values on the y-axis would become N/A and the bars on the graph would all be the same height. After discussing several potential solutions to this issue, we decided that the best option would be to create several static graphs, allowing the user to scroll to the desired graph.

Technical Report

During the last sprint, we focused mainly on the visualization and refinement stages of our development process. In the visualization stage, we brainstormed, discussed, and developed the code for the two visualizations we created. In the refinement stage, we reviewed the issues in the code and discussed potential solutions. We then cycled back to the visualization stage in order to implement the solutions we discussed.