Homework_1

Zongyi Liu

2023-09-02

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Problem 1

a, b

```
wine_data<-read.csv("wine.data")

colnames(wine_data) <- c("class", "Alcohol", "Malicacid", "Ash","Alcalinity of ash", "Magnesium", "Total

head(wine_data)
```

```
##   class Alcohol Malicacid  Ash Alcalinity of ash Magnesium Total phenols
## 1     1   13.20      1.78 2.14              11.2       100          2.65
## 2     1   13.16      2.36 2.67              18.6       101          2.80
## 3     1   14.37      1.95 2.50              16.8       113          3.85
## 4     1   13.24      2.59 2.87              21.0       118          2.80
## 5     1   14.20      1.76 2.45              15.2       112          3.27
## 6     1   14.39      1.87 2.45              14.6        96          2.50
##   Flavanoids Nonflavanoid_phenols Proanthocyanins Color_intensity  Hue
## 1       2.76                 0.26            1.28            4.38 1.05
## 2       3.24                 0.30            2.81            5.68 1.03
## 3       3.49                 0.24            2.18            7.80 0.86
## 4       2.69                 0.39            1.82            4.32 1.04
## 5       3.39                 0.34            1.97            6.75 1.05
## 6       2.52                 0.30            1.98            5.25 1.02
##   OD280_OD315_of_diluted_wines Proline
```

```
## 1                          3.40    1050
## 2                          3.17    1185
## 3                          3.45    1480
## 4                          2.93     735
## 5                          2.85    1450
## 6                          3.58    1290
```

c

1, 2

```
mean(wine_data$Alcohol)
```

```
## [1] 12.99367
```

```
wine_data_1<-wine_data%>%filter(class=="1")
mean(wine_data_1$Alcohol)
```

```
## [1] 13.73638
```

```
wine_data_2<-wine_data%>%filter(class=="2")
mean(wine_data_2$Alcohol)
```

```
## [1] 12.27873
```

```
wine_data_3<-wine_data%>%filter(class=="3")
mean(wine_data_3$Alcohol)
```

```
## [1] 13.15375
```

```
#wine_data%>%filter(class=="1")%>%filter(Alcohol)%>%mean()
```

> From the result we can say that the class 1 has the hihest alcohol content, while the class 2 has the lowest alcohol content.

```
wine_data%>%filter(Magnesium>"114")%>%nrow()
```

```
## [1] 122
```

```
wine_data_1%>%filter(Magnesium>"114")%>%nrow()
```

```
## [1] 30
```

```
wine_data_2%>%filter(Magnesium>"114")%>%nrow()
```

```
## [1] 59
```

```
wine_data_3%>%filter(Magnesium>"114")%>%nrow()
```

```
## [1] 33
```

There are 122 wines having higher levels of magnesium than the average level of German beers; there are 30 in class 1, 59 in class 2, and 33 in class 3 having higher levels respectively.

d

```
mean_0<-wine_data%>%summarise_all(mean)
mean_1<-wine_data_1%>%summarise_all(mean)
mean_2<-wine_data_2%>%summarise_all(mean)
mean_3<-wine_data_3%>%summarise_all(mean)

combined<-rbind(mean_0,mean_1,mean_2,mean_3)
combined
```

```
##      class  Alcohol Malicacid      Ash Alcalinity of ash Magnesium
## 1 1.943503 12.99367  2.339887 2.366158          19.51695  99.58757
## 2 1.000000 13.73638  2.015862 2.456034          17.06207 105.98276
## 3 2.000000 12.27873  1.932676 2.244789          20.23803  94.54930
## 4 3.000000 13.15375  3.333750 2.437083          21.41667  99.31250
##   Total phenols Flavanoids Nonflavanoid_phenols Proanthocyanins Color_intensity
## 1      2.292260  2.0234463            0.3623164        1.586949        5.054802
## 2      2.840862  2.9810345            0.2901724        1.892586        5.526379
## 3      2.258873  2.0808451            0.3636620        1.630282        3.086620
## 4      1.678750  0.7814583            0.4475000        1.153542        7.396250
##         Hue OD280_OD315_of_diluted_wines   Proline
## 1 0.9569831                     2.604294  745.0960
## 2 1.0624138                     3.144655 1116.5862
## 3 1.0562817                     2.785352  519.5070
## 4 0.6827083                     1.683542  629.8958
```

e

```
summary(aov(class~Ash,data=combined))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Ash          1 0.0123  0.0123   0.012  0.922
## Residuals    2 1.9901  0.9950
```

Here from the results we can say that p=0.922>0.05, so there are no huge differences between those three classes

To write the process of t-test. We have the null hypothesis as $H_0 : \tau_1 = \tau_2 = ... = \tau_k$ and $F = \frac{\sum_{i=1}^{k} n_i(\bar{y}_{i.} - \bar{y}_{..})^2/(k-1)}{\sum_{i=1}^{k}\sum_{j=1}^{ni}(y_{ij} - \bar{y}_{i.})^2/(N-k)}$

Problem 2

a

```
library(VeryLargeIntegers)
```

```
isPerfectPower<-function(x, power){
  is.perfectpow(x)
  root=x^(1/power)
  print(is.perfectpow(x))
  return(root)
}

isPerfectPower(27,3)
```

```
## [1] TRUE
```

```
## [1] 3
```

```
isPerfectPower(11,3)
```

```
## [1] FALSE
```

```
## [1] 2.22398
```

b

```
#findRootPower<-function(x){
#  for(x)
#}
```

Problem 3

a

```
# Function to determine the name of a poker hand
get_poker_hand_name <- function(suits, ranks) {
  is_flush <- length(unique(suits)) == 1
  is_straight <- max(ranks) - min(ranks) == 4 && length(unique(ranks)) == 5
  rank_counts <- table(ranks)

  if(is_flush && is_straight&&max(ranks)==14){
    return("Royal Flush")
  }
  else if (is_flush && is_straight) {
```

```
    return("Straight Flush")
  } else if (max(rank_counts) == 4) {
    return("Four of a Kind")
  } else if (max(rank_counts) == 3 && sum(rank_counts == 2) == 2) {
    return("Full House")
  } else if (is_flush) {
    return("Flush")
  } else if (is_straight) {
    return("Straight")
  } else if (max(rank_counts) == 3) {
    return("Three of a Kind")
  } else if (sum(rank_counts == 2) == 4) {
    return("Two Pair")
  } else if (sum(rank_counts == 2) == 2) {
    return("One Pair")
  } else {
    return("High Card")
  }
}

# Function to simulate a round of 5-card stud poker
simulate_poker_round <- function(num_players) {
  if (num_players < 2 || num_players > 10) {
    stop("Number of players must be between 2 and 10.")
  }

  # Define the suits and ranks
  suits <- c("Hearts", "Diamonds", "Clubs", "Spades")
  ranks <- c(2:14)

  # Create a deck of cards
  deck <- expand.grid(Rank = ranks, Suit = suits)

  # Shuffle the deck
  deck <- deck[sample(nrow(deck)), ]

  # Deal cards to each player
  hands <- vector("list", num_players)
  for (i in 1:num_players) {
    hands[[i]] <- deck[((i - 1) * 5 + 1):(i * 5), ]
  }

  # Determine and print the name of each hand
  for (i in 1:num_players) {
    cat("Player", i, "Hand:", "\n")
    hand <- hands[[i]]
    hand_suits <- hand$Suit
    hand_ranks <- hand$Rank
    cat("Cards:", paste(hand_ranks, hand_suits), "\n")
    cat("Hand Name:", get_poker_hand_name(hand_suits, hand_ranks), "\n\n")
  }
}
```

```r
# Example usage:
simulate_poker_round(8) # Simulate a round with 4 players
```

```
## Player 1 Hand:
## Cards: 2 Diamonds 3 Hearts 10 Diamonds 5 Diamonds 8 Clubs
## Hand Name: High Card
##
## Player 2 Hand:
## Cards: 10 Spades 3 Spades 2 Hearts 5 Hearts 14 Clubs
## Hand Name: High Card
##
## Player 3 Hand:
## Cards: 4 Diamonds 8 Hearts 12 Hearts 8 Diamonds 12 Diamonds
## Hand Name: One Pair
##
## Player 4 Hand:
## Cards: 9 Hearts 14 Hearts 3 Diamonds 11 Spades 11 Hearts
## Hand Name: High Card
##
## Player 5 Hand:
## Cards: 7 Diamonds 13 Hearts 11 Diamonds 14 Spades 4 Clubs
## Hand Name: High Card
##
## Player 6 Hand:
## Cards: 9 Diamonds 4 Spades 12 Spades 13 Clubs 7 Hearts
## Hand Name: High Card
##
## Player 7 Hand:
## Cards: 7 Clubs 11 Clubs 2 Clubs 10 Hearts 3 Clubs
## Hand Name: High Card
##
## Player 8 Hand:
## Cards: 10 Clubs 8 Spades 14 Diamonds 7 Spades 2 Spades
## Hand Name: High Card
```

b

> This code can not run without modifications. Initially the code for rank was `ranks <- c(2:10, "Jack", "Queen", "King", "Ace")`, but when we tried to calculate the rank number and the cofiguration of our cards, the system can not calculate factors and the Console says `'max' not meaningful for factors`. Thus I changed this line to `ranks <- c(1:13)`.

c

```r
# Function to determine the name of a poker hand
get_poker_hand_name <- function(suits, ranks) { #Here we defined a function
  is_flush <- length(unique(suits)) == 1 #If all cards are in the same suite then we will define it as
  is_straight <- max(ranks) - min(ranks) == 4 &&  length(unique(ranks)) == 5 #If the highest rank mins
  rank_counts <- table(ranks) #table() performs a tabulation of categorical variable and gives its freq
```

```r
  if(is_flush && is_straight&&max(ranks)==14){
    return("Royal Flush")
  }
  else if (is_flush && is_straight) { #If it is flush and straight, then we will define it as straight
    return("Straight Flush")
  } else if (max(rank_counts) == 4) { #Else if the maximum of rank_counts is 4, we define it as Four of
    return("Four of a Kind")
  } else if (max(rank_counts) == 3 && sum(rank_counts == 2) == 2) { #Else if the maximum of rank_counts
    return("Full House")
  } else if (is_flush) { #If it satisfies is_flush, we will define it as Flush
    return("Flush")
  } else if (is_straight) {#If it satisfies is_flush, we will define it as Straight
    return("Straight")
  } else if (max(rank_counts) == 3) {#Else if the maximum of rank_counts is 3, we define it as Three of
    return("Three of a Kind")
  } else if (sum(rank_counts == 2) == 4) {#If there four cards in which each two have the same rank_cou
    return("Two Pair")
  } else if (sum(rank_counts == 2) == 2) {#If there two cards and they have the same rank_counts, then
    return("One Pair")
  } else {#If all above are not satisfied, then we will say it as High Card
    return("High Card")
  }
}

# Function to simulate a round of 5-card stud poker


simulate_poker_round <- function(num_players) { # Here we defined a function
  if (num_players < 2 || num_players >10) { #If number of players are below 2 or above 8, then we will
    stop("Number of players must be between 2 and 10.")#This function stops execution of the current ex
  }

  suits <- c("Hearts", "Diamonds", "Clubs", "Spades")
  ranks <- c(2:14) #Here we define the suits and ranks; to avoid the problem of calculating a factor, w

  deck <- expand.grid(Rank = ranks, Suit = suits)  #Here we create a deck of cards

  deck <- deck[sample(nrow(deck)), ]#We shuffle the deck with the sample() function

  hands <- vector("list", num_players)#We vectorize it
  for (i in 1:num_players) {#We use a for loop here to add cards to hards
    hands[[i]] <- deck[((i - 1) * 5 + 1):(i * 5), ]
  }

  for (i in 1:num_players) {#Determine and print the name of each hand
    cat("Player", i, "Hand:", "\n") #Here we use cat() to concatenate together several objects, and \n
    hand <- hands[[i]] #[[]] = can extract one element from list or data frame, returned object
    hand_suits <- hand$Suit#We extract suit from hand
    hand_ranks <- hand$Rank#We extract rank from hand
    cat("Cards:", paste(hand_ranks, hand_suits), "\n")#Here we use cat() to concatenate together severa
    cat("Hand Name:", get_poker_hand_name(hand_suits, hand_ranks), "\n\n") #We utilize the get_poker_ha
  }
}
```

d

The inputs and outputs are as described above

The hands are valid

All the names of hands are correct

If it is larger than 10, then the function will stop to execute because there should be no more 10 players as there are only 52 cards in a standard deck