Explicación Código de la Aplicación Shiny para Análisis de Juegos de Steam

1. Carga de Bibliotecas

```
library(shiny)
library(tm)
library(wordcloud)
library(stringr)
library(RColorBrewer)
library(dplyr)
library(ggplot2)
library(DT)
library(plotly)
```

- shiny: Framework principal para crear aplicaciones web interactivas en R.
- tm: Procesamiento de texto para la nube de palabras.
- wordcloud: Generación de nubes de palabras.
- stringr: Manipulación de cadenas de texto.
- RColorBrewer: Paletas de colores para visualizaciones.
- dplyr: Manipulación y transformación de datos.
- ggplot2: Creación de gráficos avanzados.
- DT: Mostrar tablas interactivas.
- plotly: Gráficos interactivos avanzados.

2. Carga y Preprocesamiento de Datos

Carga de datos:

```
data <- read.csv("bestSelling_games.csv", stringsAsFactors = FALSE, fileEncoding =
"UTF-8")
```

Carga el archivo CSV con los datos de juegos, especificando:
- `stringsAsFactors = FALSE` para evitar conversión automática a factores
- `fileEncoding = "UTF-8"` para manejar correctamente caracteres especiales

Preprocesamiento principal:

```
prepared_data <- data %>%
  mutate(
    reviews_like_rate = as.numeric(str_replace(reviews_like_rate, "%", "")),
    all_reviews_number = as.numeric(all_reviews_number),
    estimated_downloads = as.numeric(estimated_downloads),
    rating = as.numeric(rating),
    price = as.numeric(price),
    release_year = as.numeric(substr(release_date, 1, 4)),
```

```
    difficulty = factor(difficulty, levels = 1:5,
                        labels = c("Muy fácil", "Fácil", "Media", "Difícil", "Muy difícil")),
    length = factor(case_when(
      length <= 10 ~ "Corta (≤10h)",
      length <= 20 ~ "Media (11-20h)",
      length <= 40 ~ "Larga (21-40h)",
      TRUE ~ "Muy larga (>40h)"
    ), levels = c("Corta (≤10h)", "Media (11-20h)", "Larga (21-40h)", "Muy larga (>40h)")),
    age_restriction = factor(case_when(
      age_restriction == 0 ~ "Para todos",
      age_restriction <= 10 ~ "10+",
      age_restriction <= 13 ~ "13+",
      age_restriction <= 17 ~ "17+",
      TRUE ~ "Desconocida"
    ), levels = c("Para todos", "10+", "13+", "17+", "Desconocida"))
  ) %>%
  filter(!is.na(reviews_like_rate), !is.na(all_reviews_number))
```

Transformaciones realizadas:
1. Conversión de porcentajes a valores numéricos (eliminando el símbolo %)
2. Conversión de varias columnas a numéricas
3. Extracción del año de lanzamiento
4. Categorización de la dificultad con etiquetas descriptivas
5. Clasificación de la duración en rangos con etiquetas
6. Categorización de las restricciones de edad
7. Filtrado de registros con valores NA en columnas clave

Procesamiento de etiquetas (tags):

```
process_tags <- function(tags) {
  tags %>%
    tolower() %>%
    str_split(",") %>%
    unlist() %>%
    trimws() %>%
    str_replace_all("[^[:alnum:]]", "") %>%
    .[. != ""] %>%
    .[!grepl("^[0-9]+$", .)]
}

all_tags <- process_tags(prepared_data$user_defined_tags)
tags_freq <- table(all_tags) %>% sort(decreasing = TRUE)
```

Esta función:
1. Convierte a minúsculas
2. Divide por comas
3. Elimina espacios en blanco

4. Elimina caracteres no alfanuméricos
5. Filtra cadenas vacías y números puros
6. Finalmente calcula frecuencias de cada tag

Procesamiento de sistemas operativos:

```
process_os <- function(os) {
  os_data <- os %>%
    tolower() %>%
    str_split(",") %>%
    lapply(trimws) %>%
    lapply(function(x) {
      case_when(
        all(c("win", "mac", "linux") %in% x) ~ "Windows + Mac + Linux",
        all(c("win", "mac") %in% x) ~ "Windows + Mac",
        all(c("win", "linux") %in% x) ~ "Windows + Linux",
        "win" %in% x ~ "Windows solo",
        "mac" %in% x ~ "Mac solo",
        "linux" %in% x ~ "Linux solo",
        TRUE ~ "Otros"
      )
    }) %>%
    unlist()

  table(os_data) %>%
    as.data.frame() %>%
    rename(OS = os_data, Count = Freq) %>%
    filter(OS != "Otros") %>%
    arrange(desc(Count))
}
```

Clasifica las combinaciones de sistemas operativos soportados en categorías claras.

3. Interfaz de Usuario (UI)

La interfaz se organiza en:
- Un panel lateral con controles
- Un panel principal con pestañas

Estructura general:

```
ui <- fluidPage(
  tags$head(...),  # Estilos CSS
  titlePanel(...), # Título con imagen de Steam
  sidebarLayout(
    sidebarPanel(...),  # Controles
    mainPanel(
```

```
      tabsetPanel(
        tabPanel("Análisis de especifico de cada Juego", ...),
        tabPanel("Análisis de Compañías", ...),
        tabPanel("Comparacion de cada Juego", ...)
      )
    )
  )
)
```

Componentes clave:

1. Estilos CSS:
   - Personaliza colores, bordes y sombras para mejorar la apariencia
   - Define estilos consistentes para todos los componentes

2. Controles en sidebarPanel:
   - Slider para cantidad máxima de palabras en la nube
   - Radio buttons para seleccionar mejores/peores compañías
   - Slider para mínimo de juegos por compañía

3. Pestañas principales:
   - Análisis específico de cada juego: Contiene visualizaciones sobre juegos individuales
   -Análisis de compañías: Muestra información agregada por desarrollador
   - Comparación de cada juego: Gráficos de distribución por categorías

4. Lógica del Servidor (Server)

El servidor contiene toda la lógica reactiva que responde a las interacciones del usuario.

Componentes principales:

1. Nube de palabras:

```
  output$wordcloud <- renderPlot({
    filtered_tags <- tags_freq[tags_freq >= 3]
    wordcloud(...)
  })
```

   - Filtra tags con frecuencia mínima
   - Genera la nube con colores y escala apropiados

2. Top 15 juegos más descargados:

```
  output$top15_downloads_plot <- renderPlot({
    top15 <- prepared_data %>% ... %>% head(15)
    ggplot(...) + geom_col() + coord_flip()
  })
```

- Ordena por descargas
- Muestra gráfico de barras horizontales con valores

3.Gráfico de dispersión precio-descargas:

```
output$price_downloads_scatter <- renderPlotly({
  ggplotly(
    ggplot(...) + geom_point(),
    tooltip = "text"
  )
})
```

- Interactivo con plotly
- Muestra tooltips con detalles al pasar el mouse
- Permite selección de puntos

4. Tablas de juegos:

```
output$games_table <- renderDT({...})
output$selected_games <- renderDT({...})
```

- Tabla principal con opción de selección múltiple
- Tabla secundaria que muestra detalles de juegos seleccionados

5. Análisis de compañías:

```
companies_data <- reactive({
  prepared_data %>%
    group_by(developer) %>%
    summarise(...) %>%
    filter(n_games >= input$min_games) %>%
    {if (input$rating_type == "best") arrange(., desc(avg_rating)) else arrange(., avg_rating)}
%>%
    head(15)
})
```

- Datos reactivos que cambian según selección del usuario
- Calcula rating promedio y total de descargas por compañía

6. Gráficos de distribución (pie charts):

```
output$difficulty_pie <- renderPlot({
  difficulty_data <- prepared_data %>% group_by(difficulty) %>% summarise(...)
  ggplot(...) + geom_bar() + coord_polar()
})
```

- Similar para duración, edad y sistemas operativos

- Usa coordenadas polares para crear gráficos de pie
- Incluye porcentajes como etiquetas

5. Ejecución de la Aplicación

shinyApp(ui = ui, server = server)

Une los componentes de interfaz y servidor para crear la aplicación final.

Flujo de Datos Completo

1. Carga inicial:
   - Los datos crudos se leen desde CSV
   - Se transforman a un formato adecuado para análisis

2. Procesamiento adicional:
   - Tags y sistemas operativos se procesan por separado
   - Se calculan frecuencias y categorizaciones

3. Interacción del usuario:
   - Los controles en el sidebar afectan a visualizaciones reactivas
   - Las selecciones en tablas actualizan otras visualizaciones

4. Renderizado:
   - Cada output reacciona a cambios en sus dependencias
   - Los gráficos se actualizan automáticamente

Características Avanzadas

1. Interactividad:
   - Tooltips con detalles en gráficos plotly
   - Selección múltiple en tablas
   - Filtrado dinámico de datos

2. Visualización profesional:
   - Uso de paletas de colores coherentes
   - Gráficos bien etiquetados y legibles
   - Diseño responsive que se adapta a diferentes tamaños de pantalla

3. Manejo de datos:
   - Limpieza y transformación robusta de datos
   - Agregaciones complejas con dplyr
   - Filtrado reactivo basado en entradas del usuario