Probability of winning jeopardy

Bena

2024-02-21

Background

Jeopardy is a popular TV show in the US where participants answer trivia to win money. Participants are given a set of categories to choose from and a set of questions that increase in difficulty. As the questions get more difficult, the participant can earn more money for answering correctly. Let's say we're also interested in breaking the record. In this project, we'll work with a dataset of Jeopardy questions to figure out some patterns in the questions that could help you win.

It's a subset of 20,000 rows from a much larger dataset of Jeopardy questions.

Concepts applied in resolving this include: probability hypothesis testing

Data familiarization

Setting up

```
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()
```

Importing data

```
jeopardy <- read_csv("jeopardy.csv")
```

```
## Rows: 19999 Columns: 7
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (6): Air Date, Round, Category, Value, Question, Answer
## dbl (1): Show Number
##
## 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.
```

Data exploration

```
View(jeopardy)
glimpse(jeopardy)
```

```
## Rows: 19,999
## Columns: 7
## $ `Show Number` <dbl> 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 46…
## $ `Air Date`    <chr> "12/31/2004", "12/31/2004", "12/31/2004", "12/31/2004", …
## $ Round         <chr> "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeo…
## $ Category      <chr> "HISTORY", "ESPN's TOP 10 ALL-TIME ATHLETES", "EVERYBODY…
## $ Value         <chr> "$200", "$200", "$200", "$200", "$200", "$200", "$400", …
## $ Question      <chr> "For the last 8 years of his life, Galileo was under hou…
## $ Answer        <chr> "Copernicus", "Jim Thorpe", "Arizona", "McDonald's", "Jo…
```

Cleaning column names

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
jeopardy <- jeopardy %>%
               janitor::clean_names()
```

```
glimpse(jeopardy)
```

```
## Rows: 19,999
## Columns: 7
## $ show_number <dbl> 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680…
## $ air_date    <chr> "12/31/2004", "12/31/2004", "12/31/2004", "12/31/2004", "1…
## $ round       <chr> "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeopa…
## $ category    <chr> "HISTORY", "ESPN's TOP 10 ALL-TIME ATHLETES", "EVERYBODY T…
## $ value       <chr> "$200", "$200", "$200", "$200", "$200", "$200", "$400", "$…
## $ question    <chr> "For the last 8 years of his life, Galileo was under house…
## $ answer      <chr> "Copernicus", "Jim Thorpe", "Arizona", "McDonald's", "John…
```

```
sapply(jeopardy, typeof)
```

```
## show_number    air_date       round    category       value    question
##    "double" "character" "character" "character" "character" "character"
##      answer
## "character"
```

Fixing Data Types

```
unique(jeopardy$value)
```

```
##  [1] "$200"    "$400"    "$600"    "$800"    "$2,000"  "$1,000"  "$1,200"
##  [8] "$1,600"  "$3,200"  "None"    "$5,000"  "$100"    "$300"    "$500"
## [15] "$1,500"  "$4,800"  "$1,800"  "$1,100"  "$2,200"  "$3,400"  "$3,000"
## [22] "$4,000"  "$6,800"  "$1,900"  "$3,100"  "$700"    "$1,400"  "$2,800"
## [29] "$8,000"  "$6,000"  "$2,400"  "$12,000" "$3,800"  "$2,500"  "$6,200"
## [36] "$10,000" "$7,000"  "$1,492"  "$7,400"  "$1,300"  "$7,200"  "$2,600"
## [43] "$3,300"  "$5,400"  "$4,500"  "$2,100"  "$900"    "$3,600"  "$2,127"
## [50] "$367"    "$4,400"  "$3,500"  "$2,900"  "$3,900"  "$4,100"  "$4,600"
## [57] "$10,800" "$2,300"  "$5,600"  "$1,111"  "$8,200"  "$5,800"  "$750"
## [64] "$7,500"  "$1,700"  "$9,000"  "$6,100"  "$1,020"  "$4,700"  "$2,021"
## [71] "$5,200"  "$3,389"
```

The value column has character data type because some entries are “None” & the $ sign

```
jeopardy2 <- jeopardy %>%
               filter(value != "None") %>%
               mutate(value = str_replace_all(value, "[$,]",""),
                      value = as.numeric(value))
```

Normalizing Text

We want to clean the texts to ensure that we lowercase all the words and any remove punctuation. We’ll do this for the category, question & answer columns.

```
jeopardy2 <- jeopardy2 %>%
  mutate(category = str_to_lower(category),
         category = str_replace_all(category, "[^A-Za-z0-9 ],",""),
         category = gsub("[[:punct:]]", "", category),
         question = str_to_lower(question),
         question = str_replace_all(question, "[^[:alnum:]],",""),
         question = gsub("[[:punct:]]", "", question),
         answer = str_to_lower(answer),
         answer = str_replace_all(answer, "[^A-Za-z0-9. ],",""),
         answer = gsub("[[:punct:]]", "", answer)
         )
```

Making Dates More Accessible

We’ll separate this column into year, month and day columns to make filtering easier in the future. We also need them to be numeric

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
jeopardy2$air_date <- as.Date(jeopardy2$air_date, format = "%m/%d/%Y")
jeopardy2$year <- as.numeric(format(as.Date(jeopardy2$air_date, format = "%m/%d/%Y"),"%Y"))
jeopardy2$month <- as.numeric(format(as.Date(jeopardy2$air_date, format = "%m/%d/%Y"),"%m"))
jeopardy2$day <- as.numeric(format(as.Date(jeopardy2$air_date, format = "%m/%d/%Y"),"%d"))
```

Alternatively

```
jeopardy2 <- jeopardy2 %>%
               mutate(
                      day2 = day(air_date),
                      weekday = as.character(wday(air_date, label=TRUE)),
                      month2 = month(air_date),
                      year2 = year(air_date)
                      )
```

Re-ordering columns

```
jeopardy2 <- jeopardy2[,c("show_number","air_date","day","month","year","weekday","round","ca
tegory","value","question","answer")]
```

```
glimpse(jeopardy2)
```

```
## Rows: 19,663
## Columns: 11
## $ show_number <dbl> 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680, 4680…
## $ air_date    <date> 2004-12-31, 2004-12-31, 2004-12-31, 2004-12-31, 2004-12-3…
## $ day         <dbl> 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31…
## $ month       <dbl> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12…
## $ year        <dbl> 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004, 2004…
## $ weekday     <chr> "Fri", "Fri", "Fri", "Fri", "Fri", "Fri", "Fri", "Fri", "F…
## $ round       <chr> "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeopardy!", "Jeopa…
## $ category    <chr> "history", "espns top 10 alltime athletes", "everybody tal…
## $ value       <dbl> 200, 200, 200, 200, 200, 200, 400, 400, 400, 400, 400, 400…
## $ question    <chr> "for the last 8 years of his life galileo was under house …
## $ answer      <chr> "copernicus", "jim thorpe", "arizona", "mcdonalds", "john …
```

Focusing On Particular Subject Areas

Many people seem to think that science and history facts are the most common categories to appear in Jeopardy episodes. Others feel that Shakespeare questions gets an awful lot of attention from Jeopardy. With

the chi-squared test, we can actually test these hypotheses! Let's assess if science, history and Shakespeare have a higher prevalence in the data set.

```
n_categories <- jeopardy2 %>%
                 count(category) %>%
                 summarise(no_of_categories = n())

print(n_categories)
```

```
## # A tibble: 1 × 1
##   no_of_categories
##              <int>
## 1             3368
```

There are around 3368 unique categories in the Jeopardy data set after doing all of our cleaning. If we suppose that no category stood out, the probability of picking a random category would be the same no matter what category you picked.

```
n_questions <- nrow(jeopardy2)
p_category_expected <- 1/3368
p_not_category_expected <- 3367/3368
p_expected <- c(p_category_expected, p_not_category_expected)
```

We'll conduct a hypothesis test to see if the 3 are more likely to appear than other categories. > H0: Our null hypothesis states that science, history and Shakespeare are the most prevalent categories in Jeopardy > H1: The alternative hypothesis states that science, history and Shakespeare are not the most prevalent categories in Jeopardy.

First, we'll count how many times the word "science" appears in the category column.

```
categories <- pull(jeopardy2, category)
n_science_categories <- 0

for (c in categories) {
  if ("science" %in% c) {
    n_science_categories = n_science_categories + 1
  }
}
```

```
science_obs <- c(n_science_categories, n_questions - n_science_categories)
p_expected = c(1/3368, 3367/3368)
chisq.test(science_obs, p = p_expected)    #the function is used to conduct the hypothesis te
st
```

```
##
##  Chi-squared test for given probabilities
##
## data:  science_obs
## X-squared = 145.71, df = 1, p-value < 2.2e-16
```

The p-value is below 0.05 thus we reject the null hypothesis & conclude that science doesn't have a higher prevalence than other topics in the Jeopardy data.

```
n_history_categories <- 0

for (c in categories) {
  if ("history" %in% c) {
    n_history_categories = n_history_categories + 1
  }
}
```

```
history_obs <- c(n_history_categories, n_questions - n_history_categories)
p_expected = c(1/3368, 3367/3368)
chisq.test(history_obs, p = p_expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  history_obs
## X-squared = 199.96, df = 1, p-value < 2.2e-16
```

The p-value is below 0.05 thus we reject the null hypothesis & conclude that history doesn't have a higher prevalence than other topics in the Jeopardy data.

```
n_shakespear_categories <- 0

for (c in categories) {
  if ("shakespear" %in% c) {
    n_shakespear_categories = n_shakespear_categories + 1
  }
}
```

```
shakespear_obs <- c(n_shakespear_categories, n_questions - n_shakespear_categories)
p_expected = c(1/3368, 3367/3368)
chisq.test(shakespear_obs, p = p_expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  shakespear_obs
## X-squared = 5.8399, df = 1, p-value = 0.01567
```

The p-value is below 0.05 thus we reject the null hypothesis & conclude that shakespear doesn't have a higher prevalence than other topics in the Jeopardy data.

Unique Terms In Questions

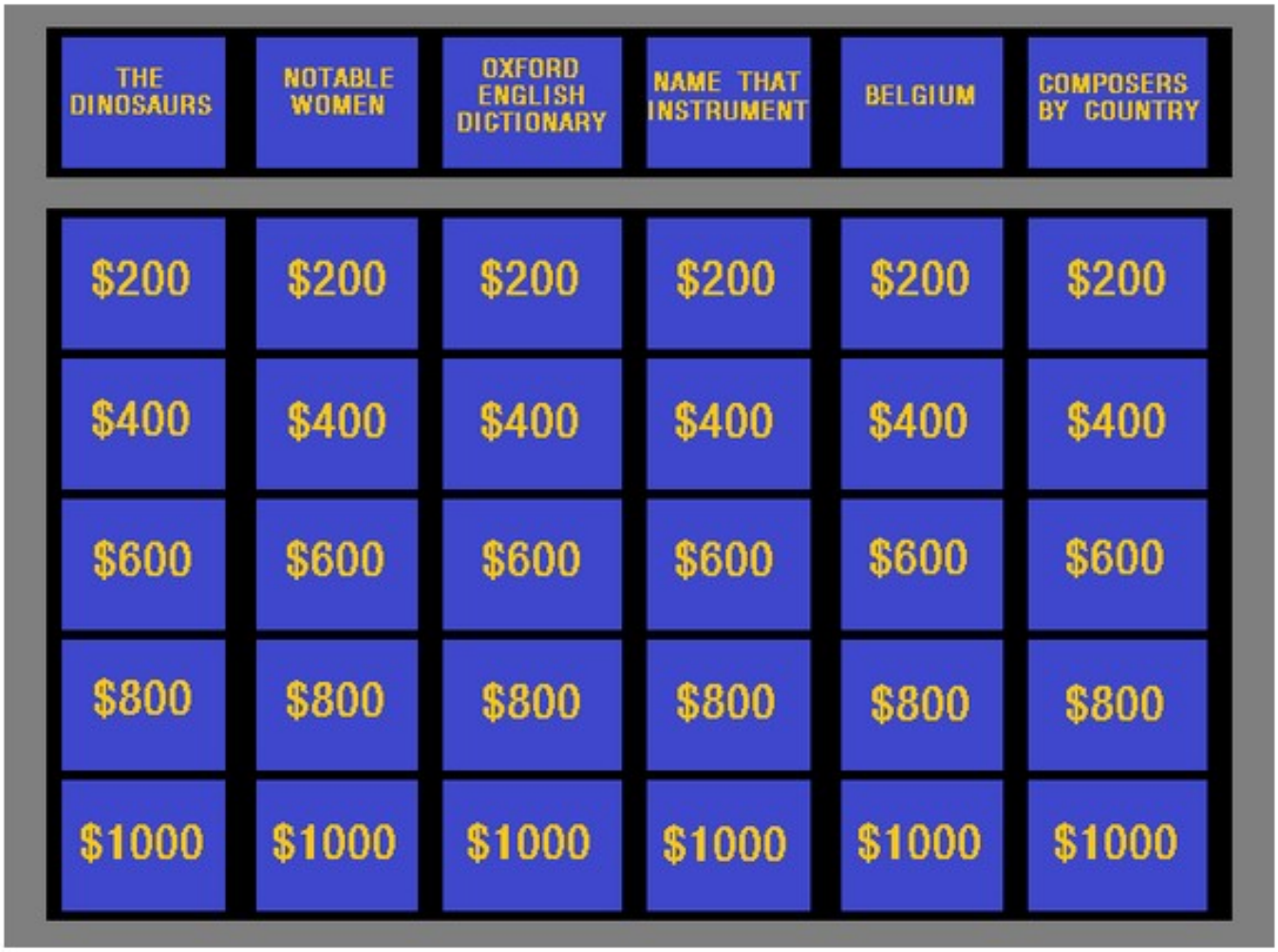
We'd like to investigate how often new questions are repeats of older ones.

```
questions <- pull(jeopardy2, question)
terms_used <- 0

for (q in questions) {
  # Split the sentence into distinct words
  split_sentence = str_split(q, " ")[[1]]

  # Check if each word is longer than 6 and if it's currently in terms_used
  for (term in split_sentence) {
    if (!term %in% terms_used & nchar(term) >= 6) {
      terms_used = c(terms_used, term)
    }
  }
}
```

Terms In Low and High Value Questions

We're more interested to study terms that have high values associated with it rather than low values. This optimization will help us earn more money when you're on Jeopardy while reducing the number of questions we have to study. We'll define low and high values as follows: >Low value: Any row where value is less than 800. >High value: Any row where value is greater or equal than 800.

Below is an image of what the question board looks like at the start of every round



For each category, we can see that for every 2 high value questions, there are 3 low value questions. If the number of high and low value questions is appreciably different from the 2:3 ratio, we would have reason to believe that a term would be more prevalent in either the low or high value questions. We can use the chi-squared test to test the null hypothesis that each term is not distributed more to either high or low value questions.

```
# Going only through the first 20 terms for shortness
# But you can remove the indexing to perform this code on all the terms
values = pull(jeopardy, value)
value_count_data = NULL

for (term in terms_used[1:20]) {
  n_high_value = 0
  n_low_value = 0

  for (i in 1:length(questions)) {
    # Split the sentence into a new vector
    split_sentence = str_split(questions[i], " ")[[1]]

    # Detect if the term is in the question and its value status
    if (term %in% split_sentence & values[i] >= 800) {
      n_high_value = n_high_value + 1
    } else if (term %in% split_sentence & values[i] < 800) {
      n_low_value = n_low_value + 1
    }
  }

  # Testing if the counts for high and low value questions deviates from what we expect
  test = chisq.test(c(n_high_value, n_low_value), p = c(2/5, 3/5))
  new_row = c(term, n_high_value, n_low_value, test$p.value)

  # Append this new row to our
  value_count_data = rbind(value_count_data, new_row)

}
```

```
## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect

## Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
## incorrect
```

```
# Take the value count data and put it in a better format
tidy_value_count_data <- as_tibble(value_count_data)
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
## `.name_repair` is omitted as of tibble 2.0.0.
## ℹ Using compatibility `.name_repair`.
```

```
colnames(tidy_value_count_data) = c("term", "n_high", "n_low", "p_value")

head(tidy_value_count_data)
```

```
## # A tibble: 6 × 4
##   term      n_high n_low p_value
##   <chr>     <chr>  <chr> <chr>
## 1 0         0      3     0.157299207050284
## 2 galileo   0      5     0.0678891548618288
## 3 arrest    0      5     0.0678891548618288
## 4 espousing 0      1     0.414216178242525
## 5 theory    0      25    4.45570906040561e-05
## 6 olympian  0      2     0.248213078989924
```

We can see from the output that some of the values are less than 5.
Recall that the chi-squared test is prone to errors when the counts in each of the cells are less than 5.
We may need to discard these terms and only look at terms where both counts are greater than 5.