Primer Archivo

Andrea B

2023-09-26

Implementación de un Juego de Cartas

Durante este tutorial se va a desarrollar la implementación de un Juego de Cartas, el primer código que se mostrará a continuación, permite crear un archivo CSV con todas las cartas de una baraja de póquer estándar.

```
# Crear una baraja de póquer estándar
suits <- c("Corazones", "Diamantes", "Tréboles", "Picas")
values <- c("2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A")

# Crear un data frame que contenga todas las cartas
all_cards <- expand.grid(Suit = suits, Value = values)

# Guardar el data frame en un archivo CSV
write.csv(all_cards, "baraja_poker.csv", row.names = FALSE)
```

Continuando con la implementación de códigos en R, a continuación, se mostrará el código correspondiente a crear una función que reciba el nombre del archivo CSV y devuelva un marco de datos con la información de la baraja de póquer estándar.

```
# Definir una función para cargar la baraja de póquer y asignar valores
load_poker_deck <- function(file_name) {
  # Cargar el archivo CSV
  poker_deck <- read.csv(file_name, header = TRUE, stringsAsFactors = FALSE)

  # Definir una función personalizada para asignar valores
  assign_card_value <- function(card) {
    card_value <- as.character(card$Value)
    if (grepl("^[2-9]$", card_value)) {
      return(as.numeric(card_value))
    } else if (card_value %in% c("10", "J", "Q", "K")) {
      return(10)
    } else if (card_value == "A") {
      return(11)  # Puedes ajustar el valor de A según las reglas del juego
    }
```

```
  }

  # Agregar una columna "CardValue" con los valores asignados
  poker_deck$CardValue <- sapply(poker_deck, assign_card_value)

  return(poker_deck)
}
```

Se continúa dando paso a crear un código en R, que permita mezclar las cartas de la baraja.

```
# Crear una baraja de 52 cartas
baraja <- 1:52

# Mezclar las cartas
baraja_mezclada <- sample(baraja)

# Mostrar la baraja mezclada
cat("Baraja mezclada:", baraja_mezclada, "\n")
```

```
## Baraja mezclada: 3 49 40 44 12 27 6 45 4 26 47 9 30 37 23 38 16 28 29 17 13 34 42 18 4
1 21 36 2 33 31 52 5 7 50 43 39 20 32 51 35 48 19 8 46 24 1 11 22 25 15 10 14
```

Vamos a crear un código en R, que reparta un número de cartas de la baraja, pero que una vez una de las cartas ha sido repartida no puede ser seleccionada de nuevo.

```
# Crear una función para repartir cartas
repartir_cartas <- function(numero_cartas) {
  # Crear una baraja de 52 cartas
  baraja <- 1:52

  # Inicializar un vector para almacenar las cartas repartidas
  cartas_repartidas <- c()

  # Comprobar si hay suficientes cartas para repartir
  if (numero_cartas > length(baraja)) {
    cat("No hay suficientes cartas en la baraja.\n")
    return(NULL)
  }

  # Repartir cartas
  for (i in 1:numero_cartas) {
```

```
    # Seleccionar una carta al azar
    carta_seleccionada <- sample(baraja, 1)

    # Agregar la carta seleccionada a las cartas repartidas
    cartas_repartidas <- c(cartas_repartidas, carta_seleccionada)

    # Eliminar la carta seleccionada de la baraja
    baraja <- baraja[-which(baraja == carta_seleccionada)]
  }

  # Mostrar las cartas repartidas
  cat("Cartas repartidas:", cartas_repartidas, "\n")
}

# Llamar a la función para repartir 5 cartas (puedes cambiar el número)
repartir_cartas(5)
## Cartas repartidas: 31 5 19 32 42
```

Black Jack

Vamos a crear un script de R que permita jugar al black Jack (21), este código tendrá la opción de preguntar al usuario si quiere tomar otra carta e indicará si pierde, gana o decide parar.

```
# Función para calcular el valor de una mano de blackjack
calcular_puntuacion <- function(mano) {
  puntuacion <- sum(mano)
  if (puntuacion > 21 && 11 %in% mano) {
    puntuacion <- puntuacion - 10
  }
  return(puntuacion)
}

# Función para jugar al blackjack
jugar_blackjack <- function() {
  # Baraja de cartas
  cartas <- c(2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, 11)

  # Repartir dos cartas al jugador y al crupier
  jugador <- sample(cartas, 2)
```

```
  crupier <- sample(cartas, 2)

  cat("¡Bienvenido al Blackjack!\n\n")

  # Comenzar el juego
  while (TRUE) {
    cat("Tus cartas:", jugador, "Puntuación:", calcular_puntuacion(jugador), "\n")
    cat("Carta del crupier:", crupier[1], "\n")

    # Verificar si el jugador ha ganado o perdido
    if (calcular_puntuacion(jugador) > 21) {
      cat("Has perdido. ¡Tienes más de 21 puntos!\n")
      break
    }

    # Preguntar al jugador si desea tomar otra carta o parar
    respuesta <- readline("¿Quieres tomar otra carta? (s/n): ")

    if (respuesta == "s") {
      # Tomar una carta adicional y agregarla a la mano del jugador
      nueva_carta <- sample(cartas, 1)
      jugador <- c(jugador, nueva_carta)
    } else {
      # El jugador decide parar, ahora el crupier juega
      while (calcular_puntuacion(crupier) < 17) {
        nueva_carta_crupier <- sample(cartas, 1)
        crupier <- c(crupier, nueva_carta_crupier)
      }

      cat("Cartas del crupier:", crupier, "Puntuación del crupier:", calcular_puntuacion(
crupier), "\n")

      # Determinar el resultado
      if (calcular_puntuacion(jugador) > 21) {
        cat("¡Gana el crupier! Tu puntuación es mayor que 21.\n")
      } else if (calcular_puntuacion(crupier) > 21 || calcular_puntuacion(jugador) > calc
ular_puntuacion(crupier)) {
        cat("¡Felicidades! ¡Has ganado!\n")
      } else if (calcular_puntuacion(jugador) == calcular_puntuacion(crupier)) {
```

```
        cat("Es un empate.\n")
      } else {
        cat("¡Gana el crupier! Su puntuación es mayor que la tuya.\n")
      }
      break
    }
  }
}

# Iniciar el juego de blackjack
jugar_blackjack()
```

```
## ¡Bienvenido al Blackjack!
##
## Tus cartas: 11 10 Puntuación: 21
## Carta del crupier: 3
## ¿Quieres tomar otra carta? (s/n):
## Cartas del crupier: 3 8 11 8 Puntuación del crupier: 20
## ¡Felicidades! ¡Has ganado!
```

Integración Básica de Phyton y R.

Avanzando con los scripts de R, comenzaremos a implementar Python de manera simultánea, a continuación se utiliza el script correcto para instalar las librerías de Python como son pandas, numpy, scipy, matplotlib y seaborn.

```
#Instalar paquetes en R
library(reticulate)

#Instalar libreria pandas
py_install("pandas")
```

```
## Using virtual environment "~/.virtualenvs/r-reticulate" ...
## + "C:/Users/Andrea/Documents/.virtualenvs/r-reticulate/Scripts/python.exe" -m pip inst
all --upgrade --no-user pandas
```

```
#Instalar libreria Pandas
import pandas as pd

#Instalar libreria matplotlib
import matplotlib as mpl

#Instalar libreria seaborn
import seaborn as sns
```

```
#Instalar libreria numpy
import numpy as np

#Instalar libreria scipy
from scipy import stats
```

Código phyton para encontrar números primos.

Se implementará un script Python que encuentre los números primos entre dos números enteros.

```
def es_primo(numero):
    if numero <= 1:
        return False
    if numero <= 3:
        return True
    if numero % 2 == 0 or numero % 3 == 0:
        return False
    i = 5
    while i * i <= numero:
        if numero % i == 0 or numero % (i + 2) == 0:
            return False
        i += 6
    return True

def encontrar_primos_entre_limites(inicio, fin):
    primos = []
    for num in range(max(2, inicio), fin + 1):
        if es_primo(num):
            primos.append(num)
    return primos

# Especifica los límites del rango
inicio = 10
fin = 50
```

```
# Encuentra los números primos en el rango
numeros_primos = encontrar_primos_entre_limites(inicio, fin)

# Imprime los números primos encontrados
print(f"Números primos entre {inicio} y {fin}: {numeros_primos}")
```

```
## Números primos entre 10 y 50: [11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Cómo simular una baraja de póquer en R.

Es una tarea fundamental cuando se trabaja en proyectos de análisis de datos o desarrollo de juegos de cartas. Una baraja de póquer consta de 52 cartas, divididas en cuatro palos (corazones, diamantes, tréboles y picas) y tres valores (del 2 al 10, y las cartas de la J a la K, más el As). Para lograrlo en R, podemos representar la baraja como un conjunto de números o caracteres que identifican de manera única cada carta. Luego, podemos utilizar la función **`sample`**o generar números aleatorios para simular la mezcla de la baraja y repartir las cartas a los jugadores en juegos de póquer simulados o para cualquier otro propósito. Este enfoque nos permite llevar a cabo análisis estadísticos, simulaciones y juegos de póquer virtuales utilizando R.

```
# Crear una baraja de póker
baraja <- c("2♠", "3♠", "4♠", "5♠", "6♠", "7♠", "8♠", "9♠", "10♠", "J♠", "Q♠", "K♠", "A♠"
,
            "2♥", "3♥", "4♥", "5♥", "6♥", "7♥", "8♥", "9♥", "10♥", "J♥", "Q♥", "K♥", "A♥"
,
            "2♦", "3♦", "4♦", "5♦", "6♦", "7♦", "8♦", "9♦", "10♦", "J♦", "Q♦", "K♦", "A♦"
,
            "2♣", "3♣", "4♣", "5♣", "6♣", "7♣", "8♣", "9♣", "10♣", "J♣", "Q♣", "K♣", "A♣"
)

# Mezclar la baraja
baraja_mezclada <- sample(baraja)

# Mostrar la baraja mezclada
cat("Baraja de póker mezclada:\n", baraja_mezclada, "\n")
```

```
## Baraja de póker mezclada:
##  K♠ 9♥ 2♦ 10♠ J♠ 8♣ 3♣ 9♠ 3♠ 5♣ 8♠ 6♠ 8♥ 7♥ K♣ K♥ 4♦ 5♥ 7♣ 10♣ 4♥ 2♣ J♥ 3♥ Q♦ K♦ A♥ 10
♥ 5♠ 7♠ 9♣ 6♥ Q♣ 6♣ 9♦ 10♦ J♣ A♦ 2♥ A♣ A♠ 2♠ 5♦ Q♥ 4♠ 3♦ 6♦ 7♦ 8♦ 4♣ J♦ Q♠
```

Código en R para crear un marco de datos.

Continuando con la creación de códigos en R, se mostrará el código para crear o cargar un marco de datos frame con la información de una baraja.

```r
# Crear un vector con los valores de las cartas
valores <- c("2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A")

# Crear un vector con los palos de las cartas
palos <- c("Corazones", "Diamantes", "Tréboles", "Picas")

# Crear todas las combinaciones de cartas
cartas <- expand.grid(Valor = valores, Palo = palos)

# Mostrar el marco de datos con la información de la baraja
print(cartas)
```

```
##    Valor      Palo
## 1      2 Corazones
## 2      3 Corazones
## 3      4 Corazones
## 4      5 Corazones
## 5      6 Corazones
## 6      7 Corazones
## 7      8 Corazones
## 8      9 Corazones
## 9     10 Corazones
## 10     J Corazones
## 11     Q Corazones
## 12     K Corazones
## 13     A Corazones
## 14     2 Diamantes
## 15     3 Diamantes
## 16     4 Diamantes
## 17     5 Diamantes
## 18     6 Diamantes
## 19     7 Diamantes
## 20     8 Diamantes
## 21     9 Diamantes
## 22    10 Diamantes
```

Código Phyton para mezclar una Baraja de Póker

Posteriormente pasando de códigos en R vamos a comenzar a aplicar códigos en phyton, a continuación se realizará una programación para mezclar una baraja de póker.

```
import random

# Crear una baraja de poker
def crear_baraja():
    palos = ['Corazones', 'Diamantes', 'Tréboles', 'Picas']
    valores = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
    baraja = [{'valor': valor, 'palo': palo} for valor in valores for palo in palos]
    return baraja

# Función para barajar la baraja
def barajar_baraja(baraja):
    random.shuffle(baraja)

# Función para mostrar la baraja
def mostrar_baraja(baraja):
    for carta in baraja:
        print(f"{carta['valor']} de {carta['palo']}")

# Crear y barajar la baraja
baraja = crear_baraja()
barajar_baraja(baraja)

# Mostrar la baraja barajada
mostrar_baraja(baraja)
```

```
## 5 de Picas
## 7 de Corazones
## K de Diamantes
## 6 de Diamantes
## 2 de Diamantes
## 5 de Diamantes
## 7 de Tréboles
## 9 de Picas
## K de Picas
## 3 de Picas
```

```
## 6 de Tréboles
## 8 de Picas
## A de Diamantes
## 9 de Diamantes
## J de Picas
## A de Tréboles
## 9 de Tréboles
## 8 de Tréboles
## 7 de Diamantes
## A de Picas
## 4 de Picas
## Q de Diamantes
## 10 de Tréboles
## 8 de Corazones
## 3 de Corazones
## 4 de Diamantes
## A de Corazones
## 5 de Tréboles
## 3 de Diamantes
## 10 de Diamantes
## 3 de Tréboles
## 6 de Picas
## K de Tréboles
## 9 de Corazones
## 6 de Corazones
## Q de Corazones
## 8 de Diamantes
## 2 de Tréboles
## Q de Picas
## J de Diamantes
## J de Corazones
## Q de Tréboles
## 10 de Corazones
## J de Tréboles
## K de Corazones
## 10 de Picas
## 2 de Picas
```