Winning Jeopardy

Emmanuel Messori

27/09/2021

The context

In June 2019, contestant James Holzhauer ended a 32-game winning streak, just barely missing the record for highest winnings. James Holzhauer dedicated hours of effort to optimizing what he did during a game to maximize how much money he earned. To achieve what he did, James had to learn and master the vast amount of trivia that Jeopardy can throw at the contestants.

Project objective

In this project, we will exploit a data set containing 20000 rows of,information about Jeopardy questions.Let's say like James, we want to prepare to become Jeopardy champions. We will have to familiarize with a vast amount of topics. Is there a way to find which topics happen more frequently and prioritize them in our studies?

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
jeopardy <- read_csv("https://query.data.world/s/7mo5lkiqhcfjjuxrfqn4ryluv4impg",
    name_repair = ~str_to_lower(.))
```

```
## Rows: 19999 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): round, category, value, question, answer
## dbl  (1): show number
## date (1): air date
```

```
##
## 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.
```

```
head(jeopardy)
```

```
## # A tibble: 6 x 7
##   `show number` `air date` round     category     value question          answer
##           <dbl> <date>     <chr>     <chr>        <chr> <chr>             <chr>
## 1          4680 2004-12-31 Jeopardy! HISTORY      $200  "For the last 8 ~ Coper~
## 2          4680 2004-12-31 Jeopardy! ESPN's TOP ~ $200  "No. 2: 1912 Oly~ Jim T~
## 3          4680 2004-12-31 Jeopardy! EVERYBODY T~ $200  "The city of Yum~ Arizo~
## 4          4680 2004-12-31 Jeopardy! THE COMPANY~ $200  "In 1963, live o~ McDon~
## 5          4680 2004-12-31 Jeopardy! EPITAPHS & ~ $200  "Signer of the D~ John ~
## 6          4680 2004-12-31 Jeopardy! 3-LETTER WO~ $200  "In the title of~ the a~
```

The `value` column contains the prize value of each question in dollars. We will convert it to numeric and exclude the NA values.

```
jeopardy %>%
    mutate(value = parse_number(value)) %>%
    filter(!is.na(value)) -> jeopardy
```

Text normalization

We want ensure that we lowercase all of the words and any remove punctuation. We remove punctuation because it ensures that the text stays as purely letters. Before normalization, the terms "Don't" and "don't" are considered to be different words, and we don't want this. For this step, we will normalize the `question`, `answer`, and `category` columns. Also a lot of question contain html anchor tags.

```
jeopardy %>%
    filter(str_detect(question, "<.*?>")) %>%
    select(question) %>%
    head()
```

```
## # A tibble: 6 x 1
##   question
##   <chr>
## 1 "<a href=\"http://www.j-archive.com/media/2004-12-31_DJ_23.mp3\">Beyond ovoid~
## 2 "The shorter glass seen <a href=\"http://www.j-archive.com/media/2004-12-31_D~
## 3 "<a href=\"http://www.j-archive.com/media/2004-12-31_DJ_26.mp3\">Ripped from ~
## 4 "<a href=\"http://www.j-archive.com/media/2004-12-31_DJ_25.mp3\">Somewhere be~
## 5 "<a href=\"http://www.j-archive.com/media/2004-12-31_DJ_24.mp3\">\"500 Hats\"~
## 6 "(<a href=\"http://www.j-archive.com/media/2010-07-06_J_22.wmv\">Tate: I'm Ta~
```

We'll clean the mentioned columns, removing html tags, converting to lowercase and finally removing all punctuation.

```
jeopardy %>%
    mutate(question = str_replace_all(question, "<.*?>", "") %>%
        map_chr(~str_glue(.x))) %>%
```

```
    mutate(across(c(question, answer, category), ~str_remove_all(str_to_lower(.x),
        "[:punctuation:]"))) -> jeop_clean
# we can safely remove the original data set
rm(jeopardy)
```

Asking questions

We are now in a place where we can properly ask questions from the data and perform meaningful hypothesis tests on it. Given the near infinite amount of questions that can be asked in Jeopardy, you wonder if any particular subject area has increased relevance in the data set. 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.

We will now conduct a chi-square test to see if science, history and Shakespeare have higher prevalence than other categories in the data set. 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, we would expect that the probability of picking a random category would be the same no matter what category you picked.

```
length(unique(jeop_clean$category))
```

```
## [1] 3369
```

Before doing the tests, let's establish some important parameters, including the expected probability if every category was chosen randomly. We have 3368 unique categories, so if we suppose that no category stood out, we would expect that the probability of picking a random category would be the same no matter what category you picked, this would also mean that the probability of not picking a particular category would be 3367/3368.

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

Now we are ready to build our hypothesis test which is the same for the three categories. We will count the number of questions which contains the category,the complementary count which doesn't present the category, and use them with the expected probabilities as inputs to the `chisq.test` function. Our null hypothesis H_0 is that the category has not higher prevalence than a random topic, while the alternative hypothesis H_1 states that the category is more likely to be chosen than random topic.

```
n_science <- filter(jeop_clean, category == "science") %>%
    nrow()/n_questions
n_not_science <- n_questions - n_science
n_obs <- c(n_science, n_not_science)
chisq.test(n_obs, p = p_expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  n_obs
## X-squared = 5.8364, df = 1, p-value = 0.0157
```

```
n_history <- filter(jeop_clean, category == "history") %>%
    nrow()/n_questions
n_not_history <- n_questions - n_history
n_obs <- c(n_history, n_not_history)
chisq.test(n_obs, p = p_expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  n_obs
## X-squared = 5.8358, df = 1, p-value = 0.0157
```

```
n_sh <- filter(jeop_clean, category == "shakespeare") %>%
    nrow()/n_questions
n_not_sh <- n_questions - n_history
n_obs <- c(n_sh, n_not_sh)
chisq.test(n_obs, p = p_expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  n_obs
## X-squared = 5.8379, df = 1, p-value = 0.01568
```

In all three cases, with a similar p-value ~ 0.016 we reject H_0 and we can affirm that there enough evidence that **these three categories are more likely to be chosen** than a random category with probability 0.00%.

Word occurrences in jeopardy questions

Let's say you want to investigate how often new questions are repeats of older ones. We're only working with about 10% of the full Jeopardy question data set, but we can at least start investigating this question. Let' use the `tm` library to create a Corpus object and then a term matrix of the words used in the questions:

```
library(tm)
```

```
## Le chargement a nécessité le package : NLP
```

```
##
## Attachement du package : 'NLP'
```

```
## L'objet suivant est masqué depuis 'package:ggplot2':
##
##     annotate
```

```
corp <- VCorpus(VectorSource(jeop_clean$question))
corp[[1]]$content
```

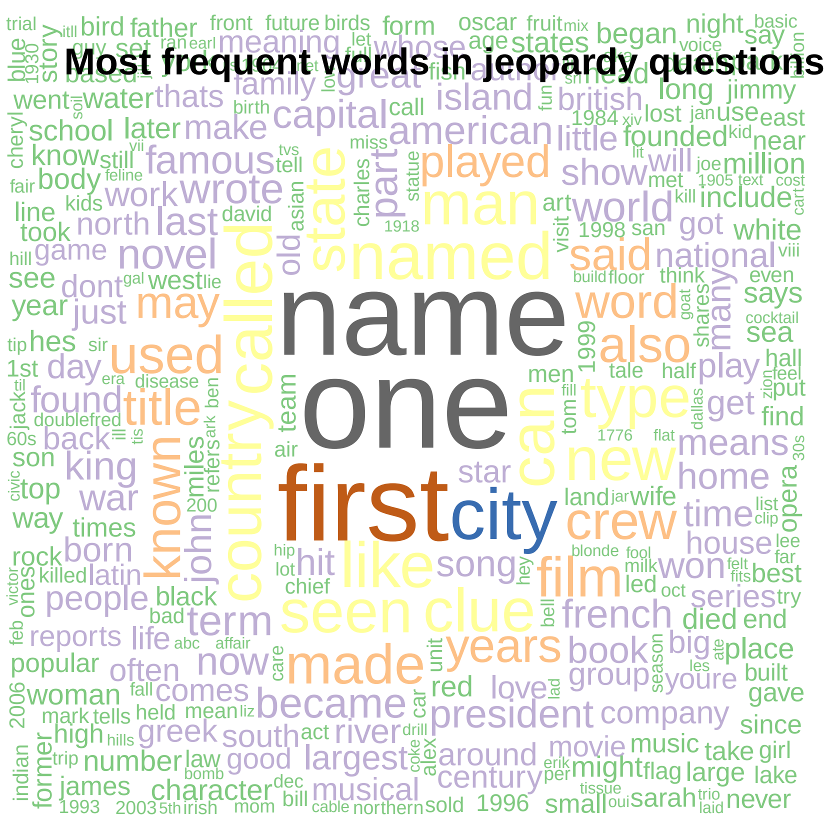
```
## [1] "for the last 8 years of his life galileo was under house arrest for espousing this mans theory"
```

Let's also remove "stop words", commonly used words like articles, pronouns and short terms, which are note really useful to understand the question theme:

```
corp <- tm_map(corp, removeWords, stopwords("english"))
```

Building a Document Term Matrix with minimum word length == 6 and then plotting a wordcloud:

```
# restricting to words with at least two occurrences and min 3 char.
quest_tdm <- DocumentTermMatrix(corp, control = list(wordLengths = c(3, Inf), bounds = list(global = c(
    Inf))))
quest_m <- as.matrix(quest_tdm)
wc <- colSums(quest_m)
wordcounts <- data.frame(count = wc)
pal <- RColorBrewer::brewer.pal(8, "Accent")
wordcloud::wordcloud(freq = wordcounts$count, words = rownames(wordcounts), random.order = FALSE,
    colors = pal)
title(main = "Most frequent words in jeopardy questions")
```




```
# top 100 word count
head(sort(wc, decreasing = TRUE), 100)
```

```
##      one     name    first     city   called    named     like      can
##     1118     1000      917      571      512      498      486      473
##  country     seen     clue      man      new     type    state     film
```

```
##       465       465       446       445       440       436       432       406
##      made      used      crew      also     title     known      word     years
##       388       380       367       355       353       343       313       311
##    played       may      said     wrote    became      term     world     novel
##       296       291       288       279       275       268       257       256
##  american   capital      part president    french    famous      king      last
##       252       251       251       245       243       240       235       235
##       now      book       won      show      home    island      john       war
##       235       223       223       222       215       211       209       209
##      song      time     great     means     found       hit       get      just
##       208       208       206       205       204       200       197       190
##      born     river      make      play  national       day    people      many
##       187       185       184       184       179       178       178       177
##    little   largest      work       old      life     group      star    around
##       176       174       174       173       172       170       168       166
##      back     house     comes      dont     south   british    author     whose
##       165       165       163       161       161       160       159       159
##   meaning     north    family     greek       got   century      love       big
##       157       157       155       155       152       151       151       148
##   musical   company     often    series     thats      will      game   reports
##       148       147       147       147       147       147       145       145
##     latin     movie      good     youre
##       144       144       141       140
```

Low & High Value terms

Let's say you only want to study terms that have high values associated with it rather than low values. This optimization will help you earn more money when you're on Jeopardy while reducing the number of questions you have to study. To do this, we need to count how many high value and low value questions are associated with each term. For our exercise, 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.

If you are not familiar with Jeopardy, below is an image of what the question board looks like at the start of every round:

Question board

For each category, we can see that under this definition that for every 2 high value questions, there are 3 low value questions. Once we count the number of low and high value questions that appear for each term, we can use this information to our advantage. 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.

```
jeop_clean %>%
    mutate(is_high = if_else(value >= 800, "high", "low")) -> jeop_clean

word_values <- data.frame(row.names = names(wc))
for (w in rownames(word_values)) {
    counts <- filter(jeop_clean, str_detect(question, paste("\\b", w, "\\b", sep = ""))) %>%
        count(is_high)
    meanvalue <- filter(jeop_clean, str_detect(question, paste("\\b", w, "\\b", sep = ""))) %>%
        summarise(mean(value)) %>%
```

```
        pull()
    word_values[w, "nhigh"] = sum(word_values[w, "nhigh"], counts$n[1], na.rm = TRUE)
    word_values[w, "nlow"] = sum(word_values[w, "nlow"], counts$n[2], na.rm = TRUE)
    word_values[w, "meanvalue"] = round(meanvalue, 2)
}
```

Now that we have obtained a data frame with the number of occurrences of each word in high and low value questions, we can calculate for each the chi square statistic, to see if there is a statistically significant difference with an equal distribution between the two (50/50). We have to bear in mind that for a significant chi square statistic the expected value for each group has to be at least 5:

```
word_values_f <- word_values %>%
    filter((nlow + nhigh) * 3/5 >= 5 & (nlow + nhigh) * 2/5 >= 5)
word_values_f$p_value <- word_values_f %>%
    pmap(~chisq.test(x = c(.x, .y), p = c(2/5, 3/5))) %>%
    map_dbl("p.value")
```

It is then possible to adjust the p-values to take into account this multiple testing and the problem of increasing the overall alpha risk. Here it is done with the adjustment method of Holm:

```
word_values_f$p_value_adj <- p.adjust(word_values_f$p_value, method = "holm")
word_values_f$p_value <- round(word_values_f$p_value, 4)
word_values_f$p_value_adj <- round(word_values_f$p_value_adj, 4)
```

We can now check the words with an adjusted p-value smaller than the significance level of 0.05:

```
word_values_f %>%
    filter(p_value_adj < 0.05) %>%
    arrange(desc(nhigh/nlow))
```

```
##           nhigh nlow meanvalue p_value p_value_adj
## particles    19    2   1333.86       0      0.0053
## monitor      46    8   1355.56       0      0.0000
## 1951         22    4   1126.92       0      0.0077
## shows        68   18   1248.84       0      0.0000
## sarah        77   32   1051.38       0      0.0000
## kelly        49   22   1094.37       0      0.0014
## crew        243  120   1062.53       0      0.0000
## clue        272  137   1049.14       0      0.0000
## african      45   23   1102.94       0      0.0236
## reports      91   54   1064.83       0      0.0001
## italian      61   38    946.46       0      0.0254
## jimmy        65   41   1029.25       0      0.0167
## latin        87   56   1037.06       0      0.0008
## french      144   97   1011.20       0      0.0000
## greek        88   65    976.47       0      0.0219
## island      111   92    902.46       0      0.0440
## word        162  139    896.35       0      0.0022
## known       175  164    872.86       0      0.0281
## named       251  237    864.14       0      0.0006
## name        489  482    828.84       0      0.0000
```

If these words were equally distributed between high and low value questions, they would have had respectively 2/5 and 3/5 probability of falling into these classes. We can reject this hypothesis and affirm that these words are probably associated with the prevalence of the high value category.