Logic for Mobile App for Lottery Addiction

Bena

2024-01-31

Background

As a data analyst at a medical institute, I've been assigned to assist in the development of a mobile app intended to guide lottery addicts through exercises that will let them better estimate their chances of winning.We'll work collaboratively with engineers & here we're going to build a logic for the app aimed at helping lottery addicts realistically understand their chances of winning using probabilities. Hopefully, the app will help the addicts realize that buying too many tickets will do little to improve their chances of winning thus stopping them from buying the tickets in an unhealthy manner. We'll apply concepts we'd learnt including: - How to calculate theoretical and empirical probabilities - How to use probability rules to calculate different probabilities - How to use combinations and permutations to count the sizes of sample spaces

We'll build functions that can answer users questions like: - What is the probability of winning the big prize with a single ticket? - What is the probability of winning the big prize if we play 40 different tickets (or any other number)? - What is the probability of having at least five (or four, or three, or two) winning numbers on a single ticket?

Understanding the data

We considered historical data coming from the national 6/49 lottery game in Canada. It was one of the first Canadian lotteries game to allow players to pick their own numbers. The data set has data for 3,665 drawings, dating from 1982 to 2018. It can be accessed using this link https://www.kaggle.com/datasets/datascienceai/lottery-dataset (https://www.kaggle.com/datasets/datascienceai/lottery-dataset)

Core functions

Needed because we'll calculate probabilities and combinations repeatedly. In the 6/49 lottery, six numbers are drawn from a set of 49 numbers that range from 1 to 49. The drawing is done without replacement.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0     ✓ purrr   1.0.1
## ✓ tibble  3.1.8     ✓ dplyr   1.1.0
## ✓ tidyr   1.3.0     ✓ stringr 1.5.0
## ✓ readr   2.1.3     ✓ forcats 1.0.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
factorial <- function(n) {
    final_product <- 1
    for (i in 1:n) {
        final_product <- final_product * i
    }
    return(final_product)
}
```

```
combination <- function(n,k) {
      numerator <- factorial(n)
      denominator <- factorial(k)*factorial(n-k)

    return(numerator/denominator)
}
```

One-ticket Probability

A player wins the big prize if the six numbers on their tickets match all the six numbers drawn.i.e., If a player has a ticket with the numbers {13, 22, 24, 27, 42, 44}, he only wins the big prize if the numbers drawn are {13, 22, 24, 27, 42, 44}.

```
one_ticket_probability <- function(sixnums) {
      total_combinations <- combination(49,6)
      prob <- (1/total_combinations)*100
      pretty_prob <- sprintf("%1.9f", prob)
      message <- paste("You have a ", pretty_prob, "% chance of winning the big prize.", sep
= " ")

   return(message)
}
```

```
chance1 <- one_ticket_probability(c(1,2,3,4,5,6))
```

The 2 code chunks above take in a list of six unique numbers and prints the probability of winning in a way that's easy to understand. It starts with calculating the total number of possible outcomes. There are 49 possible numbers, and six numbers are sampled without replacement. The user inputs 1 combination since there's only 1 successful outcome. Under the hood, the six numbers will come as an R vector, which will serve as the single input to our function. We use that to calculate the probability for 1 ticket. The printf() & paste() functions are used to print a user friendly message as the output.

Trying with other combination of vectors

```
chance2 <- one_ticket_probability(c(20,9,34,16,28,54))
chance3 <- one_ticket_probability(c(53,19,10,17,61,38))
chance4 <- one_ticket_probability(c(13,22,24,27,42,44))
```

Historical Data Check for Canada Lottery

Users should also be able to compare their ticket against past winning combinations in the historical lottery data in Canada. This functionality will allow users to determine whether they would have ever won by now.

```
lottery_data <- read_csv("649.csv")
```

```
## Rows: 3665 Columns: 11
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): DRAW DATE
## dbl (10): PRODUCT, DRAW NUMBER, SEQUENCE NUMBER, NUMBER DRAWN 1, NUMBER DRAW...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(lottery_data)
```

```
## Rows: 3,665
## Columns: 11
## $ PRODUCT           <dbl> 649, 649, 649, 649, 649, 649, 649, 649, 649, 649, 64…
## $ `DRAW NUMBER`     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1…
## $ `SEQUENCE NUMBER` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
## $ `DRAW DATE`       <chr> "6/12/1982", "6/19/1982", "6/26/1982", "7/3/1982", "…
## $ `NUMBER DRAWN 1`  <dbl> 3, 8, 1, 3, 5, 8, 18, 7, 5, 4, 7, 11, 7, 25, 8, 9, 4…
## $ `NUMBER DRAWN 2`  <dbl> 11, 33, 6, 9, 14, 20, 25, 16, 10, 15, 9, 17, 14, 28,…
## $ `NUMBER DRAWN 3`  <dbl> 12, 36, 23, 10, 21, 21, 28, 17, 23, 30, 21, 19, 17, …
## $ `NUMBER DRAWN 4`  <dbl> 14, 37, 24, 13, 31, 25, 33, 31, 27, 37, 33, 20, 20, …
## $ `NUMBER DRAWN 5`  <dbl> 41, 39, 27, 20, 34, 31, 36, 40, 37, 46, 38, 36, 37, …
## $ `NUMBER DRAWN 6`  <dbl> 43, 41, 39, 43, 47, 41, 42, 48, 38, 48, 42, 43, 47, …
## $ `BONUS NUMBER`    <dbl> 13, 9, 34, 34, 45, 33, 7, 26, 33, 3, 45, 9, 34, 3, 3…
```

```
View(lottery_data)
print(dim(lottery_data))
```

```
## [1] 3665   11
```

```
head(lottery_data,3)
```

```
## # A tibble: 3 × 11
##   PRODUCT `DRAW NUMBER` SEQUEN…¹ DRAW …² NUMBE…³ NUMBE…⁴ NUMBE…⁵ NUMBE…⁶ NUMBE…⁷
##     <dbl>         <dbl>    <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     649             1        0 6/12/1…       3      11      12      14      41
## 2     649             2        0 6/19/1…       8      33      36      37      39
## 3     649             3        0 6/26/1…       1       6      23      24      27
## # … with 2 more variables: `NUMBER DRAWN 6` <dbl>, `BONUS NUMBER` <dbl>, and
## #   abbreviated variable names ¹`SEQUENCE NUMBER`, ²`DRAW DATE`,
## #   ³`NUMBER DRAWN 1`, ⁴`NUMBER DRAWN 2`, ⁵`NUMBER DRAWN 3`, ⁶`NUMBER DRAWN 4`,
## #   ⁷`NUMBER DRAWN 5`
```

```
tail(lottery_data,3)
```

```
## # A tibble: 3 × 11
##   PRODUCT `DRAW NUMBER` SEQUEN…¹ DRAW …² NUMBE…³ NUMBE…⁴ NUMBE…⁵ NUMBE…⁶ NUMBE…⁷
##     <dbl>         <dbl>    <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1     649          3589        0 6/13/2…       6      22      24      31      32
## 2     649          3590        0 6/16/2…       2      15      21      31      38
## 3     649          3591        0 6/20/2…      14      24      31      35      37
## # … with 2 more variables: `NUMBER DRAWN 6` <dbl>, `BONUS NUMBER` <dbl>, and
## #   abbreviated variable names ¹`SEQUENCE NUMBER`, ²`DRAW DATE`,
## #   ³`NUMBER DRAWN 1`, ⁴`NUMBER DRAWN 2`, ⁵`NUMBER DRAWN 3`, ⁶`NUMBER DRAWN 4`,
## #   ⁷`NUMBER DRAWN 5`
```

```
lottery_data2 <- as_tibble(lottery_data)
```

A New Data Structure

Given

```
data1 <- c(1, 3, 5)
data2 <- c(2, 4, 6)
data3 <- c(8, 9, 7)
```

Learn & practice using pmap(). Practice creating and accessing data in lists

```
unnamed_list <- list(data1, data2, data3)
first_vector <- unnamed_list[[1]]
```

```
named_list <- list(first=data1, second=data2, third=data3)
first_item_sum <- named_list$first[1]+named_list$second[1]+named_list$third[1]
```

```
library(purrr)
averages <- pmap(unnamed_list, function(x,y,z) {(x+y+z)/3})
first_average <- unlist(averages)[1]
```

Function for Historical Data Check

We start with extracting all the winning six numbers from the historical data set into an R vector

```
historical_lots <- pmap(
  list(a <- lottery_data$`NUMBER DRAWN 1`,
       b <- lottery_data$`NUMBER DRAWN 2`,
       c <- lottery_data$`NUMBER DRAWN 3`,
       d <- lottery_data$`NUMBER DRAWN 4`,
       e <- lottery_data$`NUMBER DRAWN 5`,
       f <- lottery_data$`NUMBER DRAWN 6`),
  function(a,b,c,d,e,f) {c(a,b,c,d,e,f)}
)
```

The function below takes in two inputs: an R vector containing the user numbers and the list containing the sets of the winning numbers from historical lots above. It compares the numbers given by the user against the list we created using the historical_lots() function above. If the user numbers match the winning lot, then it returns TRUE. If not, it returns FALSE. The end result of the comparison is a vector of Boolean values. Which are unlisted & added up. This is displayed as the number of past matches.

```
check_historical_occurrence <- function(lots, hist_lots=historical_lots) {
    historical_matches <- map(historical_lots, function(x) {setequal(x, lots)})
    num_past_matches <- sum(unlist(historical_matches))
    message <- paste("The combination you entered has appeared", num_past_matches, "times in
the past.",
                     "You have a 0.000007151% chance of winning the big prize in the next dra
wing.")

    return(message)
}
```

Trying out new inputs

```
check_hist1 <- check_historical_occurrence(c(3, 11, 12, 14, 41, 43))
check_hist2 <- check_historical_occurrence(c(5, 13, 22, 30, 38, 46))
check_hist3 <- check_historical_occurrence(c(20,15,19,36,47,10))
check_hist4 <- check_historical_occurrence(c(14,24,31,35,37,46))
```

It’s highly unlikely that 6 consecutive numbers would get picked together, so we won’t see it in the data set. check_hist1 was picked from historical data, so it has appeared once before.

Multi-ticket Probability

Lottery addicts usually play more than one ticket on a single drawing, thinking that this might increase their chances of winning significantly. Here, we’re going to write a function that will allow the users to calculate the chances of winning for any number of different tickets. The user will input the number of different tickets they want to play (without inputting the specific combinations they intend to play)

```
multi_ticket_prob <- function(n) {
  total_combinations <- combination(49,6)
  prob_win <- (n/total_combinations)*100
  pretty_prob <- sprintf("%1.9f",prob_win)
  message <- paste("You have a", pretty_prob,"% chance of winning the big prize.")

  return(message)
}
```

The function above starts by calculating the total number of possible outcomes. n is the number of successful outcomes given by the number of tickets the user intends to play. We use n & total to calculate a player's probability of winning. We check probability of a gambler who tried several times as shown below

```
trials <- c(1, 10, 100, 10000, 1000000, 6991908, 13983816)
for(n in trials) {
  print(paste("For ", n, " tickets, ",  multi_ticket_prob(n)))
}
```

```
## [1] "For  1  tickets,  You have a 0.000007151 % chance of winning the big prize."
## [1] "For  10  tickets,  You have a 0.000071511 % chance of winning the big prize."
## [1] "For  100  tickets,  You have a 0.000715112 % chance of winning the big prize."
## [1] "For  10000  tickets,  You have a 0.071511238 % chance of winning the big prize."
## [1] "For  1e+06  tickets,  You have a 7.151123842 % chance of winning the big prize."
## [1] "For  6991908  tickets,  You have a 50.000000000 % chance of winning the big prize."
## [1] "For  13983816  tickets,  You have a 100.000000000 % chance of winning the big prize."
```

We see that probability increases as the number of tickets increase, but the only way a player stands a 100% chance of winning is if they buy all the 13983816 tickets in the lottery. Which is almost impossible, if not impossible.

Less Winning Numbers — Function

In most 6/49 lotteries there are smaller prizes if a player's ticket matches 3,4 or 5 of the 6 numbers drawn. Users might therefore be interested in knowing the probability of having three, four, or five winning numbers. Inside the app, the user inputs 6 different numbers ranging from 1 - 49 & an integer between 3 and 5 that represents the number of winning numbers expected. What's more important is the number between 3-5. We'll write a function to help us calculate that probability.

```
prob_less6 <- function(n) {
    n_combinations_ticket <- combination(6,n)
    n_combinations_remaining <- combination(43,6-n)
    successful_outcomes <- n_combinations_ticket * n_combinations_remaining
    n_combinations_total <- combination(49,6)

    prob <- (successful_outcomes/n_combinations_total)*100
    pretty_prob <- sprintf("%1.9f", prob)

  message <- paste("You have a ", pretty_prob, "% chance of winning the big prize.", sep =
"")
  return(message)
}
```

Trying any of the winning numbers

```
winning_nums <- c(3,4,5)
for (n in winning_nums) {
  print(paste("For ", n, " tickets, ",  prob_less6(n), sep = ""))
}
```

```
## [1] "For 3 tickets, You have a 1.765040387% chance of winning the big prize."
## [1] "For 4 tickets, You have a 0.096861972% chance of winning the big prize."
## [1] "For 5 tickets, You have a 0.001844990% chance of winning the big prize."
```

Conclusion

We managed to write four functions to help us meet our objectives that were as follows:

- one_ticket_probability() — calculates the probability of winning the big prize with a single ticket
- check_historical_occurrence() — checks whether a certain combination has occurred in the Canada lottery data set
- multi_ticket_prob() — calculates the probability for any number of tickets between 1 and 13,983,816
- prob_less6() — calculates the probability of having three, four or five winning numbers

Possible features for a second version of the app include: - Improve the probability_less6() function to show the probabilities for having two winning numbers as well. - Making the outputs even easier to understand by adding fun analogies (for example, we can find probabilities for strange events and compare with the chances of winning in lottery; for instance, we can output something along the lines “You are 100 times more likely to be the victim of a shark attack than winning the lottery”). - Combine the one_ticket_probability() and check_historical_occurrence() to output information on probability and historical occurrence at the same time.