Food Hygiene Rating Scheme

Pei Yu Lin

Get data from the website through data scraping

```r
# Fetch the page
# https://data.food.gov.uk/catalog/datasets/38dd8d6a-5ab1-4f50-b753-ab33288e3200
# And store it into a variable
data.food.gov.uk.url <-
  "https://data.food.gov.uk/catalog/datasets/38dd8d6a-5ab1-4f50-b753-ab33288e3200"
```

```r
#Read the html link and store the script into a variable
data.food.gov.uk.pages <- read_html(data.food.gov.uk.url)
data.food.gov.uk.pages
```

```
## {html_document}
## <html lang="en">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
## [2] <body class="theme--data">\n    <div class="cc_banner-wrapper" id="global ...
```

```r
# To filter out the data that is not in Welsh language, we need to find the
# individual local authority name
individual_name <- data.food.gov.uk.pages %>%
  html_nodes(".c-dataset-element__item-content") %>%
  html_nodes("h2") %>%
  html_text()

# To know the file type of each file link from this page,
# we need to find the url type
individual_type <- data.food.gov.uk.pages %>%
  html_nodes(".c-dataset-element__item-content") %>%
  html_nodes(".o-dataset-distribution--link")%>%
  html_text()

# To get a url for each file url from this page, we need to use html_attr('href')
# to get the link and store them into a variable (individual_urls)
individual_urls <- data.food.gov.uk.pages %>%
  html_nodes(".c-dataset-element__distribution-files")%>%
  html_nodes(".o-dataset-distribution--link") %>%
  html_attr('href')

# To filter out the individual_type which is application/xml, we have to merge
# variables individual_type and individual_urls into one dataframe
individual_type_table <- data.frame(individual_type)
individual_urls_table <- data.frame(individual_urls)
individual_name_table <- data.frame(individual_name)
```

```r
link_data <- cbind(individual_name,individual_type_table, individual_urls_table)

# Use filter() to filter out the rows data which is application/xml
# and not in Welsh language type of data
link_data %>% filter(individual_type == "application/xml",
                     !grepl("Welsh language", individual_name)) -> link_data

summary(link_data)
```

```
##  individual_name    individual_type    individual_urls
##  Length:376         Length:376         Length:376
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

```r
# Create a directory to store files
dir.create("FHRS_data")

# Use 'for' loop to download xml files from web to the directory
for (i in 1:nrow(link_data)){
  download.file(link_data$individual_urls[i], destfile =
  paste0(
    "..//..//Data_Management/Assignment/FHRS_data/local_authority", i, ".xml"))
}
```

```r
# Using 'for' loop to read all the downloaded files at one go
local_authority_df <- data.frame()

# Use 'for' loop to get data in multi-level nested files
for(i in 1:nrow(link_data)){
  local_authority_parse <-
   xmlParse(
    paste0(
     "..//..//Data_Management/Assignment/FHRS_data/local_authority", i, ".xml"))
  # Get data from each children
   local_authority_df_new <- xmlToDataFrame(
     nodes = getNodeSet(local_authority_parse,
                        "//EstablishmentDetail | //Geocode | //Scores"))
   # bind the new data into the whole dataframe
  local_authority_df <- rbind.fill(local_authority_df_new, local_authority_df)
}

# To save time for code running in the future, we save the large data in local
saveRDS(local_authority_df, "local_authority_df.rds")
```

```r
# Read in the local file saved previously
local_authority_df <- readRDS("local_authority_df.rds")

# Delete columns that are not used
local_authority_df$Geocode <- NULL
local_authority_df$Scores <- NULL

# Create a dataframe for Longitude and Latitude
```

```r
geo <- select(local_authority_df, Longitude, Latitude)

# Delete the extra rows
geo <- geo[-c(1,2),]

# Create a dataframe for Hygiene, Structural and ConfidenceInManagement
scores <- select(local_authority_df, Hygiene, Structural, ConfidenceInManagement)

# Delete the extra rows
scores <- scores[-1,]
scores <- head(scores, -1)

# Select the columns from the original dataframe except Longitude,
# Latitude, Hygiene, Structural and ConfidenceInManagement
LA_df <- local_authority_df[,!names(local_authority_df) %in% c("Longitude",
                "Latitude", "Hygiene", "Structural", "ConfidenceInManagement")]

#Delete the extra rows
LA_df <- head(LA_df, -2)

#Combine three dataframes into one dataframe
LA_df <- cbind(LA_df, geo, scores)

#Set a function to delete the rows that have NA for every column
removeRowsAllNa  <- function(x){x[apply(x, 1, function(y) any(!is.na(y))),]}
LA_df <- removeRowsAllNa(LA_df)
```

```r
# To further analyze the data, filter data into two groups. One for Rating stamp
# (Rating value:0,1,2,3,4,5), another group is for certain circumstances.
rating_value_num <- LA_df %>%
  filter(RatingValue == "0"|RatingValue == "1"|RatingValue == "2"|
           RatingValue == "3"|RatingValue == "4"|RatingValue == "5")

rating_value_chr <- LA_df %>% filter(RatingValue != "0"&RatingValue != "1"&
    RatingValue != "2"&RatingValue != "3"&RatingValue != "4"&RatingValue != "5")

# Inspect whether value of rating_value is uniform
unique(rating_value_chr$RatingValue)
unique(rating_value_num$RatingValue)


# Clean Data: To gain uniform rating_value in character,
# replace the original one with a space bewteen the texts
rating_value_chr$RatingValue <- rating_value_chr$RatingValue%>%
  str_replace_all(c("AwaitingInspection" = "Awaiting Inspection",
                    "AwaitingPublication" = "Awaiting Publication"))

#Reorder the columns by putting important information in the front columns
rating_value_chr <- rating_value_chr %>% select(LocalAuthorityName,BusinessName,
                  BusinessType, RatingValue, Longitude, Latitude, everything())

rating_value_num <- rating_value_num %>% select(LocalAuthorityName,BusinessName,
                  BusinessType, RatingValue, Longitude, Latitude, everything())
```

```
#download them into csv file for further use
write_csv(rating_value_num, "data_rating_num.csv")
write_csv(rating_value_chr, "data_rating_chr.csv")
```

Present the data in a Shiny dashboard

```
#Read food hygiene rating data
data_rating_chr <- read_csv("data_rating_chr.csv")
```

```
## Rows: 141970 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (15): LocalAuthorityName, BusinessName, BusinessType, RatingValue, Loca...
## dbl   (4): Longitude, Latitude, FHRSID, BusinessTypeID
## lgl   (5): NewRatingPending, RightToReply, Hygiene, Structural, ConfidenceIn...
## date  (1): RatingDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_rating_num <- read_csv("data_rating_num.csv")
```

```
## Rows: 455013 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (14): LocalAuthorityName, BusinessName, BusinessType, LocalAuthorityBus...
## dbl   (8): RatingValue, Longitude, Latitude, FHRSID, BusinessTypeID, Hygiene...
## lgl   (2): NewRatingPending, RightToReply
## date  (1): RatingDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Shiny UI
ui <- dashboardPage(

  dashboardHeader(title = "Food Hygiene"),

  dashboardSidebar(

    sidebarMenu(
      menuItem("Data Overview", tabName = "food_hygiene_data",
               icon = icon("bookmark")),
      menuItem("Pie Chart", tabName = "pie_chart", icon = icon("chart-pie")),
      menuItem("Bar Chart", tabName = "bar_chart", icon = icon("chart-bar")),
      menuItem("Map", tabName = "map", icon = icon("globe-europe"))
    )
  ),

  dashboardBody(
    tabItems(
      # Item 1: Data overview -----------------------------------------------
      tabItem(tabName = "food_hygiene_data",
              fluidRow(column(width = 12,
                              h1("Data Overview"))
              ),
              fluidRow(
                tabBox(width = 12,
                       tabPanel("Data Rating by Number",
                                DT::dataTableOutput("by_num"),
                                style = "overflow-y: scroll"
                       ),

                       tabPanel("Data Rating by Character",
                                DT::dataTableOutput("by_chr"),
                                style = "overflow-y: scroll"
                       )
                )#end of tabBox
              )
      ),#end of Item 1


      # Item 2: Pie chart ---------------------------------------------------
      tabItem(tabName = "pie_chart",
              fluidRow(column(width = 12,
                              h1("Pie Chart for Rating Value"))),
              fluidRow(
                box(width = 12,
                    solidHeader = TRUE,
                    status = "success",
                    selectInput("BusinessType1",
                                "Business Type:",
                                choices = c("All",
                                levels(as.factor(data_rating_num$BusinessType))
                                )
                    ),#end of select1
                    selectInput("LocalAuthorityName1",
                                "Local Authority Name:",
                                choices = c("All",
                          levels(as.factor(data_rating_num$LocalAuthorityName))
                                )
                    ),#end of select2
                    plotlyOutput("by_num_chr_pie")
                )#end of box 1
              )#end of fluidRow2
      ),#end of Item 2


      # Item 3: Bar chart ---------------------------------------------------
      tabItem(tabName = "bar_chart",
              fluidRow(column(width = 12,
                              h1("Bar Chart"))
              ),
              fluidRow(box(width = 8,
                           status = "primary",
                           plotlyOutput("barplot")
              ),
              box(width = 4,
                  solidHeader = TRUE,
                  status = "primary",
                  title = "Option box",
                  selectInput("LocalAuthorityName3",
                                "Local Authority Name:",
                                choices = c("All",
                          levels(as.factor(data_rating_num$LocalAuthorityName)),
                          levels(as.factor(data_rating_chr$LocalAuthorityName)))
                    ),
                  selectInput("RatingValue1",
                              "Rating Value:",
                              choices = c(
                                levels(as.factor(data_rating_num$RatingValue)),
                                levels(as.factor(data_rating_chr$RatingValue)))
                  )# end of select
              )#end of box
              )
      ),#end of Item 3


      # Item 4: Map ---------------------------------------------------------
      tabItem(tabName = "map",
              div(class = "outer",
                  tags$style(type = "text/css",
                             "#map {height: calc(100vh - 80px) !important;}"),
                  leafletOutput("map", width = "100%", height = "100%"),
                  absolutePanel(id = "controls", class = "panel panel-default",
                                fixed = TRUE, draggable = TRUE, top = 80,
                                right = 20, width = 300, height = "auto",
                                bottom = "auto",
                                column(width = 12,
                                       h2("Options"),
                                       selectInput("RatingValue2",
                                          "Select Rating Value:", choices = c(
                                  levels(as.factor(data_rating_num$RatingValue)),
                                  levels(as.factor(data_rating_chr$RatingValue)))
                                  ),# end of selectInput
                                       sliderInput(inputId = "date2",
                                            label = "Select rating date range:",
                                            min = as.Date("2000-01-01"),
                                            max = as.Date("2021-12-03"),
                                            value = c(as.Date("2010-01-01"),
                                                      as.Date("2021-12-03"))
                                       )#end of sliderUnput
                                )#end of column
                  )#end of absolutePanel
              )
      )#end of Item 4
    )#end of tabItems
  )#end of dashboardBody
)#end of dashboardPage
```

From the Shiny dashboard, the following points could be observed

1. **Tables: Show all data scraped from the food hygiene rating scheme**
2. **Pie chart: Show the food hygiene rating distribution of any type of business in any country**
3. **Bar chart: Show the business types of a certain rating in a specific local authority**
4. **Map: Show the geographic distribution of different ratings and their changes over time.**

```
# Read food hygiene rating data
my_connection <- RSQLite::dbConnect(RSQLite::SQLite(), "food_hygiene.db")
num_rating <- readr::read_csv("data_rating_num.csv")
```

```
## Rows: 455013 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (14): LocalAuthorityName, BusinessName, BusinessType, LocalAuthorityBus...
## dbl   (8): RatingValue, Longitude, Latitude, FHRSID, BusinessTypeID, Hygiene...
## lgl   (2): NewRatingPending, RightToReply
## date  (1): RatingDate
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
chr_rating <- readr::read_csv("data_rating_chr.csv")
```

```
## Rows: 141970 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (15): LocalAuthorityName, BusinessName, BusinessType, RatingValue, Loca...
## dbl   (4): Longitude, Latitude, FHRSID, BusinessTypeID
## lgl   (5): NewRatingPending, RightToReply, Hygiene, Structural, ConfidenceIn...
## date  (1): RatingDate
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
RSQLite::dbWriteTable(my_connection, "num", num_rating, overwrite=TRUE)
RSQLite::dbWriteTable(my_connection, "chr", chr_rating, overwrite=TRUE)
dbDisconnect(my_connection)

# Shiny Sever

server <- function(input, output) {

  # Connect to the database and get the data from the tables
  sqlitePath <- "./food_hygiene.db"
  my_connection <- dbConnect(SQLite(), sqlitePath)
  # Show all data scraped from the  food hygiene rating scheme in two tables
  ## Output all data
  chr_data <- dbGetQuery(my_connection, 'SELECT * FROM chr;')
  num_data <- dbGetQuery(my_connection, 'SELECT * FROM num;')

  ## Put the data into tables
  output$by_num <- DT::renderDT({
    DT::datatable(num_data,
                  filter = "top",
                  class = "cell-border stripe",
    )
  })

  output$by_chr <- DT::renderDT({
    DT::datatable(chr_data,
                  filter = "top",
                  class = "cell-border stripe",
    )
  })



# Show the food hygiene rating distribution of any type of business in any country.
  ## Set colors
  colors <- c("#8B6969","#EEB4B4", "#FFE4C4","#8EE5EE","#00C5CD","#36648B")

  ## For data of rating values in numeric, select the useful attributes
  ## for showing the plot when input is "All"
  BT_all_sql <- dbGetQuery(my_connection,
  ### Convert RatingValue data in the num table to varchar
  ### Combine rows of num and chr tables
  ### Group by RatingValue to get the proportion of each rating value
  "SELECT BusinessType, CAST(RatingValue AS VARCHAR) AS RatingValue,
   LocalAuthorityName, count(*) AS amount
   FROM num AS n
   GROUP BY n.RatingValue
   UNION ALL
   SELECT BusinessType, RatingValue, LocalAuthorityName, count(*) AS amount
   FROM chr AS c
   GROUP BY c.RatingValue;")

  output$by_num_chr_pie <- renderPlotly({

    ## If the input is not "All", select by Business Type & Local Authority Name
    if(input$BusinessType1 != "All" & input$LocalAuthorityName1 != "All"){
      BT_LA_glue_sql <- glue_sql(
      ### Convert RatingValue data in the num table to varchar
      ### Filter data based on the choice selected in the dashboard
      ### Combine rows of num and chr tables
      ### Group by RatingValue to get the proportion of each rating value
      "SELECT BusinessType, CAST(RatingValue AS VARCHAR) AS RatingValue,
      LocalAuthorityName, count(*) AS amount
      FROM num AS n
      WHERE n.BusinessType = ?
      AND n.LocalAuthorityName = ?
      GROUP BY n.RatingValue
      UNION ALL
      SELECT BusinessType, RatingValue, LocalAuthorityName, count(*) AS amount
      FROM chr AS c
      WHERE c.BusinessType = ?
      AND c.LocalAuthorityName = ?
      GROUP BY c.RatingValue;")
      BT_LA_sql <- dbSendQuery(my_connection, BT_LA_glue_sql)
      ### Input the choice from the dashboard and get the output shown in pie chart
      dbBind(BT_LA_sql, list(input$BusinessType1, input$LocalAuthorityName1,
                             input$BusinessType1, input$LocalAuthorityName1))
      pie_chart <- dbFetch(BT_LA_sql)
    }
    ## Select by Business Type
    else if(input$BusinessType1 != "All"){
      BT_glue_sql <- glue_sql(
      ### Convert RatingValue data in the num table to varchar
      ### Filter data based on the choice selected in the dashboard
      ### Combine rows of num and chr tables
      ### Group by RatingValue to get the proportion of each rating value
      "SELECT BusinessType, CAST(RatingValue AS VARCHAR) AS RatingValue,
      LocalAuthorityName, count(*) AS amount
      FROM num AS n
      WHERE n.BusinessType = ?
      GROUP BY n.RatingValue
      UNION ALL
      SELECT BusinessType, RatingValue, LocalAuthorityName, count(*) AS amount
      FROM chr AS c
      WHERE c.BusinessType = ?
      GROUP BY c.RatingValue;")
      BT_sql <- dbSendQuery(my_connection, BT_glue_sql)
      ### Input the choice from the dashboard and get the output shown in pie chart
      dbBind(BT_sql, list(input$BusinessType1, input$BusinessType1))
      pie_chart <- dbFetch(BT_sql)
    }

    ## Select by Local Authority Name
    else if(input$LocalAuthorityName1 != "All"){
      LA_glue_sql <- glue_sql(
      ### Convert RatingValue data in the num table to varchar
      ### Filter data based on the choice selected in the dashboard
      ### Combine rows of num and chr tables
      ### Group by RatingValue to get the proportion of each rating value
      "SELECT BusinessType, CAST(RatingValue AS VARCHAR) AS RatingValue,
      LocalAuthorityName, count(*) AS amount
      FROM num AS n
      WHERE n.LocalAuthorityName = ?
      GROUP BY n.RatingValue
      UNION ALL
      SELECT BusinessType, RatingValue, LocalAuthorityName, count(*) AS amount
      FROM chr AS c
      WHERE c.LocalAuthorityName = ?
      GROUP BY c.RatingValue;")
      LA_sql <- dbSendQuery(my_connection, LA_glue_sql)
      ### Input the choice from the dashboard
      ### and get the output shown in pie chart
      dbBind(LA_sql, list(input$LocalAuthorityName1, input$LocalAuthorityName1))
      pie_chart <- dbFetch(LA_sql)
    }else{
      pie_chart <- BT_all_sql
    }

    ## Draw pie chart
    g1 <- plot_ly(pie_chart, values = ~amount, labels = ~RatingValue,
                  marker = list(colors = colors,
                                line = list(color = '#FFFFFF', width = 1)
                  )
    ) %>%
      add_pie(hole = 0.4)
    g1

  })



# Show the business types of a certain rating in a specific local authority
  output$barplot <- renderPlotly({
    ## Get data if the choice of local authority is "All"
    if(input$LocalAuthorityName3 == "All"){
      ### Filter data based on the selection of RatingValue
      data_num_bar_glue_sql <- glue_sql("SELECT BusinessType, RatingValue,
                                  RatingDate
                                  FROM num
                                  WHERE RatingValue = ?;")
      data_chr_bar_glue_sql <- glue_sql("SELECT BusinessType, RatingValue,
                                  RatingDate
                                  FROM chr
                                  WHERE RatingValue = ?;")

      data_num_bar_sql <- dbSendQuery(my_connection, data_num_bar_glue_sql)
      ### Input the choice selected from the dashboard
      ### and get the output to show in bar chart
      dbBind(data_num_bar_sql, list(input$RatingValue1))
      data_num_bar <- dbFetch(data_num_bar_sql)

      data_chr_bar_sql <- dbSendQuery(my_connection, data_chr_bar_glue_sql)
      ### Input the choice selected from the dashboard
      ### and get the output to show in bar chart
      dbBind(data_chr_bar_sql, list(input$RatingValue1))
      data_chr_bar <- dbFetch(data_chr_bar_sql)

    }else{
      ## Get data if the choice of local authority is not "All"
      ### Filter data based on LocalAuthorityName and RatingValue.
      data_num_bar_glue_sql <- glue_sql("SELECT BusinessType, RatingValue,
                                  LocalAuthorityName
                                  FROM num
                                  WHERE LocalAuthorityName = ?
                                  AND RatingValue = ?;")
      data_chr_bar_glue_sql <- glue_sql("SELECT BusinessType, RatingValue,
                                  LocalAuthorityName
                                  FROM chr
                                  WHERE LocalAuthorityName = ?
                                  AND RatingValue = ?;")

      data_num_bar_sql <- dbSendQuery(my_connection, data_num_bar_glue_sql)
      ### Input the choices selected from the dashboard
      ### and get the output to show in bar chart
      dbBind(data_num_bar_sql, list(input$LocalAuthorityName3, input$RatingValue1))
      data_num_bar <- dbFetch(data_num_bar_sql)

      data_chr_bar_sql <- dbSendQuery(my_connection, data_chr_bar_glue_sql)
      ### Input the choices selected from the dashboard
      ### and get the output to show in bar chart
      dbBind(data_chr_bar_sql, list(input$LocalAuthorityName3, input$RatingValue1))
      data_chr_bar <- dbFetch(data_chr_bar_sql)
    }

    # Bind the data from the num and chr outputs based on the previous step
    data_bar <- rbind(data_num_bar, data_chr_bar)

    # Show the result in bar chart
    g3 <- ggplot(data_bar, aes(x = fct_rev(fct_infreq(BusinessType)))) +
      geom_bar() + coord_flip()
    plotly::ggplotly(g3)

  })


# Show the geographic distribution of different ratings and their changes over time.
  output$map <- renderLeaflet({
    ## To show data in the map, we also need to select latitude and longitude
    ## Filter data based on RatingDate and Rating Value for the observation
    data_map_num_glue_sql <- glue_sql("SELECT RatingValue, Longitude,
                                      Latitude, RatingDate
                                      FROM num
                                      WHERE RatingValue = ?
                                      AND RatingDate >= ?
                                      AND RatingDate <= ?;")

    data_map_chr_glue_sql <- glue_sql("SELECT RatingValue, Longitude, Latitude,
                                      RatingDate
                                      FROM chr
                                      WHERE RatingValue = ?
                                      AND RatingDate >= ?
                                      AND RatingDate <= ?;")
```

```r
    data_map_num_sql <- dbSendQuery(my_connection, data_map_num_glue_sql)
    ## Input the choices selected from the dashboard
    ## and get the output to present data on map
    dbBind(data_map_num_sql, list(input$RatingValue2, input$date2[1],
                                  input$date2[2]))
    data_map_num <- dbFetch(data_map_num_sql)

    data_map_chr_sql <- dbSendQuery(my_connection, data_map_chr_glue_sql)
    ## Input the choices selected from the dashboard
    ## and get the output to present data on map
    dbBind(data_map_chr_sql, list(input$RatingValue2, input$date2[1],
                                  input$date2[2]))
    data_map_chr <- dbFetch(data_map_chr_sql)

    # Bind the data from the num and chr outputs based on the previous step
    data_map <- rbind(data_map_num, data_map_chr)

    # Adjust the setting of map display
    leaflet(data_map) %>%
      addTiles() %>%
      addProviderTiles(providers$CartoDB.Positron) %>%
      fitBounds(~-9,50,~9,58) %>%
      addCircles(color = "#045a8d",
                 fillOpacity = 0.2,
                 radius = 5000,
                 weight = 1,
      )

  })

}

# Run the application
# shinyApp(ui = ui, server = server)
```