

Creating ShinyR Dashboards using AI

For Malaysian Healthcare Decision-making

Dr. Ahmad Hakiim Jamaluddin
PhD (Mathematics & Statistics), UNSW Sydney

School of Mathematics and Statistics, UNSW Sydney
UNSW Data Science Hub (uDASH)
Department of Mathematics and Statistics, UPM Serdang

NHTA Pre-conference Workshop
29 July 2025 | 2:00 PM - 5:00 PM | PICC Malaysia



- 1 Introduction
- 2 Shiny Fundamentals
- 3 LLM-Powered Development
- 4 Healthcare Dashboard Lab
- 5 Conclusion

Workshop Agenda

Agenda

- 2:00-2:15: Introduction to Shiny & Healthcare Dashboards
- 2:15-2:45: Shiny Fundamentals
- 2:45-3:15: LLM-Powered Development (ChatGPT/Grok/Deepseek)
- 3:15-3:30: Coffee Break
- 3:30-4:50: Healthcare Dashboard Lab
- 4:50-5:00: Q&A & Resources

Why Shiny for Healthcare?

Healthcare Applications

- Patient outcome tracking
- Resource allocation monitoring
- Epidemiological trend analysis
- Clinical trial reporting

Benefits

- Real-time data visualisation
- Interactive decision support
- Customisable for Malaysian healthcare needs

- 1 Introduction
- 2 Shiny Fundamentals
- 3 LLM-Powered Development
- 4 Healthcare Dashboard Lab
- 5 Conclusion

Shiny App Anatomy

Core Components

- **UI (User Interface):** Controls layout, inputs (e.g., shinyWidgets dropdowns)
- **Server:** Handles data processing, reactivity
- **Reactivity:** Automatic UI updates with modular code (e.g., moduleServer)
- **Modularity:** Use reusable UI/server modules for scalability

Healthcare-Ready Components

Inputs

- Date range selectors (e.g., shinyWidgets sliders)
- Patient group filters
- Clinical parameter sliders
- Facility selectors with input validation

Outputs

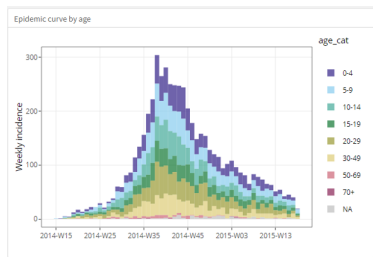
- Interactive epidemic curves (plotly/ggplot2)
- Patient outcome tables (DT)
- Resource utilisation charts
- Geographic heatmaps (leaflet)

Date range

2018-04-15 to 2018-04-16

[1] "2018-04-15" "2018-04-16"

April 2018						
Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5



- 1 Introduction
- 2 Shiny Fundamentals
- 3 LLM-Powered Development**
- 4 Healthcare Dashboard Lab
- 5 Conclusion

LLM Workflow for Shiny

Workflow Steps

- 1 Define clear requirements (objectives, data, visualisations, UI)
- 2 Generate modular code with LLMs (e.g., "ShinyR module for plotly bar chart")
- 3 Optimise prompts with R/Shiny versions, libraries, error handling
- 4 Validate, debug, and enhance interactivity (e.g., shinyWidgets, leaflet)

Effective Prompt Engineering

Good Prompt Structure

“Create a ShinyR app (Shiny 1.8, R 4.4) using `medicaldata::covid_testing`:

- Sidebar with facility dropdown, date range input
- Plotly positivity rate trend chart
- DT table for patient demographics
- Add try-catch for missing data, input validation for positive numbers”

Key Elements

- Specify R/Shiny versions, libraries, dataset structure
- Define visualisations (e.g., plotly, leaflet)
- Include error handling, interactivity
- Use synthetic data for sensitive projects

Hands-on Time!

Steps

- Open RStudio and your preferred LLM (ChatGPT, Grok, Deepseek)
- Use prompt: “Create ShinyR app (Shiny 1.8) with NHANES data, plotly scatterplot of BMI vs. Height, age range slider, DT table, error handling for missing data.”
- Copy, paste, and run the script in RStudio
- Compare outputs from different LLMs

Shiny App Code: Libraries and Data Preparation

```
# Load libraries
library(shiny)
library(NHANES)
library(plotly)
library(DT)
library(dplyr)

# Prepare NHANES data with error handling
data(NHANES)
nhanes_data <- tryCatch({
  NHANES %>%
    select(Age, BMI, Height) %>%
    filter(complete.cases(.)) # Remove rows with missing values
}, error = function(e) {
  message("Error-in-data-preparation:-", e)
  return(NULL)
})
```

Shiny App Code: UI Definition

```
# Define UI with modular structure
ui <- fluidPage(
  titlePanel("NHANES-Dashboard"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("age_range", "Select-Age-Range:",
        min = min(nhanes_data$Age, na.rm = TRUE),
        max = max(nhanes_data$Age, na.rm = TRUE),
        value = c(min(nhanes_data$Age, na.rm = TRUE), max(nhanes_data$Age, na.rm = TRUE)),
        step = 1),
      # Input validation
      tags$script("Shiny.addCustomMessageHandler('alert',function(message){alert(message);});")
    ),
    mainPanel(
      plotlyOutput("scatterPlot"),
      DTOutput("dataTable")
    )
  )
)
```

Shiny App Code: Server Logic and Execution

```
# Define server logic with modular reactivity
server <- function(input, output, session) {
  # Reactive data filtering
  filtered_data <- reactive({
    req(nhanes_data)
    validate(need(input$age_range[1] >= 0, "Age-must-be-positive"))
    nhanes_data %>%
      filter(Age >= input$age_range[1] & Age <= input$age_range[2])
  })

  # Render plotly scatterplot
  output$scatterPlot <- renderPlotly({
    plot_ly(filtered_data(), x = ~Height, y = ~BMI, colour = ~Age, type = "scatter", mode = "markers") %>%
      layout(title = "BMI-vs-Height-by-Age", xaxis = list(title = "Height(cm)", yaxis = list(title = "BMI("
        kg/m^2)"))
  })

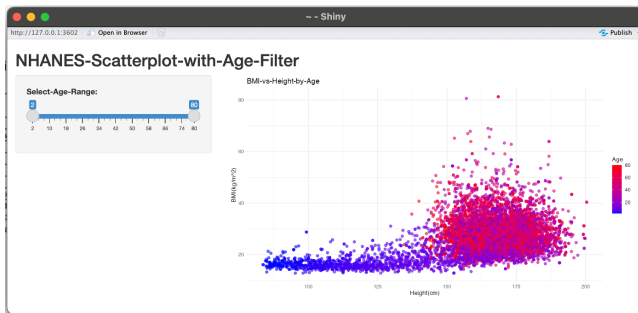
  # Render DT table
  output$dataTable <- renderDT({
    datatable(filtered_data(), options = list(pageLength = 10))
  })
}

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

NHANES Dashboard Example

Interactive Dashboard

Visualises BMI vs. Height with an age range filter and interactive DT table.



Explore live dashboards: [Diabetes & Hypertension Insights](#), [Malaysian News Hub](#)

- 1 Introduction
- 2 Shiny Fundamentals
- 3 LLM-Powered Development
- 4 Healthcare Dashboard Lab
- 5 Conclusion

Lab Exercise: Diabetes Dashboard

Dataset and Requirements

Dataset: *PimaIndiansDiabetes* (via `mlbench` package)

Columns: `pregnant` (times pregnant), `glucose` (mg/dL), `pressure` (mm Hg), `triceps` (mm), `insulin` (mu U/ml), `mass` (BMI), `pedigree`, `age` (years), `diabetes` (pos, neg)

Requirements:

- 1 Age histogram (adjustable bins: 5–20, `ggplot2`)
- 2 Bar chart of patient count by diabetes status
- 3 Box plot of glucose levels by diabetes status
- 4 Scatter plot of glucose vs. BMI, colored by diabetes status
- 5 Filters: Diabetes status dropdown (All, pos, neg), age range slider (0–100)
- 6 Metrics: Total patients, average glucose, % diabetes positive

Lab Exercise: Diabetes Dashboard

LLM Prompt Template

“Create a ShinyR dashboard for the PimaIndiansDiabetes dataset (via mlbench):

- Sidebar with diabetes status dropdown (All, pos, neg), age range slider (0–100)
- Plots: age histogram (bins: 5–20), bar chart of patient count by diabetes status, box plot of glucose by diabetes status, glucose vs. BMI scatter plot coloured by diabetes status
- Metrics: total patients, average glucose, % diabetes positive
- Use shiny, shinydashboard, ggplot2, dplyr, viridis
- Handle missing or invalid data (e.g., NA or zero values) with error messages
- Save filtered dataset as diabetes.csv.”

- 1 Introduction
- 2 Shiny Fundamentals
- 3 LLM-Powered Development
- 4 Healthcare Dashboard Lab
- 5 Conclusion

Key Takeaways

Summary

- ➊ Shiny enables rapid, interactive healthcare dashboards
- ➋ LLMs (ChatGPT, DeepSeek, or Grok) reduce coding barriers
- ➌ Ensure security (local LLMs, synthetic data) and compliance
- ➍ Stay updated with Shiny features, optimise, and test thoroughly

Resources

- **Shiny Basics:** <https://shiny.posit.co/>
- **Healthcare Datasets:**
 - ▶ NHANES: <https://wwwn.cdc.gov/nchs/nhanes/>
 - ▶ MIMIC: <https://mimic.mit.edu/>
 - ▶ medicaldata: <https://higgi13425.github.io/medicaldata/>
- **LLM Platforms:**
 - ▶ ChatGPT: <https://chat.openai.com/>
 - ▶ Grok: <https://grok.com/>
 - ▶ Deepseek: <https://chat.deepseek.com/>
- **Advanced References:**
 - ▶ EpiR Handbook:
https://epirhandbook.com/en/new_pages/shiny_basics.html
 - ▶ ShinyR dashboards: <https://tilburgsciencehub.com/examples>
 - ▶ R packages: <https://cran.r-project.org/web/views/>

Questions are welcomed!