Índice general

Capítulo 1

Apéndice: Codigo R utilizado en las simulaciones

1.1. Codigo para calcular la suma final del crupier

```
library(knitr)
library(kableExtra)
library(tidyverse)
Cartas <- c("2","3","4","5","6","7","8","9","Figura","As")
Cantidad_de_cada_carta <- c(4,4,4,4,4,4,4,4,16,4)
Valor_cartas <- c(2,3,4,5,6,7,8,9,10,11)
Probabilidades_sacar_carta = Cantidad_de_cada_carta/sum(Cantidad_de_cada_carta)
Valor_cartas_mano_dura =c(2,3,4,5,6,7,8,9,10,1)
Valor_cartas_mano_blanda = Valor_cartas

encuentra_as_en_mano <- function(mano){
  if ("As" %in% mano) {
    return(TRUE)
  } else {
    return(FALSE)
  }
}

cuenta_ases_mano <- function(mano){
  numero_ases=0
  for (carta in mano) {
    if (carta=="As") {
      numero_ases=numero_ases+1
    }
  }
  return(numero_ases)
}
```

```
Devuelve_salida <- function(suma,mano){
  if ((all(sort(mano) == sort(c("Figura", "As"))))) {
    return("BlackJack")
  } else{
    if (suma >21) {
      return("Se pasa")
    } else{
      return(as.character(suma))
    }
  }
}


Calcular_suma_mano_crupier <- function(mano){
  suma=0
  if ((all(sort(mano) == sort(c("Figura", "As"))))) {
    suma = 21
  } else {
    if (encuentra_as_en_mano(mano)) {
      mano_sin_ases <- mano[mano != "As"]
      mano_solo_ases <- mano[mano == "As"]
      for (carta in mano_sin_ases) {
        suma= suma + Valor_cartas[which(Cartas == carta)]
      }
      while (cuenta_ases_mano(mano_solo_ases)>0) {
        if ((suma + 11>=17) && (suma + 11 <= 21)) {
          suma = suma+11
        } else{
          suma=suma+1}
        mano_solo_ases =mano_solo_ases[-1]
      }
    }
    else {
      for (carta in mano) {
        suma= suma + Valor_cartas[which(Cartas == carta)]
      }
    }
  }
  return(suma)
}

Mano_dada_crupier <- function(b){ #b es la carta que todos ven que tiene el crupier
  suma=0
  mano=c(b)
  while (suma<17) {
    carta_1 <- sample(Cartas,size = 1,prob = Probabilidades_sacar_carta)
    mano = c(mano,carta_1)
```

```
    suma=Calcular_suma_mano_crupier(mano)
  }
  Devuelve_salida(suma,mano)
}

resultados_tabla <- data.frame()

# Calcular probabilidades para cada carta visible
for (carta in Cartas) {
  salida <- replicate(10000, Mano_dada_crupier(carta))
  tabla <- prop.table(table(factor(salida,
                                   levels = c("17","18","19","20","21","BlackJack","Se
  resultados_tabla <- rbind(resultados_tabla, as.numeric(tabla))
  colnames(resultados_tabla)=c("17","18","19","20","21","BlackJack","Se pasa")
}
rownames(resultados_tabla)=c("2","3","4","5","6","7","8","9","Figura","As")


Calculo_G_0 <- function(x, b, tabla, es_blackjack = FALSE) {
  probabilidades_final_crupier <- tabla[b, ]

  # Extraemos probabilidades
  pr_T_17_20 <- as.numeric(probabilidades_final_crupier[c("17", "18", "19", "20")])
  pr_T_21 <- probabilidades_final_crupier$"21"
  pr_T_bj <- probabilidades_final_crupier$"BlackJack"
  pr_T_pasa <- probabilidades_final_crupier$`Se pasa`

  if (x > 21) {
    return(-1)
  }

  if (x == 21 && es_blackjack) {
    return(1.5 * (1 - pr_T_bj))  # 0 * pr_T_bj es innecesario
  }

  pr_T_menor <- sum(pr_T_17_20[which(17:20 < x)])
  pr_T_igual <- sum(pr_T_17_20[which(17:20 == x)]) + ifelse(x == 21, pr_T_21, 0)
  pr_T_mayor <- sum(pr_T_17_20[which(17:20 > x)]) + ifelse(x < 21, pr_T_21, 0) + pr_T

  ganancia <- (+1) * (pr_T_menor + pr_T_pasa) + (0) * pr_T_igual + (-1) * pr_T_mayor

  return(ganancia)
}
```

1.2. Codigo para la ganancia y estrategia en caso de mano dura

```
estrategia_G_optima <- matrix("", nrow = 28, ncol = 10)
rownames(estrategia_G_optima) <- as.character(4:31)
colnames(estrategia_G_optima) <- Cartas  # del 2 al As
ganancia_G_optima <- matrix(NA, nrow = 28, ncol = 10)
rownames(ganancia_G_optima) <- as.character(4:31)
colnames(ganancia_G_optima) <- Cartas  # del 2 al As
for (x in 22:31) {
  ganancia_G_optima[as.character(x),] <- -1
  estrategia_G_optima[as.character(x),] <- "Parar"
}

for (x in 21:4) {
  for (b in Cartas) {

    # Verifica si es blackjack (21 con 2 cartas), solo posible si x == 21
    es_blackjack <- (x == 21)  # Aquí podrías añadir verificación con número de cart

    G0 <- Calculo_G_0(x, b, resultados_tabla,es_blackjack = es_blackjack)

    # Esperanza de continuar
    G_continuar <- 0
    for (j in 1:10) {
      nueva_x <- x + Valor_cartas_mano_dura[j]
      if (nueva_x > 21) {
        G_continuar <- G_continuar - Probabilidades_sacar_carta[j]
      } else {
        G_continuar <- G_continuar + Probabilidades_sacar_carta[j] *ganancia_G_optima
      }
    }

    if (G0 >= G_continuar) {
      estrategia_G_optima[as.character(x), b] <- "Parar"
      ganancia_G_optima[as.character(x),b] <- G0
    } else {
      estrategia_G_optima[as.character(x), b] <- "Continuar"
      ganancia_G_optima[as.character(x),b] <- G_continuar
    }
  }
}

kable(estrategia_G_optima,
      caption = "Tabla de procedimientos si el jugador posee una mano dura",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
```

```
                full_width = F, font_size = 12)

knitr::kable(round(ganancia_G_optima,3),
      caption = "Tabla de ganancias si el jugador posee una mano dura",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
                full_width = F, font_size = 12)
```

1.3. Codigo para la ganancia y estrategia en caso de mano blanda

```
estrategia_G_optima_mano_blanda <- matrix("", nrow = 28, ncol = 10)
rownames(estrategia_G_optima_mano_blanda) <- as.character(4:31)
colnames(estrategia_G_optima_mano_blanda) <- Cartas  # del 2 al As
ganancia_G_optima_mano_blanda <- matrix(NA, nrow = 28, ncol = 10)
rownames(ganancia_G_optima_mano_blanda) <- as.character(4:31)
colnames(ganancia_G_optima_mano_blanda) <- Cartas  # del 2 al As
for (x in 22:31) {
  ganancia_G_optima_mano_blanda[as.character(x),] <- -1
  estrategia_G_optima_mano_blanda[as.character(x),] <- "Parar"
}

for (x in 21:4) {
  for (b in Cartas) {

    # Verifica si es blackjack (21 con 2 cartas), solo posible si x == 21
    es_blackjack <- (x == 21)  # Aquí podrías añadir verificación con número de cart

    G0 <- Calculo_G_0(x, b, resultados_tabla,es_blackjack = es_blackjack)

    # Esperanza de continuar
    G_continuar <- 0
    for (j in 1:10) {
      nueva_x <- x + Valor_cartas_mano_blanda[j]
      if (nueva_x > 21) {
        nueva_x_dura <- nueva_x-10
        G_continuar <- G_continuar - Probabilidades_sacar_carta[j]*ganancia_G_optima[a
      } else {
        G_continuar <- G_continuar + Probabilidades_sacar_carta[j] *ganancia_G_optima
      }
    }

    if (G0 >= G_continuar) {
      estrategia_G_optima_mano_blanda[as.character(x), b] <- "Parar"
      ganancia_G_optima_mano_blanda[as.character(x),b] <- G0
```

```
    } else {
      estrategia_G_optima_mano_blanda[as.character(x), b] <- "Continuar"
      ganancia_G_optima_mano_blanda[as.character(x),b] <- G_continuar
    }
  }
}

kable(estrategia_G_optima_mano_blanda,
      caption = "Tabla de procedimientos si el jugador posee una mano blanda",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
                full_width = F, font_size = 12)

kable(round(ganancia_G_optima_mano_blanda,5),
      caption = "Tabla de ganancias si el jugador posee una mano blanda",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
                full_width = F, font_size = 12)
```

1.4. Codigo para la ganancia y estrategia en caso de doblarse o no

```
estrategia_doblarse_NoDoblarse <- matrix("", nrow = 3, ncol = 10)
rownames(estrategia_doblarse_NoDoblarse) <- as.character(9:11)
colnames(estrategia_doblarse_NoDoblarse) <- Cartas  # del 2 al As
ganancia_doblarse_NoDoblarse <- matrix(NA, nrow = 3, ncol = 10)
rownames(ganancia_doblarse_NoDoblarse) <- as.character(9:11)
colnames(ganancia_doblarse_NoDoblarse) <- Cartas  # del 2 al As

for (x in 9:11) {
  for (b in Cartas) {
    G_estrella=ganancia_G_optima[as.character(x),b]
    # Verifica si es blackjack (21 con 2 cartas), solo posible si x == 21
    es_blackjack <- (x == 21)  # Aquí podrías añadir verificación con número de cart
    # Esperanza de continuar
    G_continuar <- 0
    for (j in 1:10) {
      nueva_x <- x + Valor_cartas_mano_blanda[j]
      G_continuar <- G_continuar + Probabilidades_sacar_carta[j] * Calculo_G_0(nueva_
    }
    if (2*G_continuar > G_estrella) {
      estrategia_doblarse_NoDoblarse[as.character(x), b] <- "Doblarse"
      ganancia_doblarse_NoDoblarse [as.character(x),b] <- 2*G_continuar
    } else {
      estrategia_doblarse_NoDoblarse[as.character(x), b] <- "No doblarse"
```

```
      ganancia_doblarse_NoDoblarse [as.character(x),b] <- G_estrella
    }
  }
}

kable(estrategia_doblarse_NoDoblarse,
      caption = "Tabla de procedimientos para decidir si doblarse o no",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
                full_width = F, font_size = 12)

kable(round(ganancia_doblarse_NoDoblarse,5),
      caption = "Tabla de ganancias al doblarse o no hacerlo",
      align = "c") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
                full_width = F, font_size = 12)
```

1.5. Codigo para la ganancia y estrategia en caso de doblarse o no

```
estrategia_abrirse_NoAbrirse <- matrix("", nrow = 10, ncol = 10)
rownames(estrategia_abrirse_NoAbrirse) <- as.character(2:11)
colnames(estrategia_abrirse_NoAbrirse) <- Cartas  # del 2 al As
ganancia_abrirse_NoAbrirse <- matrix(NA, nrow = 10, ncol = 10)
rownames(ganancia_abrirse_NoAbrirse) <- as.character(2:11)
colnames(ganancia_abrirse_NoAbrirse) <- Cartas  # del 2 al As

for (z in 2:10) {
  for (b in Cartas) {
    G_estrella=ganancia_G_optima[as.character(2*z),b]
    # Esperanza de continuar
    G_continuar <- 0
    for (j in 1:10) {
      nueva_z <- z + Valor_cartas_mano_blanda[j]
      G_continuar <- G_continuar + Probabilidades_sacar_carta[j] * ganancia_G_optima[a
    }
    if (2*G_continuar > G_estrella) {
      estrategia_abrirse_NoAbrirse[as.character(z), b] <- "Abrirse"
      ganancia_abrirse_NoAbrirse[as.character(z),b] <- 2*G_continuar
    } else {
      estrategia_abrirse_NoAbrirse[as.character(z), b] <- "No abrirse"
      ganancia_abrirse_NoAbrirse[as.character(z),b] <- G_estrella
    }
  }
}
```

```r
for (b in Cartas) {
  G_estrella=ganancia_G_optima_mano_blanda[as.character(12),b]
  # Esperanza de continuar
  G_continuar <- 0
  for (j in 1:10) {
    nueva_z <- z + Valor_cartas_mano_blanda[j]
    G_continuar <- G_continuar + Probabilidades_sacar_carta[j] * Calculo_G_0(nueva_z,
  }
  if (2*G_continuar > G_estrella) {
    estrategia_abrirse_NoAbrirse[as.character(11), b] <- "Abrirse"
    ganancia_abrirse_NoAbrirse[as.character(11),b] <- 2*G_continuar
  } else {
    estrategia_abrirse_NoAbrirse[as.character(11), b] <- "No abrirse"
    ganancia_abrirse_NoAbrirse[as.character(11),b] <- G_estrella
  }
}
rownames(estrategia_abrirse_NoAbrirse) <- paste0(Cartas,"-",Cartas)
rownames(ganancia_abrirse_NoAbrirse) <- paste0(Cartas,"-",Cartas)

kable(estrategia_abrirse_NoAbrirse,
      caption = "Tabla de procedimientos para decidir si abrirse o no",
      align = "c") %>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
              full_width = F, font_size = 12)

kable(round(ganancia_abrirse_NoAbrirse,5),
      caption = "Tabla de ganancias al abrirse o no hacerlo",
      align = "c") %>%
kable_styling(bootstrap_options = c("striped", "hover", "condensed"),
              full_width = F, font_size = 12)
```