dmungekar

Devvrat Mungekar

2023-04-17

Introduction:

With advancements in technology, it is now possible to create web apps that provide real-time data, maps, and graphs. Programming languages are used to speed up servers since these services have significant processing and synchronization needs. A R package called Shiny enables users to create interactive web applications. This program converts Shiny code into a web application that is HTML-compatible. To develop a visually appealing application, we combine R Shiny functions with native HTML and CSS code. Shiny mixes the R's processing capability with the current web's interaction. We have the option of utilizing your server or R Shiny's hosting services to deploy the web apps that Shiny creates. R's Shiny can be used to make web apps without JavaScript, automatically "live" in the same sense that spreadsheets are "live." An automated update is initiated by input modifications and is effective in any R setting. It has the Twitter Bootstrap-based attractive default UI theme. It is used to plot, table, and printed output widgets for R objects that are pre-built and programmable.

Description of the Libraries:

1. shiny- With the help of the online app framework Shiny, users may create interactive web applications using R. It provides a wide range of tools and user interface components for building responsive and dynamic web apps.

2. ggplot2- ggplot2 is a well-liked data visualization toolkit that provides a high-level, flexible interface for creating aesthetically beautiful and instructive presentations. It is based on the concept of the "Grammar of Graphics" and allows for the production of a broad variety of static and dynamic representations.

3. dplyr- The dplyr data manipulation package makes data purging, transformation, and summarization straightforward. It is commonly used for data wrangling tasks including data filtering, data organization, and data aggregation.

4. shinydashboard- Shinydashboard is a package that provides a framework for creating responsive and aesthetically appealing dashboards in Shiny projects. It offers a variety of user interface (UI) components, such as sidebars, boxes, and tabs, that are specifically designed for the creation of dashboards.

5. DT: A R package for creating dynamic, customisable data tables. It provides a variety of options for web-based sorting, filtering, and presenting data tables.

6. sninyWidgets- Shiny may now use additional UI elements including sliders, drop-down menus, and file uploaders thanks to the package shinyWidgets. It facilitates the development of interactive, user-friendly internet applications.

7. plotly- Plotly is a library for building dynamic and interactive infographics as well as animated, 3D plots and static charts. It has interactive capabilities like panning, zooming, and hovering and supports a variety of display formats.

8. magnittr- For combining various data transformation and manipulation tasks, the Magrittr library offers a choice of comprehensible operators. It enables the creation of data wrangling programs that are shorter and simpler to read.

9. scales- Scales is a tool for scaling and formatting pictures. It also includes labeling, changing axis boundaries, and formatting numbers. To increase readability and attractiveness, visualizations might be changed in appearance.

10. tinytex- LaTeX is a well-liked typesetting program for producing documents with a professional appearance. Tinytex is a software for typesetting LaTeX text. It offers resources for creating LaTeX documents inside of R and configuring LaTeX settings.

11. knitr- knitr is a library that combines text, code, and graphics to produce dynamic reports and publications. It allows for the quick updating of articles with fresh data or analytical findings.

```
library(shiny)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(shinydashboard)
```

```
##
## Attaching package: 'shinydashboard'

## The following object is masked from 'package:graphics':
##
##     box
```

```
library(DT)
```

```
##
## Attaching package: 'DT'

## The following objects are masked from 'package:shiny':
##
##     dataTableOutput, renderDataTable
```

```
library(shinyWidgets)
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(magrittr)
library(scales)
library(tinytex)
library(knitr)
```

Dataset Description:

```
# Load the superstore data
superstore <- read.csv("/Users/devvratmungekar/Library/CloudStorage/OneDrive-Personal/Canada/Trent Univ
```

The dataset is taken from Kaggle and is labelled under the Superstore. It has a total of 24 columns.

Column name	Column Description
RowID	The unique identifier of the row data.
OrderID	The unique identifier of the order data.
OrderDate	The date when the sales transaction was completed.
ShipDate	The date when order was shipped to the customer.
ShipMode	The category by which the shipment was sent to the customer.
CustomerID	A unique identifier of each customer.
CustomerName	The name of the customer.
Segment	The segment in which a customer belongs, e.g., consumer, corporate, or home office.
PostalCode	The postal code of the customer.
City	The city in which the customer lives.
State	The state in which the customer lives.
Country	The country in which the customer lives.
Region	The region where the sales transaction took place.
Market	The region where the sales transaction took place.
ProductID	The unique identifier of the product.
Category	The category in which the product belongs, for e.g., technology, furniture, or office supplies.
Sub-Category	The sub-category in which the product belongs, for e.g., phones, bookcases, tables, etc.

Column name	Column Description
ProductName	The name of the product sold.
Sales	The total sales amount in each transaction.
Quantity	The quantity of product sold in each transaction.
Discount	The discount received on the sale of each product.
Profit	The profit made on the sales of each product.
ShippingCost	The shipping cost of the product to the customer.
OrderPriority	The priority of the order in shipping the product to the customer.

Purpose:

The superstore dataset will be visualized to derive insights and information from the data by creating visual representations that emphasize patterns, trends, and relationships within the dataset. A more approachable and understandable method to examine, analyze, and display complex data is through the use of charts, graphs, and other visualization tools.

The purpose of the dataset is to summarize the following:
1. Data Visualization: Visualization can assist in exploring the data to comprehend its structure, distribution, and features. For instance, bar charts may show the distribution of product categories or clientele, scatter plots can highlight trends in geographic data, and heatmaps can highlight relationships between variables like sales and profit.
2. Data Analysis: By making it possible to spot patterns, trends, and anomalies in the data, visualization may help in data analysis. Line charts, box plots, and tree maps, for instance, can show how sales have changed over time, the distribution and variability of numerical data, and the breakdown of sales by area, category, and product, respectively. 3. Decision-making: Visualization may enhance data-driven decision-making by offering visual clues and insights. Dashboards may show a visual summary of key performance indicators (KPIs), geographic maps can show which locations have the best or worst sales performance, and funnel charts can show how sales are converted.
4. Communication: Visualization may aid stakeholders in understanding the data findings by making complex data more understandable and accessible. Visualizations may be used in presentations, reports, and dashboards to convey insights, trends, and patterns to non-technical stakeholders.

The superstore dataset might be visualized to facilitate data analysis and decision making, uncover important insights, enhance data results communication, and ultimately enhance comprehension and utilization of the data.

Description of the Libraries:

1. shiny- With the help of the online app framework Shiny, users may create interactive web applications using R. It provides a wide range of tools and user interface components for building responsive and dynamic web apps.

2. ggplot2- ggplot2 is a well-liked data visualization toolkit that provides a high-level, flexible interface for creating aesthetically beautiful and instructive presentations. It is based on the concept of the "Grammar of Graphics" and allows for the production of a broad variety of static and dynamic representations.

3. dplyr- The dplyr data manipulation package makes data purging, transformation, and summarization straightforward. It is commonly used for data wrangling tasks including data filtering, data organization, and data aggregation.

4. shinydashboard- Shinydashboard is a package that provides a framework for creating responsive and aesthetically appealing dashboards in Shiny projects. It offers a variety of user interface (UI) components, such as sidebars, boxes, and tabs, that are specifically designed for the creation of dashboards.

5. DT: A R package for creating dynamic, customisable data tables. It provides a variety of options for web-based sorting, filtering, and presenting data tables.

6. sninyWidgets- Shiny may now use additional UI elements including sliders, drop-down menus, and file uploaders thanks to the package shinyWidgets. It facilitates the development of interactive, user-friendly internet applications.

7. plotly- Plotly is a library for building dynamic and interactive infographics as well as animated, 3D plots and static charts. It has interactive capabilities like panning, zooming, and hovering and supports a variety of display formats.

8. magnittr- For combining various data transformation and manipulation tasks, the Magrittr library offers a choice of comprehensible operators. It enables the creation of data wrangling programs that are shorter and simpler to read.

9. scales- Scales is a tool for scaling and formatting pictures. It also includes labeling, changing axis boundaries, and formatting numbers. To increase readability and attractiveness, visualizations might be changed in appearance.

10. tinytex- LaTeX is a well-liked typesetting program for producing documents with a professional appearance. Tinytex is a software for typesetting LaTeX text. It offers resources for creating LaTeX documents inside of R and configuring LaTeX settings.

11. knitr- knitr is a library that combines text, code, and graphics to produce dynamic reports and publications. It allows for the quick updating of articles with fresh data or analytical findings.

```
# Define the UI
ui <- dashboardPage(
  dashboardHeader(
    title= "Superstore Sales Analysis"
  ),

  dashboardSidebar(
    dateRangeInput('dateRange',
                   label = 'Date Range',
                   start = as.Date('2012-01-01'), end = as.Date('2015-12-31')),

    selectInput(inputId = "category",
                label = "Select Product Category:",
                choices = unique(superstore$Category))

  ),

  dashboardBody(
    fluidRow(
      valueBoxOutput("RetailSales"),
      valueBoxOutput("ProfitMargin"),
      valueBoxOutput("CustomersVisited")
    ),

    fluidRow(
      box(
```

```
        title = "Sales & Profit by Category", background = "maroon", solidHeader = TRUE, width = 12, col
      )
    ),

    fluidRow(
      box(
        title = "Sales by Sub-Category", background = "navy", solidHeader = TRUE, width = 12, collapsibl
      )
    ),

    fluidRow(
      box(
        title = "State-wise Shipping Cost", background = "fuchsia", solidHeader = TRUE, width = 12, coll
      )
    ),


  ) #dashboardBody
) # dashboardPage

# Define the server
server <- function(input, output) {
  # Filter the data based on user inputs

  # Sales
  output$RetailSales <- renderValueBox({
    deliveredToDate <- superstore %>%
      filter(OrderDate <= input$dateRange[2] & OrderDate >= input$dateRange[1])

    valueBox(
      paste0("$", prettyNum(ceiling(sum(deliveredToDate$Sales)), big.mark = ",")),
      "Total Retail Sales",
      color = "teal",
      icon = icon("dollar-sign")
    )
  }) # Sales

  #Profit
  output$ProfitMargin <- renderValueBox({
    deliveredToDate <- superstore %>%
      filter(OrderDate <= input$dateRange[2] & OrderDate >= input$dateRange[1])

    valueBox(
      paste0("$", prettyNum(ceiling(sum(deliveredToDate$Profit)), big.mark = ",")),
      "Profit Margin",
      color = "fuchsia",
      icon = icon("dollar-sign")
    )
  }) # Profit

  # Total Customers Visited
  output$CustomersVisited <- renderValueBox({
    UserToDate <- superstore %>%
```

```
      filter(OrderDate <= input$dateRange[2] & OrderDate >= (input$dateRange[2] - 30))
    valueBox(
      prettyNum(length(unique(UserToDate$OrderID)),big.mark = ","),
      "Customers",
      color = "light-blue",
      icon = icon("user-alt")
    )
  }) # Total Customers Visited

  # Sales & Profit by Category
  output$CategorySalesPlot <- renderPlotly({
    dailyRetail <- superstore %>%
      filter(OrderDate <=input$dateRange[2] & OrderDate >= (input$dateRange[2] - 30) &
      Category %in% input$category)

    cateSal <- ggplot(dailyRetail, aes(x = Sales, y = Profit)) + geom_point(color= 'blue', alpha= .6, na
    ggplotly(cateSal)
  }) # Sales & Profit by Category

  # Sales by Sub-Category
  output$SubSales <- renderPlotly({
    dailyRetail <- superstore %>%
      filter(OrderDate <=input$dateRange[2] & OrderDate >= (input$dateRange[2] - 30) &
      Category %in% input$category)

    SubSale <- ggplot(dailyRetail, aes(x= factor(Sub.Category), y= Sales)) + geom_boxplot(aes(fill= fac
    ggplotly(SubSale)
  }) # Sales by Sub-Category

  #State-wise Shipping Cost
  output$StateShippingCost <- renderPlotly({
    dailyRetail <-superstore %>%
      filter(OrderDate <=input$dateRange[2] & OrderDate >= (input$dateRange[2] - 30) &
      Category %in% input$category)

    ShipState <- ggplot(dailyRetail, aes(x= State, y= ShippingCost)) + geom_violin(alpha= .6) + labs(x=
    ggplotly(ShipState)
  })
} # end

# Run the Shiny app
shinyApp(ui = ui, server = server)
```

Link to the application:

The application is uploaded in the R Shiny's online portal known as the Shinyio. The link to the application is https://dmungekar.shinyapps.io/superstore/

Results:

The above shiny app is used to analyze and visualized the sales data of the superstore and is utilized to make informed decisions about the company's different path and decisions. The Shiny API for R has the power to make some beautiful dashboards that can be shared to show the analysis and the work of the data scientist and the analyst to the world. Using such a powerful tool helps the statisticans to showcase their work on the global platform or within a company in a very intuitive manner.

Reference:

1. https://shiny.rstudio.com/tutorial/

2. https://towardsdatascience.com/beginners-guide-to-creating-an-r-shiny-app-1664387d95b3

3. https://medium.com/edureka/r-shiny-tutorial-47b050927bd2

4. https://anishsinghwalia.medium.com/developing-web-apps-in-r-using-shiny-f868f51a4122