# Blackjack

```r
# change working directory
setwd("~/Documents/Projet_R/Blackjack/01_Input")

# read csv
df <- read.csv2("deck.csv")
```

## fix the seed if necessary

```r
set.seed(2022)
```

```r
# combine into four full decks
four_full_decks <- rbind(df, df, df, df)

# add the variable id (eg. 1,2,3, etc) so that each card will have its unique id
four_full_decks$id <- seq(1, nrow(four_full_decks), 1)
```

## Definition of global variables

```r
casino_deck_current <- NULL

dealer_hand <- NULL

my_hand <- NULL
```

## Definition of the function shuffle_deck

```r
shuffle_deck <- function (casino_deck) {

  # use the sample function to shuffle deck randomly
  res_casino_deck <- casino_deck[sample(1:nrow(casino_deck)), ]

  return(res_casino_deck)
  }
```

## Definition of the function start_game

```r
start_game <- function() {
  # this function shuffles deck, deals 2 cards for you and dealer. and prints state

  # suffle deck
  casino_deck_shuffled <- shuffle_deck(four_full_decks)

  # tidy up
  # use <<- to overwrite global variables when we are inside a function
  dealer_hand <<- NULL
  my_hand <<- NULL

  # deal first card for the dealer
  dealer_hand <<- casino_deck_shuffled[1, ]

  # deal first card for me
  my_hand <<- casino_deck_shuffled[2, ]

  # deal second card for the dealer
  dealer_hand <<- rbind(dealer_hand,  casino_deck_shuffled[3, ])

  # deal second card for me
  my_hand <<- rbind(my_hand,  casino_deck_shuffled[4, ])

  casino_deck_id_current <- setdiff(casino_deck_shuffled$id, union(my_hand$id, dealer_hand$id))
  casino_deck_current <<- subset(casino_deck_shuffled, id %in% casino_deck_id_current)

  sum_my_hand <- 0

  sum_dealer_hand <- 0

  print("******************Current state******************")

  print("Dealers hand:")
  for (i in 1:nrow(dealer_hand)) {
    print(paste(dealer_hand[i,"face"],
                dealer_hand[i,"suite"],
                dealer_hand[i,"value"]))

    sum_dealer_hand <- sum_dealer_hand + dealer_hand[i,"value"]
  }
  print(paste("sum", sum_dealer_hand))



  print("Your hand:")
  for (i in 1:nrow(my_hand)) {
    print(paste(my_hand[i,"face"],
                my_hand[i,"suite"],
                my_hand[i,"value"]))

    sum_my_hand <- sum_my_hand + my_hand[i,"value"]
  }
  print(paste("sum", sum_my_hand))
  # $$$ COMPUTE CHANCES $$$ ------------------------------------------------------
  # To compute chances mean to count a probability that next card for you will bring you sum of points
  if ((sum_my_hand >= sum_dealer_hand ) & (sum_my_hand < 22)){
    chances <- 1
  } else {
    distance_to_21 <- 21 - sum_my_hand        # this will be the maximum value of my desired card
    distance_to_beat_dealer <- sum_dealer_hand - sum_my_hand  # this will be the minimum value of my de
    # then, all cards [distance_to_beat_dealer, distance_to_21] will be values of my desired next card
    # and Proba(my chances to win with my next card) = Proba(getting one of these cards as my next card
    # = number of these cards avaible / total number of cards in the deck

    if (distance_to_21 >= 10) {
      total_number_lucky_cards_inf_10 <- (10 - distance_to_beat_dealer) * 4
      total_number_lucky_cards_10 <- 4 * 4
      total_number_lucky_cards <- total_number_lucky_cards_inf_10 + total_number_lucky_cards_10
    } else {
       total_number_lucky_cards <- (distance_to_21 - distance_to_beat_dealer + 1) * 4
    }


    if (distance_to_beat_dealer > 10) {
      # in this case, next card will not be big enough to win
      total_number_lucky_cards <- 0
    }

    # now we need to consider if these lucky cards are already distributed
    for (lucky_number in distance_to_beat_dealer : distance_to_21) {
      for (j in 1:nrow(my_hand)) {
        if  (my_hand[j,"value"] == lucky_number) {
          total_number_lucky_cards <- total_number_lucky_cards - 1
        }
      }

      for (j in 1:nrow(dealer_hand)) {
        if  (dealer_hand[j,"value"] == lucky_number) {
          total_number_lucky_cards <- total_number_lucky_cards - 1
        }
      }
    }

    chances <- max(total_number_lucky_cards,0) / nrow(casino_deck_current)
  }

  print(paste("chances", chances*100, "%"))


  print("******************End of current state******************")
}
```

## Definition of the function deal

```r
deal <- function() {
  # This function deals you a card and prints state

  # deal a card to me

  my_hand <<- rbind(my_hand, casino_deck_current[1,])



  # remove this card from the deck
  casino_deck_id_current <- setdiff(casino_deck_current$id, union(my_hand$id, dealer_hand$id))
  casino_deck_current <<- subset(casino_deck_current, id %in% casino_deck_id_current)




  sum_my_hand <- 0

  sum_dealer_hand <- 0

  print("******************Current state******************")

  print("Dealers hand:")
  for (i in 1:nrow(dealer_hand)) {
    print(paste(dealer_hand[i,"face"],
                dealer_hand[i,"suite"],
                dealer_hand[i,"value"]))

    sum_dealer_hand <- sum_dealer_hand + dealer_hand[i,"value"]
  }
  print(paste("sum", sum_dealer_hand))



  print("Your hand:")
  for (i in 1:nrow(my_hand)) {
    print(paste(my_hand[i,"face"],
                my_hand[i,"suite"],
                my_hand[i,"value"]))

    sum_my_hand <- sum_my_hand + my_hand[i,"value"]
  }
  print(paste("sum", sum_my_hand))


  # $$$ COMPUTE CHANCES $$$ ---------------------------------------------------
  # To compute chances mean to count a probability that next card for you will bring you sum of points
  if ((sum_my_hand >= sum_dealer_hand ) & (sum_my_hand < 22)){
    chances <- 1
  } else if (sum_my_hand < 22) {
    distance_to_21 <- 21 - sum_my_hand        # this will be the maximum value of my desired card
    distance_to_beat_dealer <- sum_dealer_hand - sum_my_hand    # this will be the minimum value of my
    # then, all cards [distance_to_beat_dealer, distance_to_21] will be values of my desired next card
    # and Proba(my chances to win with my next card) = Proba(getting one of these cards as my next card
    # = number of these cards avaible / total number of cards in the deck

    if (distance_to_21 >= 10) {
      total_number_lucky_cards_inf_10 <- (10 - distance_to_beat_dealer) * 4
      total_number_lucky_cards_10 <- 4 * 4
      total_number_lucky_cards <- total_number_lucky_cards_inf_10 + total_number_lucky_cards_10
    } else {
       total_number_lucky_cards <- (distance_to_21 - distance_to_beat_dealer + 1) * 4
    }

    if (distance_to_beat_dealer > 10) {
      # in this case, next card will not be big enough to win
      total_number_lucky_cards <- 0
    }

    # now we need to consider if these lucky cards are already distributed
    for (lucky_number in distance_to_beat_dealer : distance_to_21) {
      for (j in 1:nrow(my_hand)) {
        if  (my_hand[j,"value"] == lucky_number) {
          total_number_lucky_cards <- total_number_lucky_cards - 1
        }
      }

      for (j in 1:nrow(dealer_hand)) {
        if  (dealer_hand[j,"value"] == lucky_number) {
          total_number_lucky_cards <- total_number_lucky_cards - 1
        }
      }
    }

    chances <- max(total_number_lucky_cards,0) / nrow(casino_deck_current)

  } else {     # case when the sum of my cards already > 21, then I lose, so the function stop_game sho

    chances <- 0

    stop_game()

  }

  print(paste("chances", chances*100, "%"))

  print("******************End of current state******************")



}
```

## Definition of the function stop_game

```
stop_game <- function(){
  # This function prints result: win or loose

  sum_my_hand <- 0
  sum_dealer_hand <- 0

  for (i in 1:nrow(dealer_hand)) {
    sum_dealer_hand <- sum_dealer_hand + dealer_hand[i,"value"]
  }

  for (i in 1:nrow(my_hand)) {
    sum_my_hand <- sum_my_hand + my_hand[i,"value"]
  }

  # If my card sum more than 21 I lose.
  if (sum_my_hand > 21) {
    print("lose")
  } else if (sum_my_hand >= sum_dealer_hand) { # I win if my card sum is more or equal than dealers car
    print("win")
  } else  {
    print("lose")
  }
}
```

Notice that by the definition of success, I win if my card sum is more or EQUAL than dealers card sum.

## Example 1

```
# Game starts when dealer shuffle all cards and give 2 card for you and 2 for himself.
start_game()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "four clubs 4"
## [1] "jack spades 10"
## [1] "sum 14"
## [1] "Your hand:"
## [1] "three hearts 3"
## [1] "four clubs 4"
## [1] "sum 7"
## [1] "chances 13.2352941176471 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "four clubs 4"
## [1] "jack spades 10"
## [1] "sum 14"
## [1] "Your hand:"
## [1] "three hearts 3"
## [1] "four clubs 4"
## [1] "king hearts 10"
## [1] "sum 17"
## [1] "chances 100 %"
## [1] "******************End of current state******************"
```

```
stop_game()
```

```
## [1] "win"
```

### Example 2

```
# Game starts when dealer shuffle all cards and give 2 card for you and 2 for himself.
start_game()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "eight spades 8"
## [1] "seven clubs 7"
## [1] "sum 15"
## [1] "Your hand:"
## [1] "six spades 6"
## [1] "two spades 2"
## [1] "sum 8"
## [1] "chances 12.7450980392157 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "eight spades 8"
## [1] "seven clubs 7"
## [1] "sum 15"
## [1] "Your hand:"
## [1] "six spades 6"
## [1] "two spades 2"
## [1] "two diamonds 2"
## [1] "sum 10"
## [1] "chances 16.256157635468 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "eight spades 8"
## [1] "seven clubs 7"
## [1] "sum 15"
## [1] "Your hand:"
## [1] "six spades 6"
## [1] "two spades 2"
## [1] "two diamonds 2"
## [1] "ten clubs 10"
## [1] "sum 20"
## [1] "chances 100 %"
## [1] "******************End of current state******************"
```

```
stop_game()
```

```
## [1] "win"
```

### Example 3

```
start_game()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "ten diamonds 10"
## [1] "queen hearts 10"
## [1] "sum 20"
## [1] "Your hand:"
## [1] "seven diamonds 7"
## [1] "three clubs 3"
## [1] "sum 10"
## [1] "chances 6.86274509803922 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "ten diamonds 10"
## [1] "queen hearts 10"
## [1] "sum 20"
## [1] "Your hand:"
## [1] "seven diamonds 7"
## [1] "three clubs 3"
## [1] "queen diamonds 10"
## [1] "sum 20"
## [1] "chances 100 %"
## [1] "******************End of current state******************"
```

```
stop_game()
```

```
## [1] "win"
```

## Example 4

```
start_game()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "king hearts 10"
## [1] "ace spades 1"
## [1] "sum 11"
## [1] "Your hand:"
## [1] "four clubs 4"
## [1] "two hearts 2"
## [1] "sum 6"
## [1] "chances 17.156862745098 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "king hearts 10"
## [1] "ace spades 1"
## [1] "sum 11"
## [1] "Your hand:"
## [1] "four clubs 4"
## [1] "two hearts 2"
## [1] "three spades 3"
## [1] "sum 9"
## [1] "chances 21.6748768472906 %"
## [1] "******************End of current state******************"
```

```
deal()
```

```
## [1] "******************Current state******************"
## [1] "Dealers hand:"
## [1] "king hearts 10"
## [1] "ace spades 1"
## [1] "sum 11"
## [1] "Your hand:"
## [1] "four clubs 4"
## [1] "two hearts 2"
## [1] "three spades 3"
## [1] "four diamonds 4"
## [1] "sum 13"
## [1] "chances 100 %"
## [1] "******************End of current state******************"
```

```r
stop_game()
```

```
## [1] "win"
```