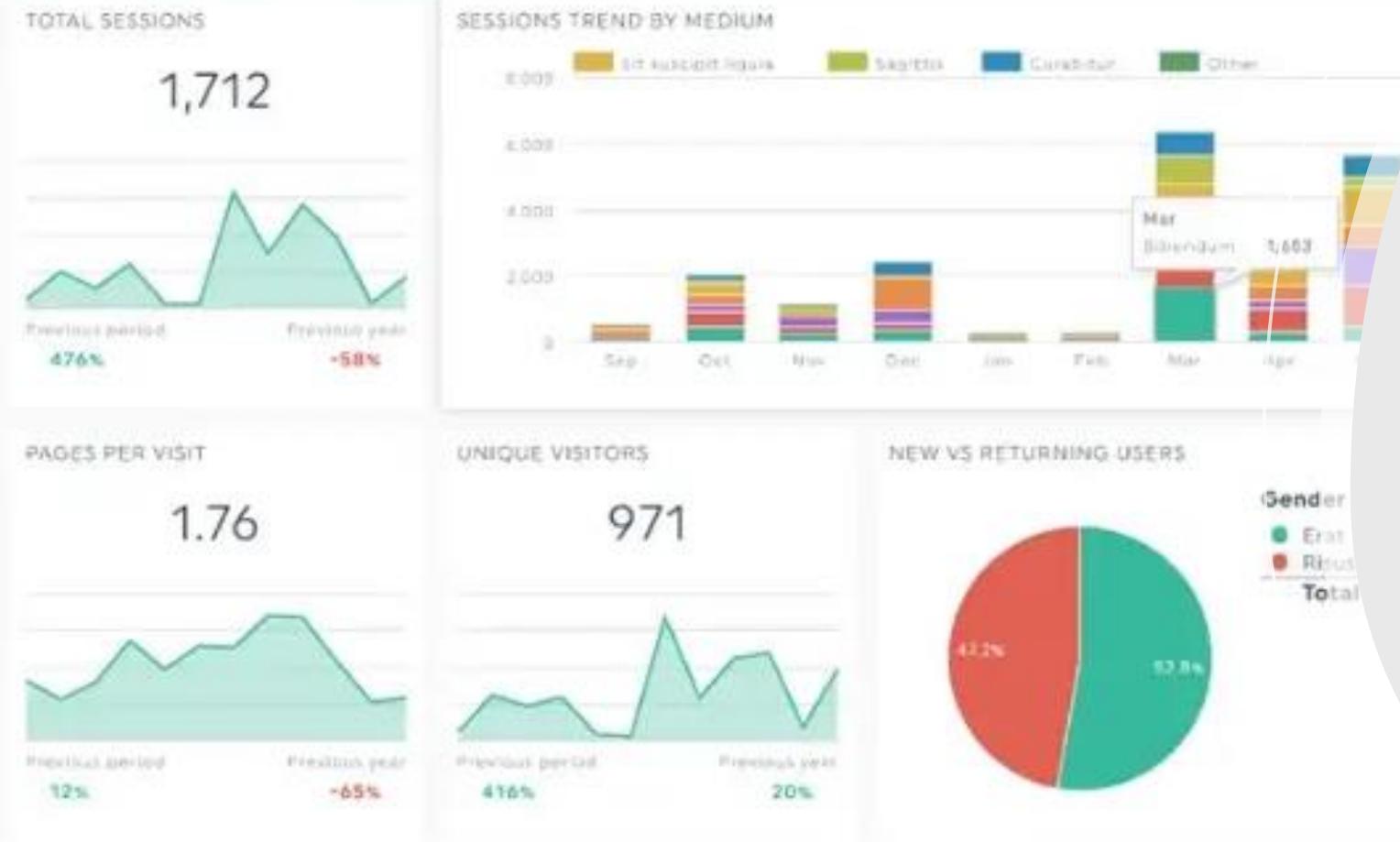




Why Web Visualizations?



Interactivity: Unlike static charts, web visualizations can be dynamic and interactive, allowing users to explore the data on their own.

Accessibility: Web visualizations can be shared easily with a wider audience through a browser.

Real-time updates: With web visualizations, data can be updated in real-time from APIs or databases, which is crucial for live dashboards.



WHO COVID-19 dashboard

WHO Health Emergencies Programme

World ▾

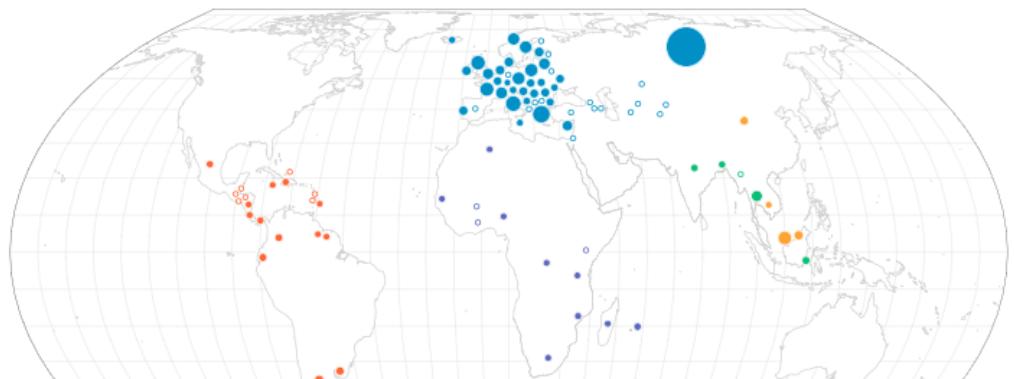
Circulation Cases Deaths Vaccines Variants Wastewater Data More ▾

Last 7 days Last 28 days Total cumulative

Count Rate per 100 000

Number of COVID-19 cases reported to WHO

World, 28 days to 5 January 2025

**161,264** -41,556decrease on previous 28 days
Reported COVID-19 cases

World, 28 days to 5 January 2025

Number of COVID-19 cases reported to WHO

World, 28 days to 5 January 2025

Country Cases ▾

World	161k
Russian Federation	93.5k
Greece	10k

<https://data.who.int/dashboards/covid19/cases>

WHO Regions Africa Americas Eastern Mediterranean Europe South-East Asia Western Pacific

Turks And Caicos Islands	0
Uzbekistan	0



SALES TEAM PERFORMANCE KPIS

Last 30 days (Mar 7 - Apr 5) ▾

Metric	Last 30 Days	Δ
New Contacts	3,316	▲ 5%
New Companies	2,235	▼ 16%
New Deals	61.3	▲ 12%
All Deals	57.4	▼ 4%
Closed Won	59.9	▼ 5%
Closed Lost	18.2	▲ 18%
Avg. time to close	2d 15h	▼ 10%

SALES FUNNEL (1)

Last 30 days (Mar 7 - Apr 5) ▾



SALES REVENUES LEADERBOARD

Last 30 days (Mar 7 - Apr 5) ▾

#	NAME	DEALS	AMOUNT
1	Jane Doe	711	\$7

Sales and Marketing Dashboards



CLOSED WON AMOUNT VS GOAL (2)

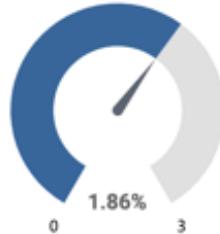
Last 30 days (Mar 7 - Apr 5) ▾

\$0



Financial Dashboards

Current Ratio



DSI



DSO



DPO



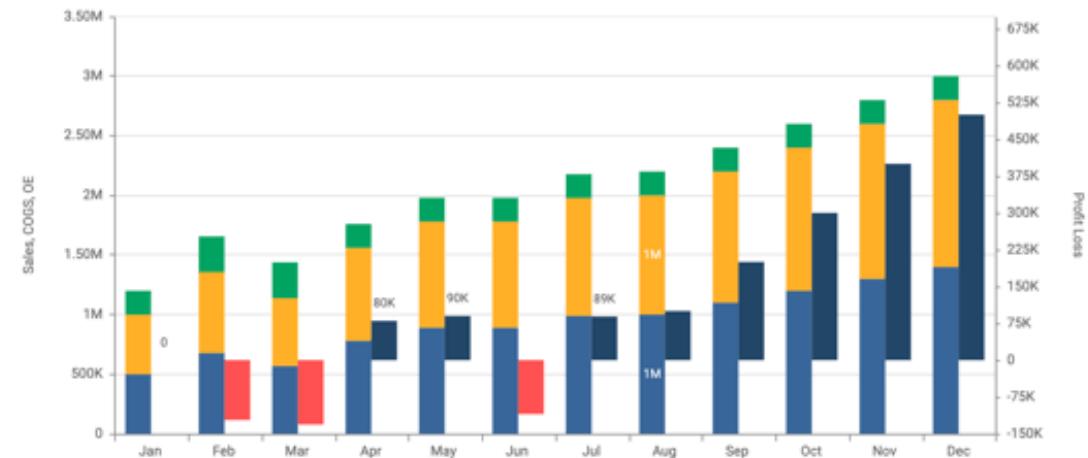
Total Accounts Receivable and Payable Aging



Net Working Capital vs Gross Working Capital



Profit and Loss summary



 Phone

Available
12

In queue

1

In Progress

5

Total

Current week

56

Average wait time

Today

00:20

Oldest missed call

2 days 15:15:52

% good ratings

Current month

91%

Average duration

Current week

22:59

CSAT Score

Current month



Tyler Bryant

65%



Jennifer Mata

64%



Susan Anderson

63%

Current month		
EMPLOYEE	STATUS	COMPLETED
1 Anna Cole	Away	35
2 Curtis Miller	Away	25
3		
4		

Current month		
EMPLOYEE	TOTAL	MISSING
1 Anna Cole	73	36
2 Curtis Miller	63	28
3		
4		

Current month		
EMPLOYEE	OPEN	CLOSED
1 Anna Cole	35	74
2 Curtis Miller	35	80
3		
4		

Customer Service Dashboards

5	Karen Castillo	?	Active	27
6	Kyle Daniels	?	Away	37
7	Mike Novak	?	Away	37

5	Karen Castillo	?	Active	27
6	Kyle Daniels	?	Away	37
7	Mike Novak	?	Away	37

5	Karen Castillo	?	Active	27
6	Kyle Daniels	?	Away	37
7	Mike Novak	?	Away	37
TOTAL			446	914

Instagram



120.5K

Followers

80%

150k

Latest post performance

39K

Impressions

1,600

Likes

217

Comments

4.11%

Engagement rate



Website traffic from Instagram (this month)

212.7K

sessions



▲ Social Media and Brand Monitoring Dashboards

Revenue by promo code (this month)

AXA10

\$7.5K

RODRI10

\$4.6K

MONQUE10

\$3.4K

\$1K

\$0.6K

Tools / Libraries for Web Visualization

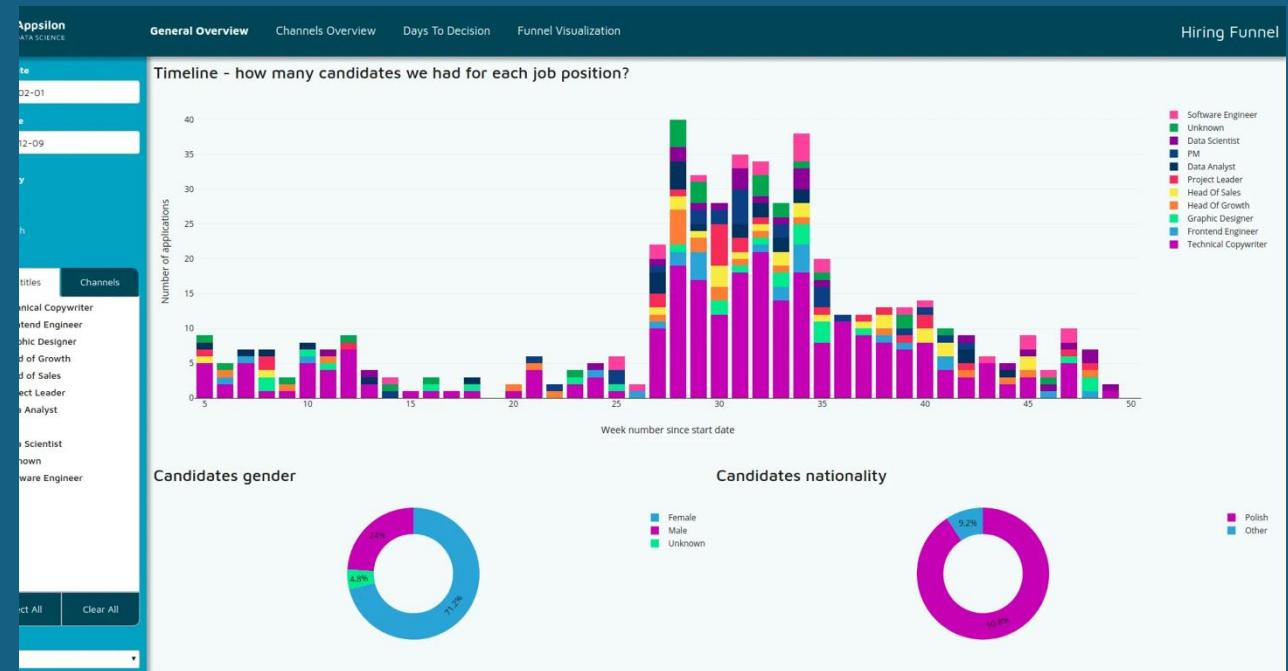


Agenda

- 1. Introduction to Web Visualizations (Shiny)**
- 2. Practice: Create a Dashboard in Shiny**
- 3. Review: Final Exam**
- 4. Outlook Capstone Project**

Shiny

www.rstudio.com



Simplest possible Shiny dashboard

```
library(shiny)

# Define the UI
ui <- fluidPage(
  titlePanel("Hello, Shiny!"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("num", "Choose a number:", min = 1, max = 100, value = 50)
    ),
    mainPanel(
      textOutput("outputText")
    )))
}

# Define the server logic
server <- function(input, output) {
  output$outputText <- renderText({
    paste("You chose the number:", input$num)
  })
}

# Combine the UI and server into the app
shinyApp(ui = ui, server = server)
```

UI (User Interface): Defines what the app looks like (e.g., sliders, buttons, charts).

Server: Handles the logic (e.g., calculations, rendering outputs).

User Interface (UI)

Defines how the dashboard looks and which interactive elements the user can see.

layout fluidPage: Defines the layout of the page

title titlePanel: The title of the dashboard

sidebar sidebarPanel: Input elements such as dropdowns, sliders, etc.

main panel mainPanel: Main area for diagrams, tables or other output

```
ui <- fluidPage(  
  titlePanel("A Dashboard"),  
  sidebarLayout(  
    sidebarPanel(  
      selectInput("variable", "choose variable:", choices = c("Option 1", "Option 2")),  
      sliderInput("range", "value range:", min = 0, max = 100, value = c(10, 50))),  
    mainPanel(  
      plotOutput("plot"),  
      tableOutput("table"))))
```

Server

How the Server Works:

1. The **UI** sends input values (e.g., slider values, text input) to the **server**.
2. The **server** processes these inputs (e.g., calculations, plotting) based on the user's actions.
3. The **server** then generates output values (e.g., a plot, a text message) and sends them back to the **UI** for display.

The server is a **function** that takes two main arguments:

- **input**: This contains all the inputs from the UI (e.g., values from sliders, text boxes, etc.).
- **output**: This contains the outputs that the server will send back to the UI (e.g., plots, tables, or text).

Server

```
library(shiny)

# UI: Define the interface with a slider and a plot output
ui <- fluidPage(
  titlePanel("Dynamic Histogram"),
  sidebarLayout(
    sidebarPanel(
      sliderInput("num", "Choose a number", min = 1, max = 100, value = 50)),
    mainPanel(
      plotOutput("distPlot"))))

# Server: Define the logic that updates the plot based on slider input
server <- function(input, output) {

  # Render the plot dynamically based on slider input
  output$distPlot <- renderPlot({
    # Generate a histogram of random numbers based on the slider input
    hist(rnorm(input$num), main = "Histogram of Random Numbers", xlab = "Value", col = "skyblue")})

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

User Interface (UI): layout or fluidPage

The layout is the overall structure of your UI. In Shiny, `fluidPage()` is commonly used to create a flexible and responsive layout, where elements automatically adjust to the screen size.

European Ski Resorts Dashboard

Minimum Total Slope Length (km):
100

Day Pass Price (EUR):
0 81

Select a Country:
All

Show Resorts with Snowparks
 Show Resorts with Nightski

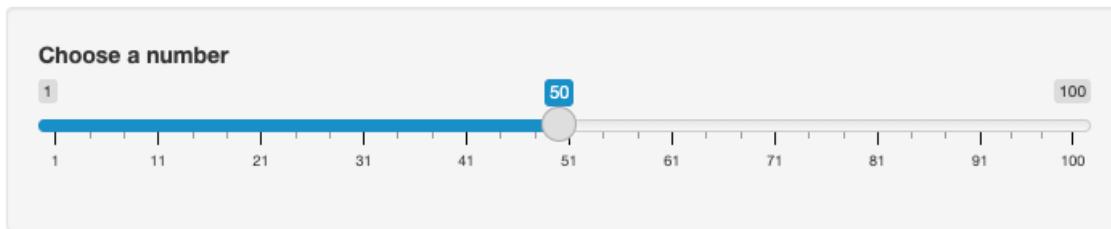
Country	Average_Price	Total_Slope	Number_of_Resorts
Andorra	47.00	630	3
Austria	51.20	4762	25
France	45.78	10405	37
Italy	46.31	2238	13
Spain	46.50	535	4
Switzerland	64.44	3646	16

```
...  
fluidPage(  
  titlePanel("European Ski Resorts Dashboard"),  
  sidebarLayout(  
    sidebarPanel(  
      sliderInput("num", "Select number", min = 1, max = 100, value = 50)),  
    mainPanel(  
      textOutput("result"))))  
...  
...
```

User Interface (UI): layout or fluidPage

The layout is the overall structure of your UI. In Shiny, `fluidPage()` is commonly used to create a flexible and responsive layout, where elements automatically adjust to the screen size.

My Dynamic App



```
# Layout or fluid page
library(shiny)

ui <- fluidPage(
  titlePanel("My Dynamic App"),

  # Define the layout structure
  fluidRow(
    column(4,
      sidebarPanel(
        sliderInput("num",
          "Choose a number",
          min = 1, max = 100,
          value = 50))),
    column(8,
      mainPanel(
        plotOutput("plot"))))

server <- function(input, output) {
  output$plot <- renderPlot({
    hist(rnorm(input$num)) }})

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

User Interface (UI): titlePanel

You can adjust the title by changing the string inside the `titlePanel()` function.

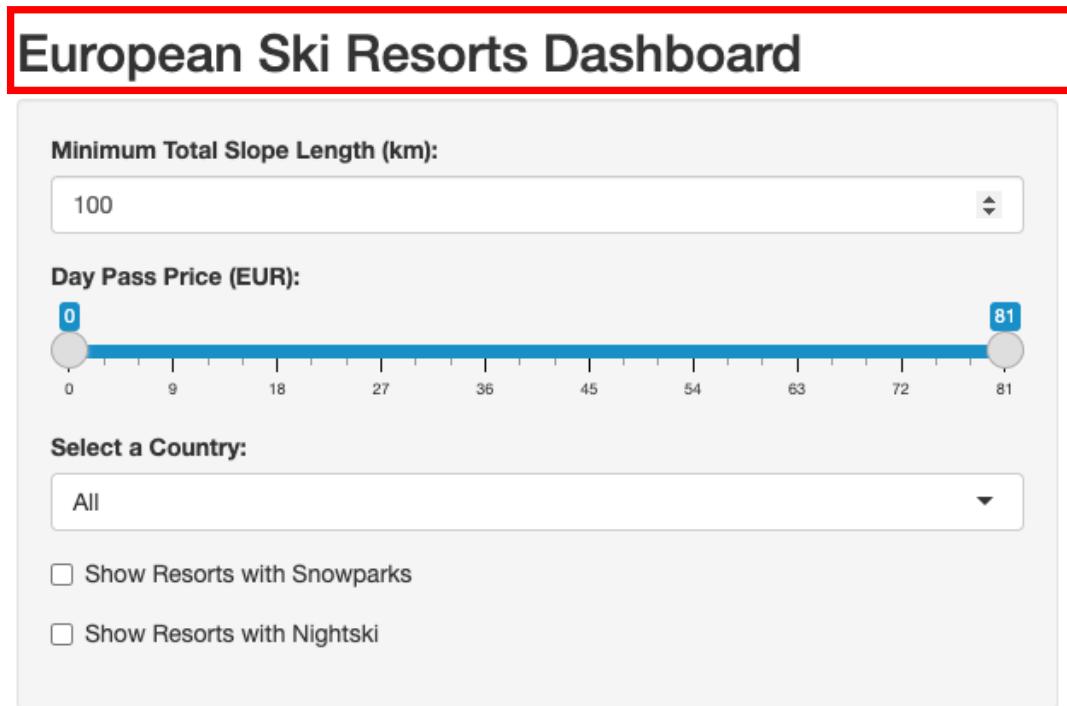
European Ski Resorts Dashboard

Minimum Total Slope Length (km):
100

Day Pass Price (EUR):
0 81

Select a Country:
All

Show Resorts with Snowparks
 Show Resorts with Nightski



Filtered Resorts		Summary by Country	
Country	Average_Price	Total_Slope	Number_of_Resorts
Andorra	47.00	630	3
Austria	51.20	4762	25
France	45.78	10405	37
Italy	46.31	2238	13
Spain	46.50	535	4
Switzerland	64.44	3646	16

```
ui <- fluidPage(  
  titlePanel("European Ski Resorts Dashboard"),  
  ...)
```

User Interface (UI): sidebarPanel

The sidebarPanel is used to create a sidebar in your app where inputs or navigation controls are placed.

European Ski Resorts Dashboard

Minimum Total Slope Length (km):

Day Pass Price (EUR):

Select a Country:

Show Resorts with Snowparks

Show Resorts with Nightski

Filtered Resorts		Summary by Country		
Country	Average_Price	Total_Slope	Number_of_Resorts	
Andorra	47.00	630	3	
Austria	51.20	4762	25	
France	45.78	10405	37	
Italy	46.31	2238	13	
Spain	46.50	535	4	
Switzerland	64.44	3646	16	

```
...
sidebarPanel(
  sliderInput("pricerange", "Day Pass Price(EUR)", value = c(30, 100))
...
)
```

User Interface (UI): mainPanel

The mainPanel is used to display the main content area of the application. This is where the output or results of the user interaction (like plots, tables, or text) will appear.

European Ski Resorts Dashboard

Minimum Total Slope Length (km):
100

Day Pass Price (EUR):
0 to 81

Select a Country:
All

Show Resorts with Snowparks

Show Resorts with Nightski

Filtered Resorts Summary by Country

Country	Average_Price	Total_Slope	Number_of_Resorts
Andorra	47.00	630	3
Austria	51.20	4762	25
France	45.78	10405	37
Italy	46.31	2238	13
Spain	46.50	535	4
Switzerland	64.44	3646	16

```
...
mainPanel(
  tableOutput("result")
)
...
```

User Interface (UI): some more adjustment options

width & height	adjusts the width and height of elements
style	allows to apply custom CSS styles to UI components
tabset &tabpanel	organizes content into tabs, allowing to switch between different sections of your app
wellPanel	Groups elements together in a styled container, typically with a subtle border and padding
collapse	makes a UI component collapsible. This can be used for sidebars or other panels to save space.
offset	adds an offset to a column, shifting it to the right
conditionalPanel	Allows for rendering specific UI components based on user input or other conditions
navbarPage	Creates a navigation bar at the top of the app with tabs to switch between different pages

Load the dataset „European_Ski_Resorts.csv“



Input elements: numericInput

Allows users to filter the data by a given numeric input e.g., based on their desired minimum slope length.

Minimum Total Slope Length (km):										
X	Resort	Country	HighestPoint	LowestPoint	DayPassPriceAdult	BeginnerSlope	IntermediateSlope	DifficultSlope	TotalSlope	
16	Méribel (Les 3 Vallées)	France	3230	1110	61	312	216	72	600	
17	Les Menuires (Les 3 Vallées)	France	3230	1110	61	312	216	72	600	
265	Courchevel (Les 3 Vallées)	France	3230	1110	61	312	216	72	600	
266	Les Gets (Les Portes du Soleil)	France	2466	1000	51	310	210	60	580	

```
ui <- fluidPage(
  numericInput("min_slope", "Minimum Total Slope Length (km):", value = 50, min = 0),
  tableOutput("filtered_resorts"))

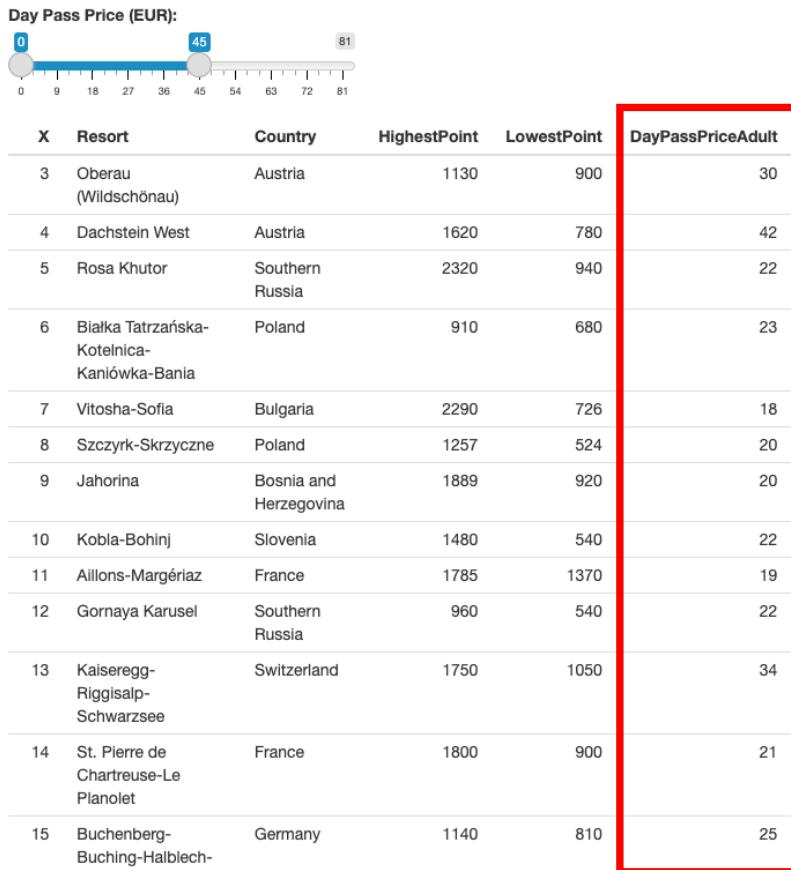
server <- function(input, output) {
  filteredData <- reactive({
    data %>% filter(TotalSlope >= input$min_slope)})

  output$filtered_resorts <- renderTable({
    filteredData()
  })}

shinyApp(ui, server)
```

Input elements: sliderInput

Allows users to filter the data based on a range of the input variable such as e.g. day pass prices



```
ui <- fluidPage(
  sliderInput("price_range", "Day Pass Price (EUR):",
    min = min(data$DayPassPriceAdult, na.rm = TRUE),
    max = max(data$DayPassPriceAdult, na.rm = TRUE),
    value = c(30, 100)),
  tableOutput("price_filtered_resorts")
)

server <- function(input, output) {
  filteredData <- reactive({
    data %>% filter(DayPassPriceAdult >= input$price_range[1],
                      DayPassPriceAdult <= input$price_range[2])
  })

  output$price_filtered_resorts <- renderTable({
    filteredData()
  })
}

shinyApp(ui, server)
```

Input elements: selectInput

Provides a dropdown menu to filter the data based on the selected input.

```
ui <- fluidPage(  
  selectInput("country", "Select a Country:",  
             choices = c("All", unique(data$Country))),  
  tableOutput("country_filtered_resorts"))  
  
server <- function(input, output) {  
  filteredData <- reactive({  
    if (input$country == "All") {  
      data  
    } else {  
      data %>% filter(Country == input$country)  
    }  
  })  
  
  output$country_filtered_resorts <- renderTable({  
    filteredData()  
  })  
}  
  
shinyApp(ui, server)
```

Select a Country:

Germany

X	Resort	Country	HighestPoint	LowestPoint
15	Buchenberg-Buching-Halblech-	Germany	1140	810
128	Zugspitze	Germany	2000	700
142	Garmisch-Classic-Garmisch-Partenkirchen	Germany	2050	732
174	Wendelstein-Brannenburg-Osterhofen	Germany	1723	791
182	Feldberg-Seebuck-Grafenmatt-Fahl	Germany	1448	888
199	Mitterdorf-Almberg--Mitterfirmiansreut	Germany	1139	841
201	Geißkopf-Bischofsmäis	Germany	1116	835
203	Indoor ski area Snow Dome Bispingen	Germany	122	90
204	Sudelfeld-Bayrischzell	Germany	1563	850
205	Grünenlifte-Kranzegg-Rettenberg-	Germany	1700	900
208	Nebelhorn-Oberstdorf	Germany	2224	828

Input elements: checkboxInput

Provides a dropdown menu to filter the data based on the selected input.

```
ui <- fluidPage(
  checkboxInput("show_all", "Show All Resorts", value = TRUE),
  checkboxInput("filter_slope", "Filter by Slope Length >= 100 km", value = FALSE),
  checkboxInput("filter_price", "Filter by Day Pass Price <= 50 EUR", value = FALSE),
  tableOutput("resorts_table"))

server <- function(input, output) {
  filteredData <- reactive({
    df <- data

    if (input$show_all) {
      return(df)
    }

    if (input$filter_slope) {
      df <- df %>% filter(TotalSlope >= 100)

    if (input$filter_price) {
      df <- df %>% filter(DayPassPriceAdult <= 50) }df} )

  output$resorts_table <- renderTable({
    filteredData()})}

shinyApp(ui, server)
```

- Show All Resorts
- Filter by Slope Length >= 100 km
- Filter by Day Pass Price <= 50 EUR

X	Resort	Country	HighestPoint	LowestPoint
1	Alpendorf (Ski amedé)	Austria	1980	740
2	Soldeu-Pas de la Casa/Grau Roig/El Tarter/Canillo/Encamp (Grandvalira)	Andorra	2640	1710
16	Méribel (Les 3 Vallées)	France	3230	1110
17	Les Menuires (Les 3 Vallées)	France	3230	1110

Input elements: radioButtons

Allows the user to select a specific metric to summarize the data

Select a Metric to Summarize:

- Average Price
- Total Slope Length
- Number of Resorts

Country	Value
Andorra	5.00
Austria	89.00
Bosnia and Herzegovina	1.00
Bulgaria	4.00
Czech Republic	2.00
Denmark	8.00
Finland	3.00
France	83.00
Germany	24.00
Greece	1.00

```
ui <- fluidPage(  
  radioButtons("summary_metric", "Select a Metric to Summarize:",  
    choices = c("Average Price" = "price",  
               "Total Slope Length" = "slope",  
               "Number of Resorts" = "count")),  
  tableOutput("summary_table"))  
)  
  
server <- function(input, output) {  
  summaryData <- reactive({  
    data %>%  
      group_by(Country) %>%  
      summarise(Metric = case_when(  
        input$summary_metric == "price" ~ mean(DayPassPriceAdult, na.rm = TRUE),  
        input$summary_metric == "slope" ~ sum(TotalSlope, na.rm = TRUE),  
        input$summary_metric == "count" ~ n())  
      ) %>%  
      rename(Value = Metric)  
  })  
  
  output$summary_table <- renderTable({  
    summaryData()  
  })  
}  
  
shinyApp(ui, server)
```

Input elements: combined example

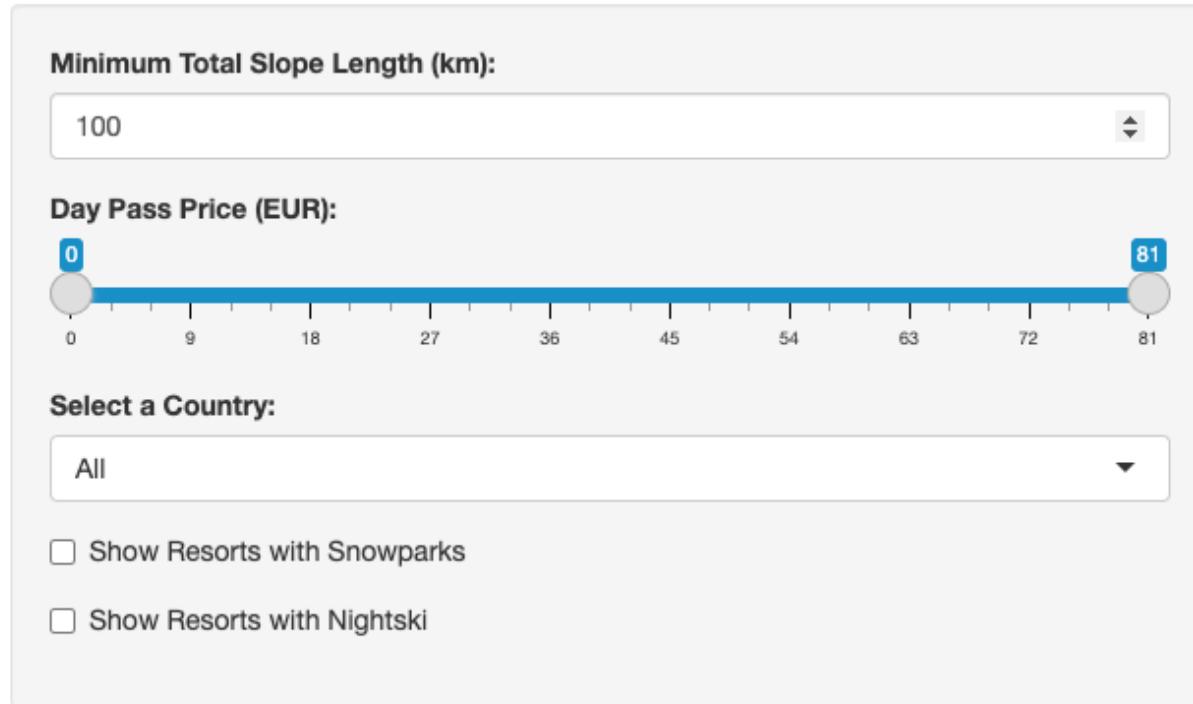
European Ski Resorts Dashboard

Minimum Total Slope Length (km):
100

Day Pass Price (EUR):
0 81

Select a Country:
All

Show Resorts with Snowparks
 Show Resorts with Nightski



Filtered Resorts		Summary by Country	
Country	Average_Price	Total_Slope	Number_of_Resorts
Andorra	47.00	630	3
Austria	51.20	4762	25
France	45.78	10405	37
Italy	46.31	2238	13
Spain	46.50	535	4
Switzerland	64.44	3646	16

Combined Example: Interactive Dashboard with Multiple Inputs

Output elements

Output elements show data visualizations or tables based on user input.

TextOutput Example: Summary of Data

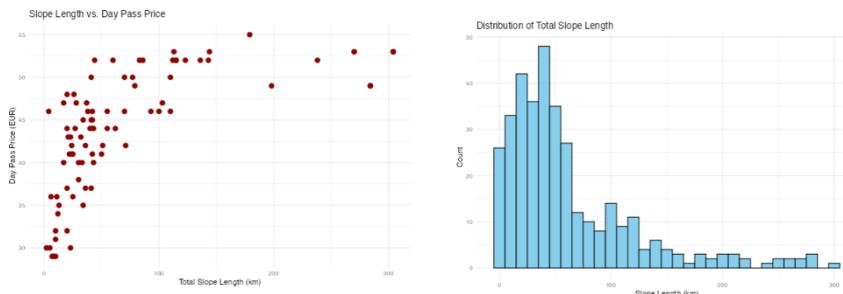
Minimum Total Slope Length (km):

Number of Resorts: 187, Average Slope Length: 149.36 km

X	Resort	Country	HighestPoint	LowestPoint	DayPassPriceAdult	BeginnerSlope	IntermediateSlope	DifficultSlope
16	Méribel (Les 3 Vallées)	France	3230	1110	61	312	216	72
17	Les Menuires (Les 3 Vallées)	France	3230	1110	61	312	216	72
265	Courchevel (Les 3 Vallées)	France	3230	1110	61	312	216	72

`textOutput`: display simple, dynamically generated text

`tableOutput`: is for showing data frames or tables



`plotOutput`: `plotOutput` is used for visualizing data through charts or graphs. Integrates seamlessly with ggplot

Output elements: textOutput

`textOutput` is used to display simple, dynamically generated text. It can be used for summaries, key metrics, or any text-based feedback.

TextOutput Example: Summary of Data



A screenshot of a Shiny application interface. On the left, there is a sidebar panel containing a single input element: a numeric input field labeled "Minimum Total Slope Length (km)" with the value set to 50. To the right of the sidebar, the main content area displays a summary message: "Number of Resorts: 187, Average Slope Length: 149.36 km".

```
text_ui <- fluidPage(
  titlePanel("TextOutput Example: Summary of Data"),
  sidebarLayout(
    sidebarPanel(
      numericInput("min_slope", "Minimum Total Slope Length (km):", value = 50, min = 0)),
    mainPanel(
      textOutput("summary_text"))))

text_server <- function(input, output) {
  output$summary_text <- renderText({
    filtered <- data %>% filter(TotalSlope >= input$min_slope)
    paste0(
      "Number of Resorts: ", nrow(filtered), ", ",
      "Average Slope Length: ", round(mean(filtered$TotalSlope, na.rm = TRUE), 2), " km"
    )}))}

# Run TextOutput App
shinyApp(ui = text_ui, server = text_server)
```

Output elements: tableOutput

Used to display data frames or tables. It's like embedding an Excel table into your dashboard.

X	Resort	Country	HighestPoint	LowestPoint	DayPassPriceAdult	BeginnerSlope	IntermediateSlope	DifficultSlope
16	Méribel (Les 3 Vallées)	France	3230	1110	61	312	216	72
17	Les Menuires (Les 3 Vallées)	France	3230	1110	61	312	216	72
265	Courchevel (Les 3 Vallées)	France	3230	1110	61	312	216	72
266	Les Gets (Les Portes du Soleil)	France	2466	1000	51	310	210	60
267	Avoriaz (Les Portes du Soleil)	France	2466	1000	51	310	210	60
268	Châtel (Les Portes du Soleil)	France	2466	1000	51	310	210	60
281	Saint Martin de Belleville (Les 3 Vallées)	France	3230	1110	61	312	216	72
330	La Tania-Val Thorens/ Les Menuires/ Méribel (Les 3 Vallées)	France	3230	1110	61	312	216	72
371	Val Thorens (Les 3	France	3230	1110	61	312	216	72

```
# Table Output Example with Slider Input
ui <- fluidPage(
  titlePanel("Table Output Example with Slider Input"),
  sidebarLayout(
    sidebarPanel(
      # Slider input for slope length range
      sliderInput("slope_range", "Select Total Slope Length (km):",
                 min = min(data$TotalSlope, na.rm = TRUE),
                 max = max(data$TotalSlope, na.rm = TRUE),
                 value = c(50, 150)),

    mainPanel(
      tableOutput("filtered_table")))

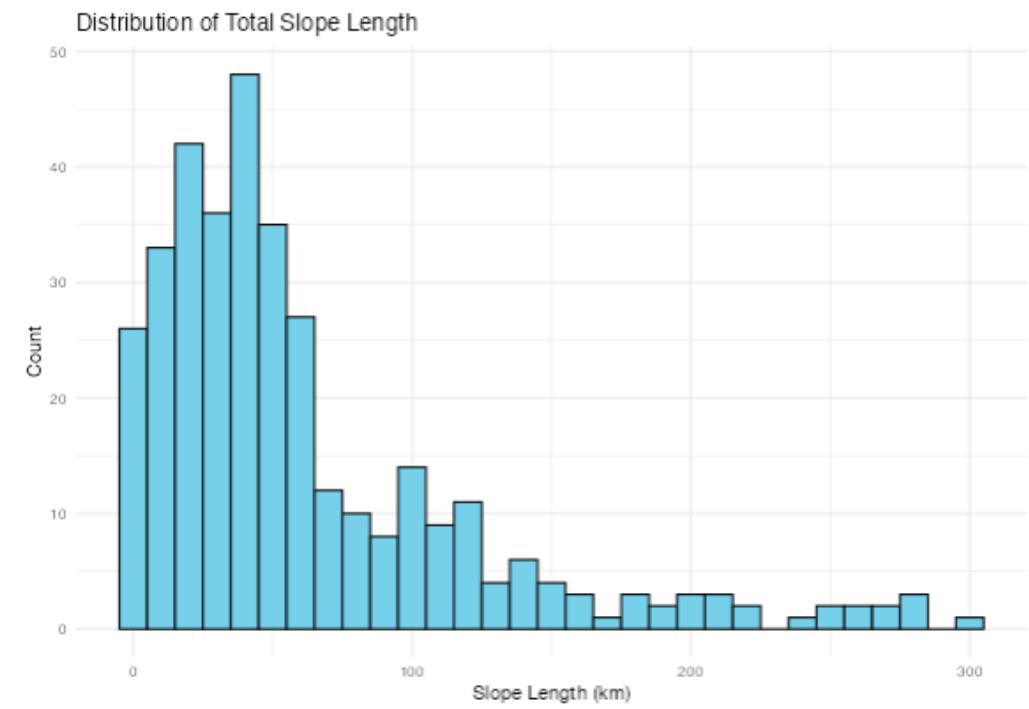
server <- function(input, output) {
  # Reactive filtering based on slider input
  output$filtered_table <- renderTable({
    data %>%
      filter(TotalSlope >= input$slope_range[1],
             TotalSlope <= input$slope_range[2]))}

shinyApp(ui, server)
```

Output elements: plotOutput

Used to display plots (e.g., bar charts, line graphs) created with ggplot2 or base R. This is the “chart area” where you visualize trends and comparisons.

```
histogram_ui <- fluidPage(  
  titlePanel("Histogram: Distribution of Total Slope Length"),  
  sidebarLayout(  
    sidebarPanel(  
      sliderInput("slope_range", "Slope Length Range (km):",  
                  min = min(data$TotalSlope, na.rm = TRUE),  
                  max = max(data$TotalSlope, na.rm = TRUE),  
                  value = c(0, 300))),  
    mainPanel(  
      plotOutput("histogram_plot"))))  
  
histogram_server <- function(input, output) {  
  output$histogram_plot <- renderPlot({  
    filtered <- data %>%  
      filter(TotalSlope >= input$slope_range[1],  
             TotalSlope <= input$slope_range[2])  
  
    ggplot(filtered, aes(x = TotalSlope)) +  
      geom_histogram(binwidth = 10, fill = "skyblue", color =  
      "black") +  
      labs(title = "Distribution of Total Slope Length", x = "Slope  
Length (km)", y = "Count") +  
      theme_minimal()})}  
  
# Run Histogram App  
shinyApp(ui = histogram_ui, server = histogram_server)
```



Output elements: plotOutput

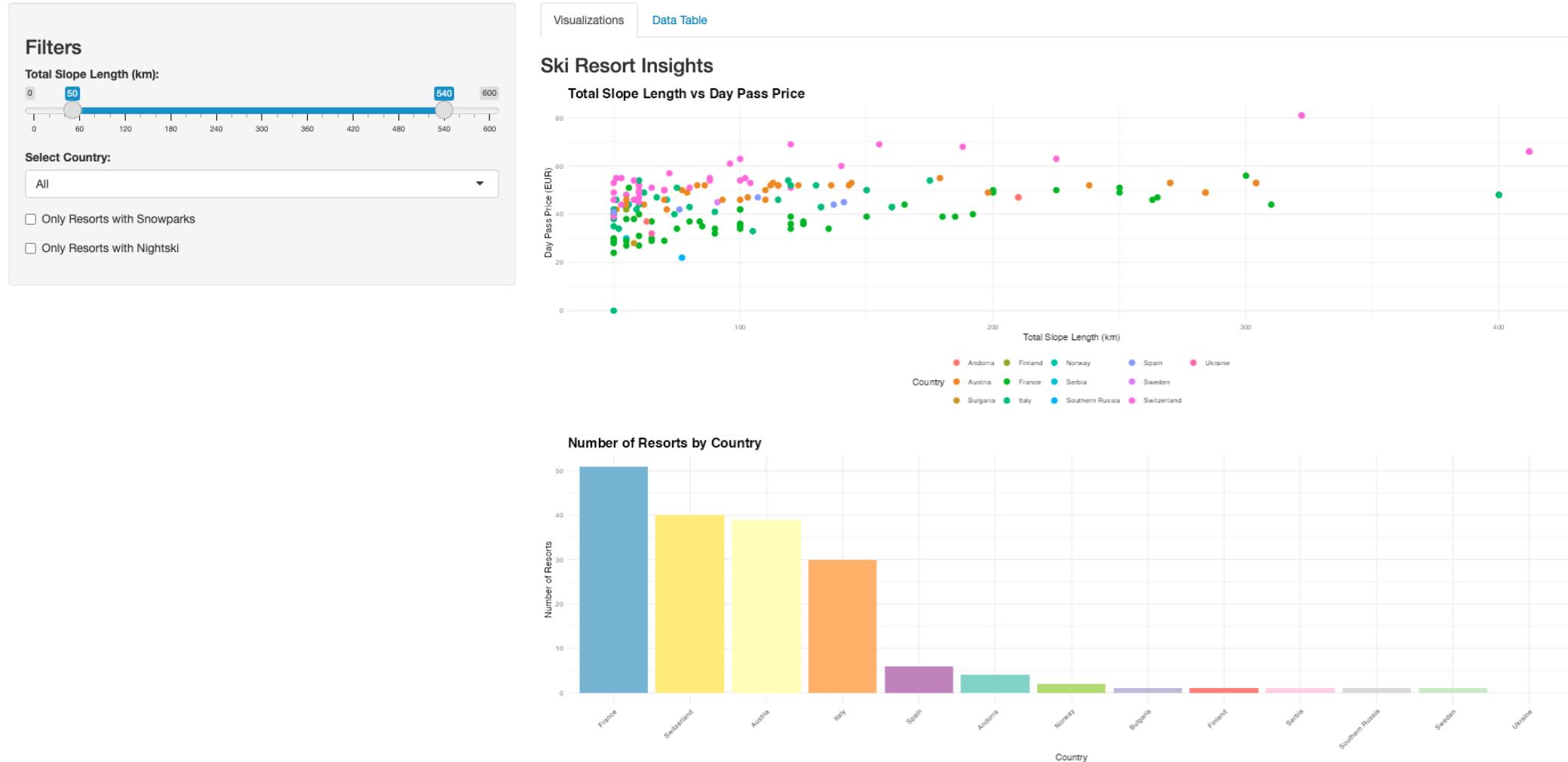
Used to display plots (e.g., bar charts, line graphs) created with ggplot2 or base R. This is the “chart area” where you visualize trends and comparisons.



```
scatter_plot_ui <- fluidPage(  
  titlePanel("Scatter Plot: Slope Length vs. Day Pass Price"),  
  sidebarLayout(  
    sidebarPanel(  
      selectInput("country", "Select Country:",  
                 choices = unique(data$Country),  
                 selected = unique(data$Country)[1])),  
    mainPanel(  
      plotOutput("scatter_plot"))))  
  
scatter_plot_server <- function(input, output) {  
  output$scatter_plot <- renderPlot({  
    filtered <- data %>%  
      filter(Country == input$country)  
  
    ggplot(filtered, aes(x = TotalSlope, y = DayPassPriceAdult)) +  
      geom_point(color = "darkred", size = 3) +  
      labs(title = "Slope Length vs. Day Pass Price", x = "Total Slope  
Length (km)", y = "Day Pass Price (EUR)") +  
      theme_minimal()})}  
  
# Run Scatter Plot App  
  
shinyApp(ui = scatter_plot_ui, server = scatter_plot_server)
```

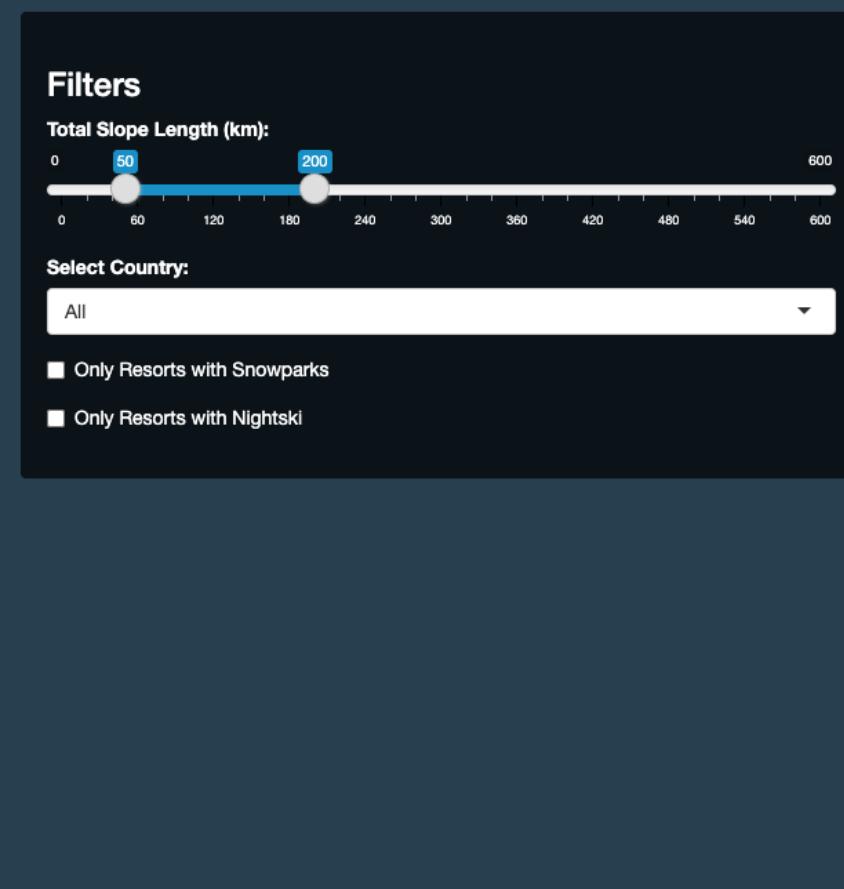
Example: Building a dashboard for European Ski Resorts

European Ski Resorts Dashboard

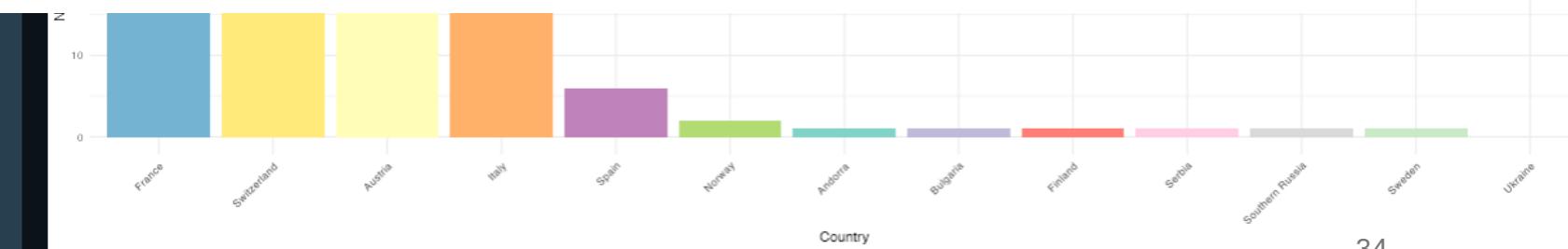


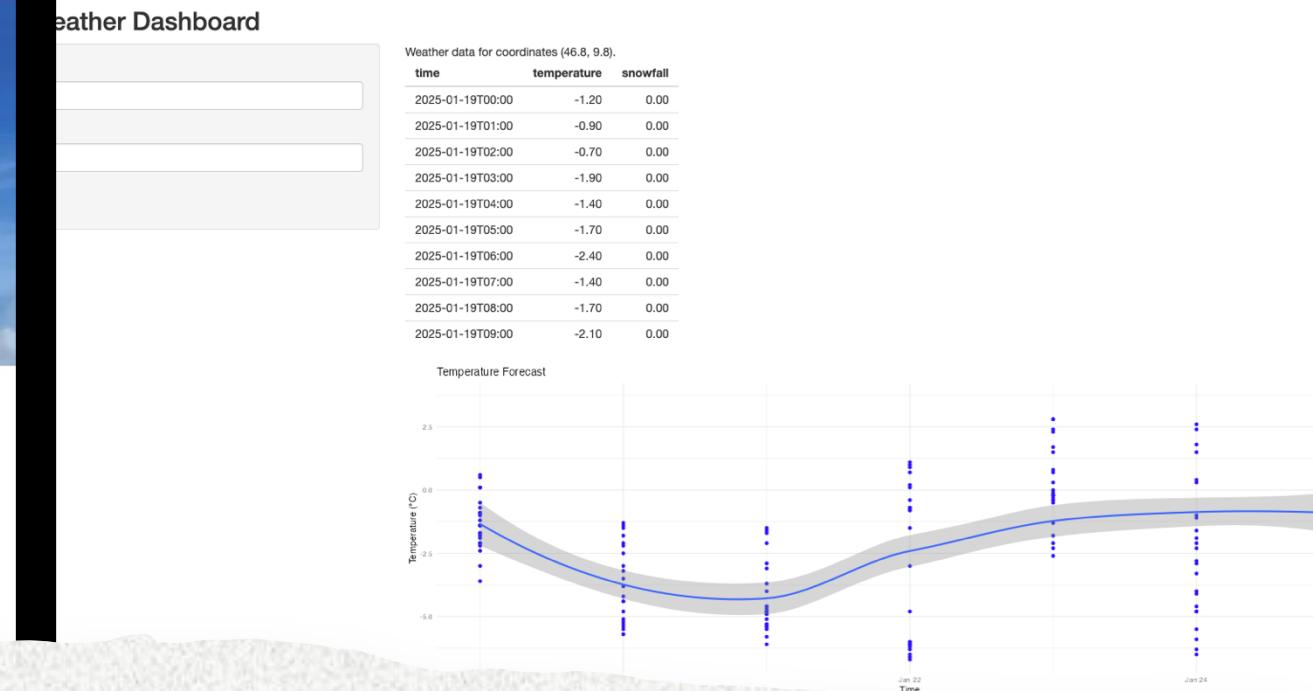
```
## example complete dashboard
```

European Ski Resorts Dashboard



Extension: a little bit of styling

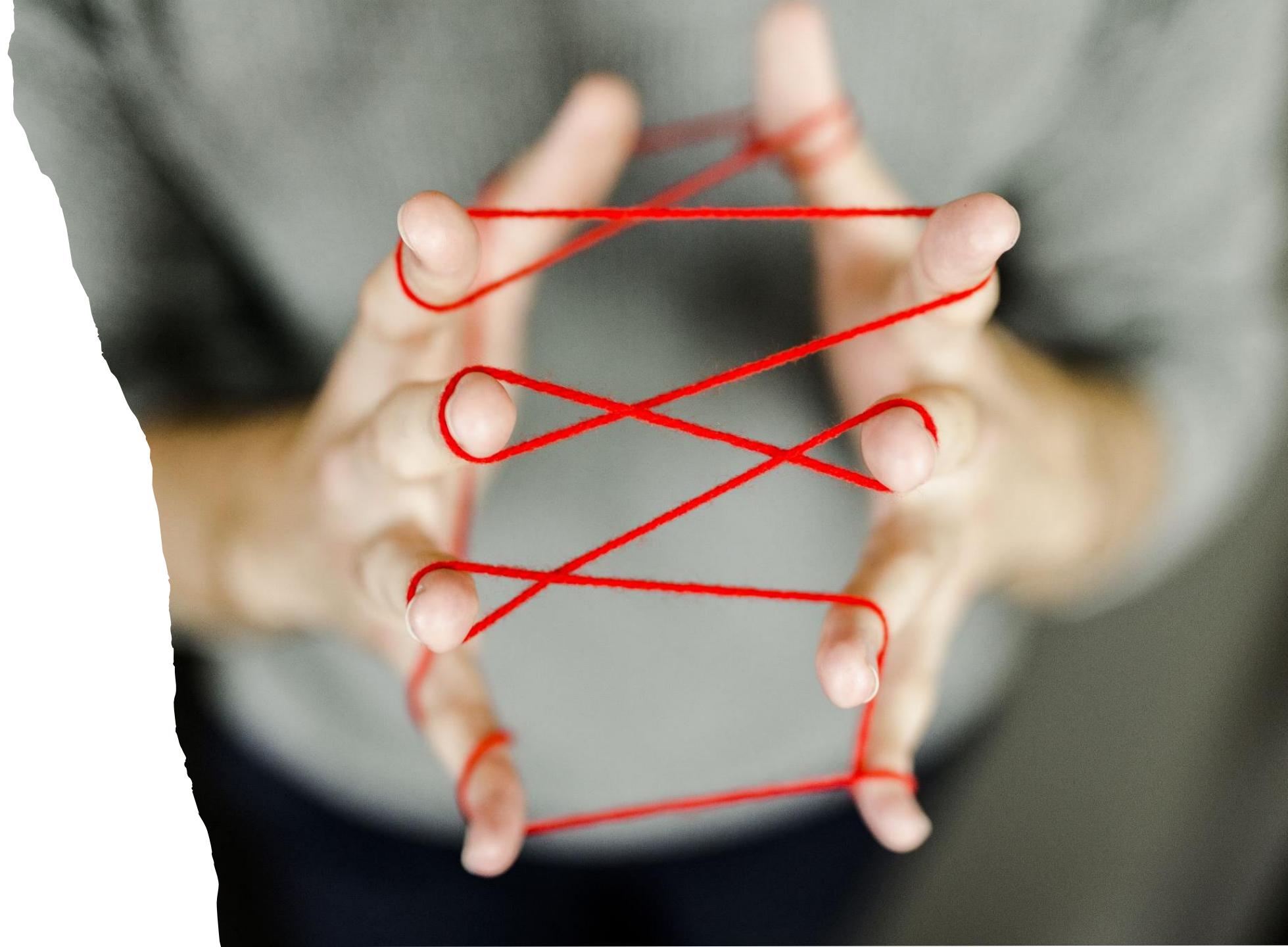


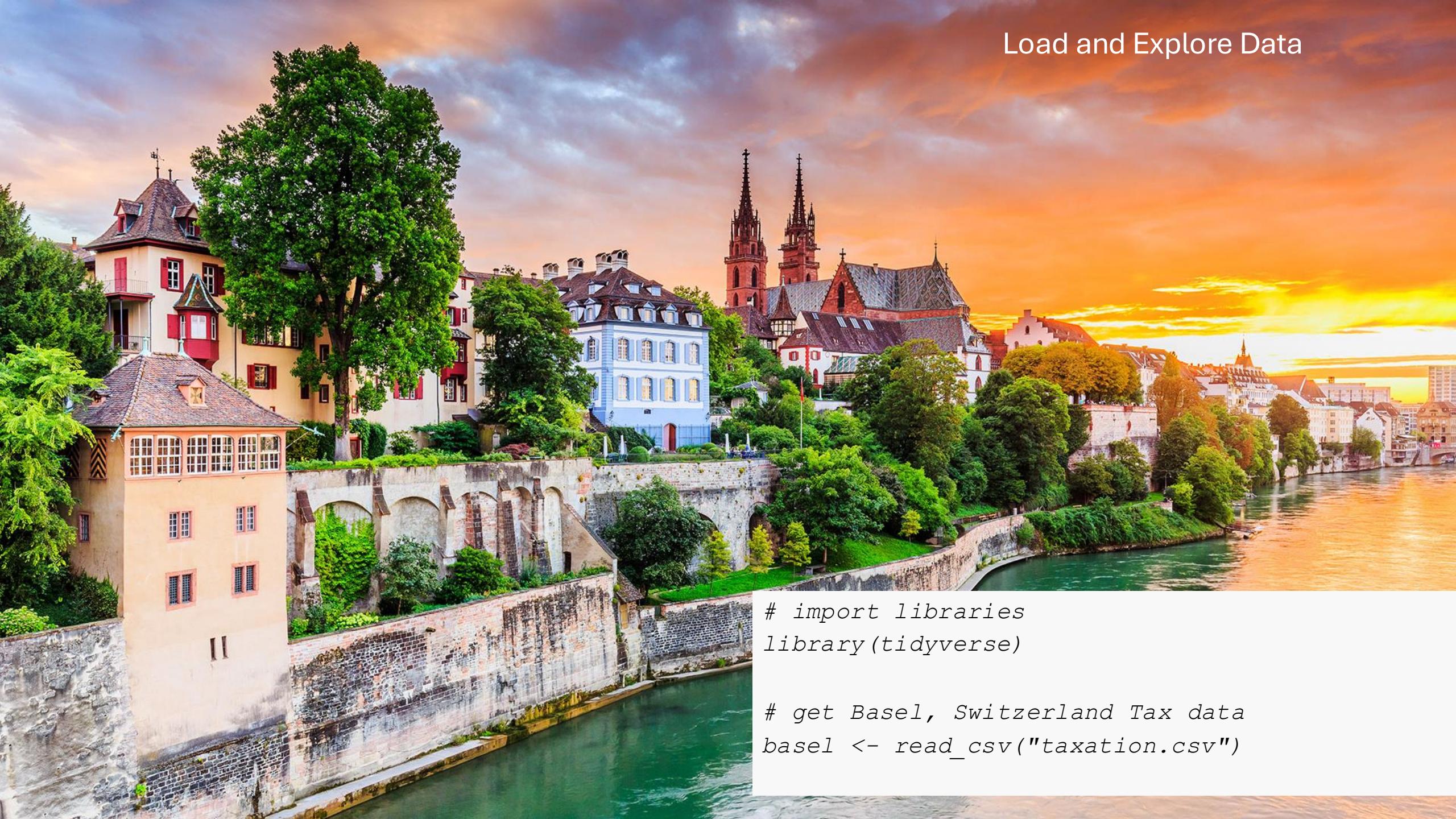


Integration of APIs

API integration

Practice

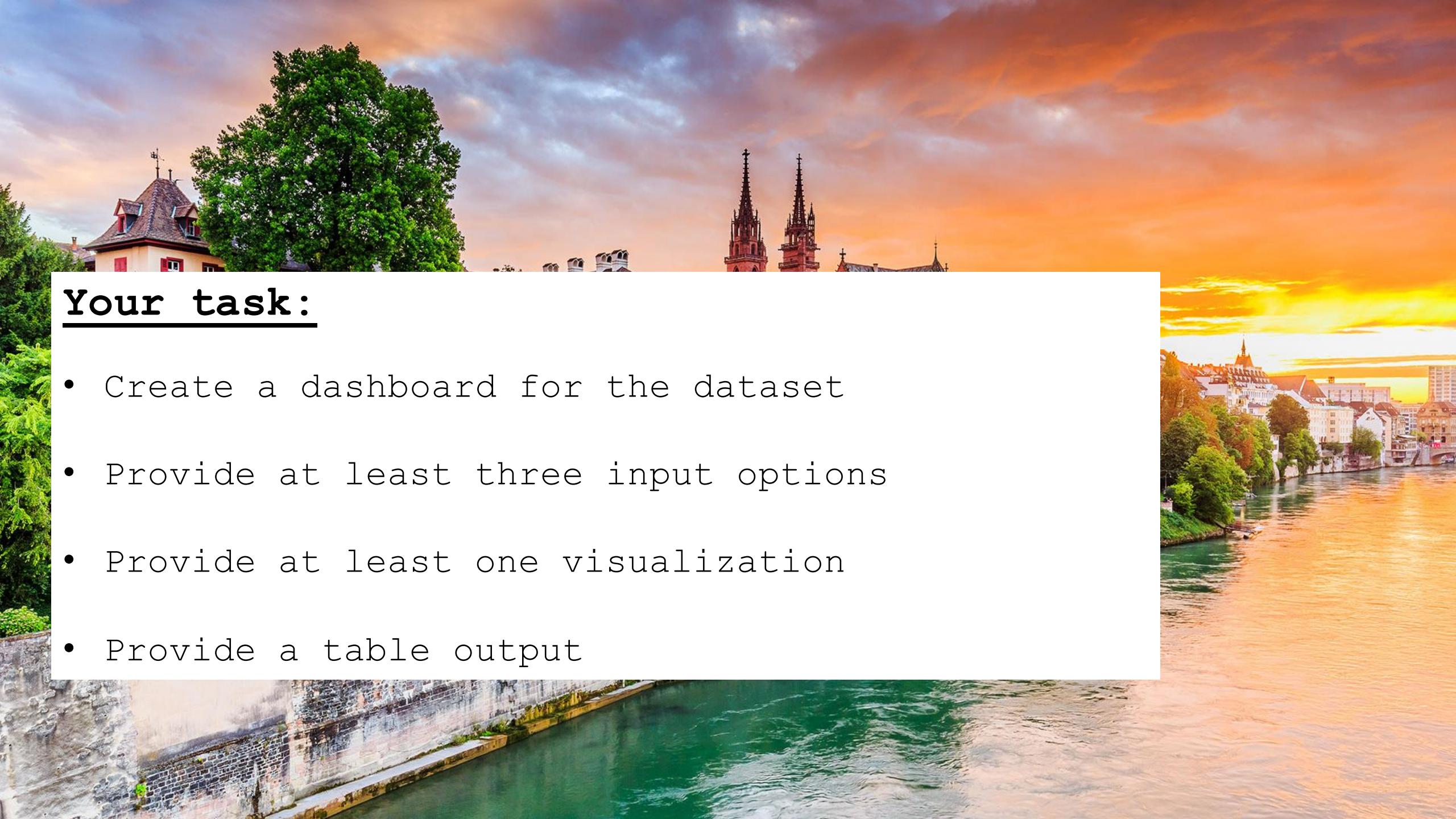


A panoramic photograph of the Basel Minster (Münster) and the Rhine River in Basel, Switzerland. The sky is filled with warm orange and yellow hues of a setting sun. In the foreground, a stone bridge with arches spans the river. On the left bank, there are several traditional Swiss houses with red roofs and white or light-colored facades. In the background, the tall, Gothic spires of the Basel Minster rise against the sky. The river water reflects the warm colors of the sunset.

Load and Explore Data

```
# import libraries
library(tidyverse)

# get Basel, Switzerland Tax data
basel <- read_csv("taxation.csv")
```



Your task:

- Create a dashboard for the dataset
- Provide at least three input options
- Provide at least one visualization
- Provide a table output

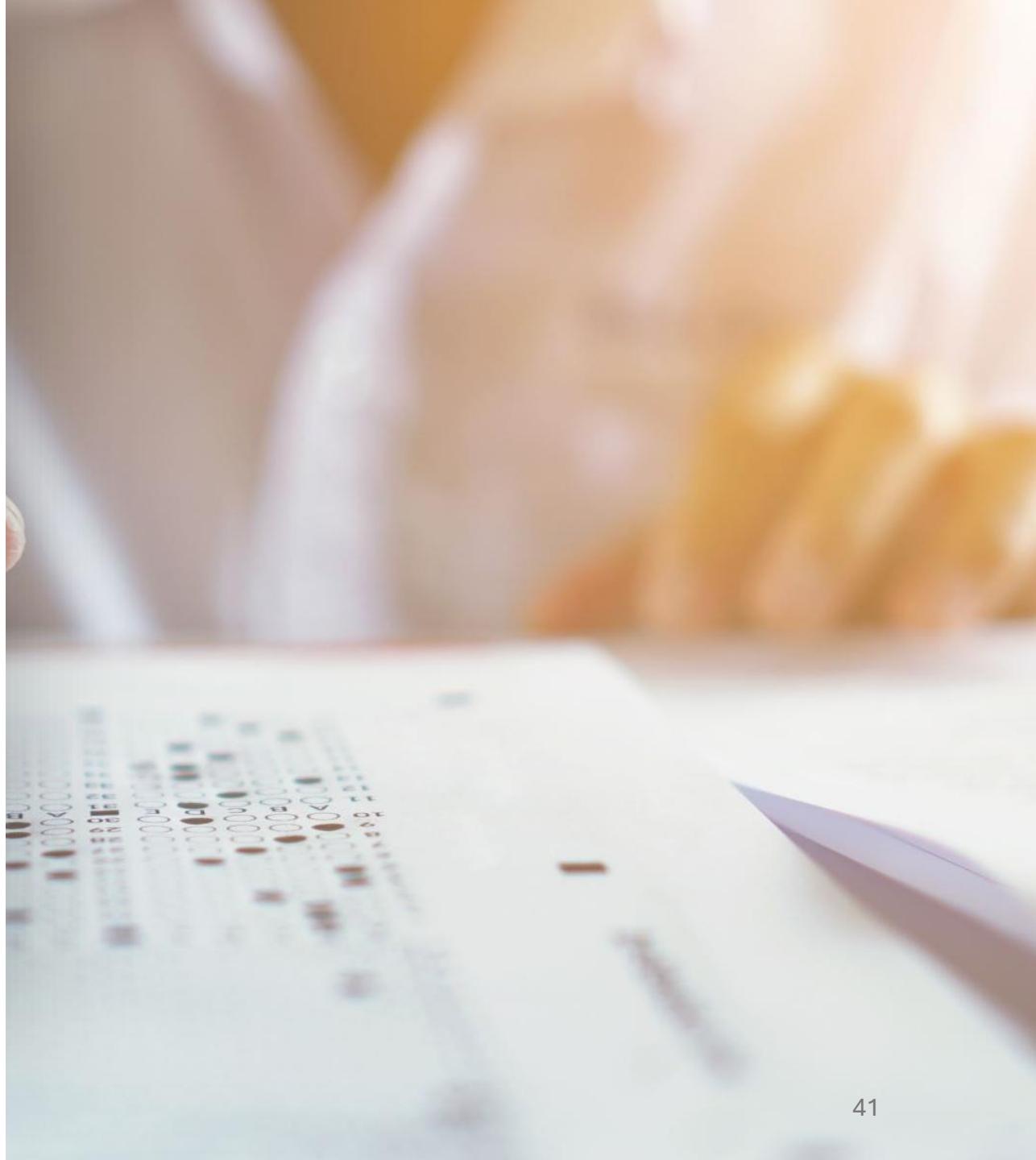
And the winner is...



Final Exam

Final Exam

- First Exam Day DMAV
 - Tuesday, February 23, 2026 at 10:40
 - Building 103, S89 & S58
- Second Exam Day DMAV
 - Thursday, March 23, 2026 at 15:40
 - Building 103, S56



Final Exam Information

- 30 questions
- 30 minutes time limit
- Each question is worth 1 point
- e-Exam
- Multiple choice
- Closed book, closed note, closed internet, etc.
- Individual
- Please choose the best answer for each question.

Final Exam Review

API Data Collection

- Which of the following are examples of data sources?
- What are the two primary ways to get data?
- Third party data typically comes in which of the following formats?
- What is an API call?
- What is the difference between REST, SOAP, and RPC API calls?
- What is the difference between GET and POST API calls?
- List what information you typically need to make an API call?
- What R library can you use to make API calls?
- Some high-level questions about R syntax to make API calls (what certain lines of code do)

Web Scraping Data Collection

- What is web scraping?
- What are the pros and cons of web scraping compared to API calls?
- What are some examples of use cases where web scraping is commonly used?
- What legal and ethical considerations should you consider when web scraping? How does this differ across different contexts (private versus public data; personal use, research use, and commercial use; etc.)
- How does it differ for private use, research use, and commercial use?
- How are targets identified in HTML that scrapers can use?
- What are the steps in web scraping?
- Some high-level questions about R syntax to scrape data.

Data Wrangling

- What is regex / gsub?
- What do common regex / gsub commands do (`\w \d \s . * + ^ $?`)
- How do you get data versus replace data in gsub?
- What do common commands do in tidyverse (select, summarize, filter, group by, arrange desc, mutate, table, pivot_longer, pivot_wider)
- What is the difference between long versus wide data

Data Visualization

- Why use data visualization?
- What makes a good data visualization?
- Why are the characteristics of your audience important to know about?
- What is the difference between top-down biases and bottom-up biases?
- What are the three data visualization (design) principles?
- What are good reasons to use colors?
- When are good times to use different types of charts (heat map, bar plot, violin, radar plot, scatter plot, bubble plot, line chart, time series)?
- Name a few commercial data visualization tools.

ggplot

- If I were to give you a line of code, could you select the correct answer explaining what the code does?
- If I give a plot to create, could you select the right line of code?

Storytelling

- What is data storytelling?
- What tools can be used to create stories?
- What are best practices of story telling?
- What do basic elements of r markdown do?

Introduction to Web-Visualizations

- What is the basic structure of a Shiny app (UI & server components?)
- What are the basic elements of the UI?
- What are different input elements?
- What are different output elements?
- Some questions about R code to build dashboards with shiny.

M.Sc. BUSINESS ANALYTICS & ECONOMETRICS

Capstone Kickoff 2025

04 February, 16:00

The Ship, Vitalisstraße 67, 50827 Köln

[Directions \(GoogleMaps\)](#)



Our Partners 2026



consolut



FOND OF

HDI

**IoT
Venture**



**UNIKLINIK
KÖLN**



VfL WOLFSBURG

Agenda 04.02.2026

Start	End	Topic
16:00	16:10	Arrival & Registration
16:10	16:30	Welcome
16:30	16:45	xDeck Keynote
16:45	17:05	Break
17:05	17:50	Company Pitches (1)
17:50	18:00	Break
18:00	18:45	Company Pitches (2)
18:45	19:30	Aperitif & Networking
19:30	21:00	Speed Dating Dinner
21:00	Open end	Networking & Get-Together