stsci4100

Tony Oh (do256), William Rhee (wr86)

2025-03-25

App Ideas: - Filter recommendation system: basic, but we could try to implement AI. - Rating estimator: basic regression analysis. - Success estimator: I was thinking we could take rating to be a high heuristic value while number of votes and help determine success. If there's many votes, the more reliable the ratings can be, so we can deem more extreme ratings with many votes as more correct in terms of success. Then, we can estimate the level of success a movie would have based on their input of title, is adult, start year, end year, endtime in minutes, and genres. Basically a regression with heuristics for success.

Possible questions: 1. Does the number of votes affect the ratings? - Graph: x = numVotes, y = rating - See if normal distribution 2. Does having a writer affect the ratings? 3. Does having certain actors have an affect on ratings? Or is this just an effect of hiring good actors for good movies? 4. How does runtime affect ratings? 5. Which genres get the highest ratings?

TODO: - Combine datasets with unique tconst: basics, crew, ratings (excluding principals). - Figure out what principals dataset is.

In this project, we will be using four datasets from IMDb:

1. **title.basics.tsv**: A dataset providing all the basic details about a broad range of films from 1892 to 2026.
2. **title.crew.tsv**: A dataset providing the ID's specifically of the director and writers for the movies from Dataset 1.
3. **title.principals.tsv**: A dataset providing details about other crew (editors, producers, etc.) for the movies from Dataset 1.
4. **title.ratings.tsv**: A dataset that contains the rating data from IMDb's users for the movies from Dataset 1,

[1] DATA EXPLORATION:

There are three points in this section:

A. Importing the Datasets.
B. A Quick Glance.
C. Specific Breakdowns.
D. Combining the Datasets.

[A] Importing the datasets.

```r
library(readr)

basics <- read.delim("title.basics.tsv", sep = "\t", header = TRUE, na.strings = "\\N", nrows = 200000)
crew <- read.delim("title.crew.tsv", sep = "\t", header = TRUE, na.strings = "\\N", nrows = 200000)
principals <- read.delim("title.principals.tsv", sep = "\t", header = TRUE, na.strings = "\\N", nrows =
ratings <- read.delim("title.ratings.tsv", sep = "\t", header = TRUE, na.strings = "\\N", nrows = 20000

#head(basics)
#head(crew)
#head(principals)
#head(ratings)

my.datasets <- list(basics, crew, principals, ratings)
attr(my.datasets, "names") <- c("basics", "crew", "principals", "ratings")
```

[B] A Quick Glance.

```r
# Let's view our datasets.
#View(basics)
#View(crew)
#View(principals)
#View(ratings)

# How many columns are in our datasets?
for (i in names(my.datasets)){
  column.names <- paste(colnames(my.datasets[[i]]), collapse = ", ")
  print(paste("In the", i, "dataset, there are", ncol(my.datasets[[i]]), "columns."))
  print(paste("Its columns are:", column.names))
  cat("\n")
}
```

```
## [1] "In the basics dataset, there are 9 columns."
## [1] "Its columns are: tconst, titleType, primaryTitle, originalTitle, isAdult, startYear, endYear, r
##
## [1] "In the crew dataset, there are 3 columns."
## [1] "Its columns are: tconst, directors, writers"
##
## [1] "In the principals dataset, there are 6 columns."
## [1] "Its columns are: tconst, ordering, nconst, category, job, characters"
##
## [1] "In the ratings dataset, there are 3 columns."
## [1] "Its columns are: tconst, averageRating, numVotes"
```

```r
# How many rows are in our datasets?
for (i in 1:4) print(paste("There are", nrow(my.datasets[[i]]), "rows in the", names(my.datasets)[i], "
```

```
## [1] "There are 200000 rows in the basics dataset."
## [1] "There are 200000 rows in the crew dataset."
## [1] "There are 200000 rows in the principals dataset."
## [1] "There are 200000 rows in the ratings dataset."
```

[C] Specific Breakdowns.

[C1] The basics dataset

The **basics** dataset:

- **tconst**: The unique id for a specific film.
- **titleType**: The kind of film — it can be a short, movie, tvShort, and more.
- **primaryTitle**: The final title of the film.
- **originalTitle**: The title of the film (before it was changed).
- **isAdult**: The indicator variable — 1 if it's an adult film, 0 otherwise.
- **startYear**: The film's release year.
- **endYear**: The film's ending year, IF the film was a TV show (NA otherwise).
- **runtimeMinutes**: The film's runtime (in minutes).
- **genres**: The film's list of genres.

Here is a quick exploration of the **basics** dataset's columns:

```r
anyNA(basics$tconst)                                  # Investigating the column: tconst.
length(unique(basics$tconst)) == nrow(basics)

anyNA(basics$titleType)                               # Investigating the column: titleType.
unique(basics$titleType)

anyNA(basics$primaryTitle)                            # Investigating the column: primaryTitle.
length(unique(basics$primaryTitle))

anyNA(basics$originalTitle)                           # Investigating the column: originalTitle.
length(unique(basics$originalTitle))

anyNA(basics$isAdult)                                 # Investigating the column: isAdult.
mean(basics$isAdult)

anyNA(basics$startYear)                               # Investigating the column: startYear.
min(basics$startYear, na.rm = TRUE)
max(basics$startYear, na.rm = TRUE)

anyNA(basics$endYear)                                 # Investigating the column: endYear.
min(basics$endYear, na.rm = TRUE)
max(basics$endYear, na.rm = TRUE)

anyNA(basics$runtimeMinutes)                          # Investigating the column: runtimeMinutes.
min(basics$runtimeMinutes, na.rm = TRUE)
max(basics$runtimeMinutes, na.rm = TRUE)
mean(basics$runtimeMinutes, na.rm = TRUE)
```

Here are the takeaways from the given code:

- **tconst**: There are no missing values. There are exactly 200,000 id's (i.e. individual films), one for every row.
- **titleType**: There are no missing values. There are 10 unique kinds of films in this dataset, ranging from shorts to video games.
- **primaryTitle**: There are no missing values. There are 179081 primaryTitles in the dataset (< 200,000). This should be looked into.
- **originalTitle**: There are no missing values. There are 182034 originalTitles in the dataset (< 200,000 and > 179081). This should be looked into.
- **isAdult**: There are no missing values. About 7% of all the films in the dataset are adult films. (OMG)
- **startYear**: There are missing values! Ignoring missing values, the earliest release date was 1892, while the most recent release date was 2025.
- **endYear**: There are missing values! Ignoring missing values, the earliest end date was 1945, while the most recent end date was 2026.
- **runtimeMinutes**: There are missing values! Ignoring missing values, the shortest film is a minute long, while the longest film is 1,620 minutes (27 hours) long. The average film length is around 75 minutes (1.25 hours).

[C2] The crew dataset

The **crew** dataset:

- **tconst**: The unique id for a specific film.
- **directors**: The film's list of directors.
- **writers**: The film's list of writers.

Here is a quick exploration of the **crew** dataset's columns:

```r
anyNA(crew$tconst)                                  # Investigating the column: tconst.
length(unique(crew$tconst)) == nrow(crew)

anyNA(crew$directors)                               # Investigating the column: directors.
anyNA(crew$writers)                                 # Investigating the column: writers.
```

Here are the takeaways from the given code:

- **tconst**: There are no missing values. There are exactly 200,000 id's (i.e. individual films), one for every row.
- **directors**: There are some missing values. This means not every movie in our data will have a specified director.
- **writers**: There are some missing values. This means not every movie in our data will have a specified writer.

[C3] The principals dataset

The **principals** dataset:

- **tconst**: The unique id for a specific film.
- **ordering**: The order in credits, where 1 is the most important and 35 is least important.
- **nconst**: The unique id for a specific person such as an actor, director, and more.
- **category**: The role of a specific person such as an actor, director, and more.
- **job**: The job title such as a producer, editor, and more.
- **characters**: The character that a specific person has played as.

Here is a quick exploration of the **principals** dataset's columns:

```r
anyNA(principals$tconst)                                      # Investigating the column: tconst.
length(unique(principals$tconst)) == nrow(principals)
```

```r
anyNA(principals$ordering)                                # Investigating the column: ordering.
unique(principals$ordering)

anyNA(principals$nconst)                                  # Investigating the column: nconst.

anyNA(principals$category)                                # Investigating the column: category.
unique(principals$category)

anyNA(principals$job)                                     # Investigating the column: job.
# unique(principals$job)                                  # Do not run. Leave this commented out.

anyNA(principals$characters)                              # Investigating the column: characters.
# unique(principals$characters)                           # Do not run. Leave this commented out.
```

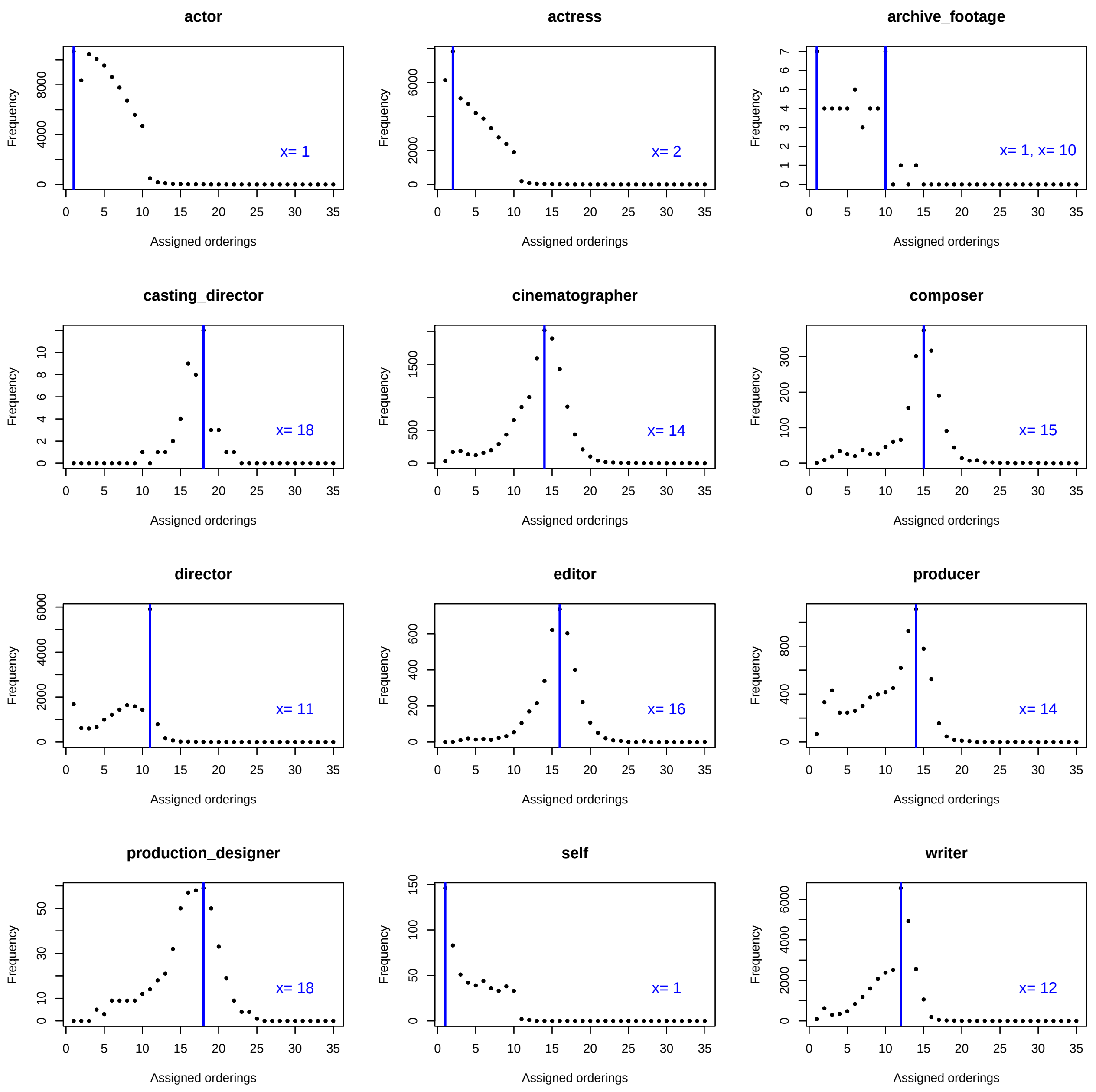
Here are the takeaways from the given code:

- **tconst**: There are no missing values. However, the number of unique film ID's does not equal the number of rows — this is because upon close inspection, we can see that there are duplicated ID's across several rows. This is different from the other three datasets, where there is one unique film ID for each row.
- **ordering**: There are no missing values. The ordering from most to least important ranges from 1 to 35.
- **nconst**: There are no missing values.
- **category**: There are no missing values. There are 12 unique categories in the dataset, from director to casting director.
- **job**: There are missing values — this is because of the duplicate values from the first column. There are too many unique values within this column for R to run through without crashing (several times).
- **characters**: There are missing values — this is because of the duplicate values from the first column. There are too many unique values within this column for R to run through without crashing (several times).

A side-note

We know that there are 12 unique categories within the **category** column. We also know that the ordering ranges from 1 to 35, where 1 is most important and 35 is least important.

So as a SIDE-NOTE, let's make a DataFrame (and plots) that count how many times each job **category** appears in a specific **ordering**.



```
##                category ordering
## 1                 actor        1
## 2               actress        2
## 3       archive_footage    1, 10
## 4      casting_director       18
## 5       cinematographer       14
## 6              composer       15
## 7              director       11
## 8                editor       16
## 9              producer       14
## 10 production_designer       18
## 11                 self        1
## 12               writer       12
```

As we can see, the code matches what we'd expect if we eyeball what the code is doing to **my.table**.

- In addition, we can see that the ordering for the category **archive_footage** has an unclear assignment, where it can be assigned to an ordering of either 1 or 10.
- This is because the category was equally assigned to those orderings an equal number of times.

[C4] The ratings dataset

The **ratings** dataset:

- **tconst**: The unique id for a specific film. There are exactly 200,000 id's (i.e. individual films), one for every row.
- **averageRating**: The average rating, which was given to reviewers on a 1-10 scale.
- **numVotes**: The number of reviewers.

Here is a quick exploration of the **ratings** dataset's columns:

```r
anyNA(ratings$tconst)                                        # Investigating the column: tconst.
length(unique(ratings$tconst)) == nrow(ratings)

anyNA(ratings$averageRating)                                 # Investigating the column: averageRating.
unique(ratings$averageRating)
min(ratings$averageRating)
max(ratings$averageRating)
mean(ratings$averageRating)

anyNA(ratings$numVotes)                                      # Investigating the column: numVotes.
min(ratings$numVotes)
length(which(ratings$numVotes == min(ratings$numVotes)))
max(ratings$numVotes)
length(which(ratings$numVotes == max(ratings$numVotes)))
which(ratings$numVotes == max(ratings$numVotes))
```

Here are the takeaways from the given code:

- **tconst**: There are no missing values. There are exactly 200,000 id's (i.e. individual films), one for every row.
- **averageRating**: There are no missing values.
- **numVotes**: There are no missing values. The worst review ever given was 1/10, while the best were 10/10. As a fun fact, the average rating across movies from the 1890's all the way to present day is roughly 6/10. Of all 200000 films, 513 films have received the lowest number of reviews (only 5 reviews). Of all 200000 films, only one film (i.e. the film in the 84776th row) has the largest number of reviews (3026798 reviews).

D. Combining the Datasets.

Excluding the principals dataset, let's combine the three datasets that share the same first column **tconst** (i.e. the same unique ID values):

1. The basics dataset.
2. The crew dataset.
3. The ratings dataset.

```r
# COMBINING DATASETS:
# [1] Combining all three datasets into one final dataset.
final.dataset <- merge(basics, crew, by = "tconst")
final.dataset <- merge(final.dataset, ratings, by = "tconst")
#View(final.dataset)
```

```
# [2] Checking the dimensions (rows, col).
dim(basics)                                         # This is correct - it should be (200000 rows, 9 col
```

```
## [1] 200000      9
```

```
dim(crew)                                           # This is correct - it should be (200000 rows, 3 col
```

```
## [1] 200000      3
```

```
dim(ratings)                                        # This is correct - it should be (200000 rows, 3 col
```

```
## [1] 200000      3
```

```
dim(final.dataset)                                  # This is WRONG - it should be (200000 rows, 13 colu
```

```
## [1] 139602     13
```

As we can see:

1. We are able to merge the basics dataset with the crew dataset based on the first column. This is because all of their values within the first column match.
2. However, our final dataset does **not** contain 200,000 rows. It has the right number of columns, but this is still an issue we need to fix.

Let's run diagnostics to see why this is happening.

```
# DIAGNOSTICS: [PART 1]
# [1] Comparing the first column of each dataset.
all(basics$tconst == crew$tconst)                                       # Check: are the first colum
all(basics$tconst == ratings$tconst)                                    # Check: are the first colum

# Here, we found out that the datasets basics and crew have the exact same first column - both their ro
# Here, we found out that the dataset ratings does not match - either by a difference in their rows (i.

# [2] Comparing the first column's rows (order).
mismatched.rows <- which(basics$tconst != ratings$tconst)               # A vector of all the rows t
c(min(mismatched.rows), max(mismatched.rows), length(mismatched.rows))  # A quick exploration of mis
all(mismatched.rows == 178:200000)                                      # It seems that mismatched.r

all(basics$tconst[1:177] == ratings$tconst[1:177])                      # Double check: does the fir
all(basics$tconst[178:200000] == ratings$tconst[178:200000])            # Double check: does the fir
```

Here, we found out that rows 178 to 200,000 of the first column between the basic and ratings dataset do not match. But here's the weird thing:

- The merge() function doesn't seem to care whether the ROWS of our datasets match.
- Why? Because the number of mismatched rows is 199,823 rows. If the merge() function cared about this number, then the final dataset will only have 200,000 - 199,823 = 177 rows.
- In reality, our final dataset ended up with 139602 rows. That is only 200,000 - 139,602 = 60,398 rows short (not 177).
- Therefore, our diagnosis is **not over** yet.

In summary: the fact that only the first 177 rows in the first column match between the basics and ratings dataset is **NOT** the reason why our final merged dataset didn't end up with 200,000 rows.

```
# DIAGNOSTICS: [PART 2]
# [1] Comparing the first column's values (order does not matter).
length(unique(basics$tconst))                                        # Check: Do we have 200,000 uni
length(unique(crew$tconst))                                          # Check: Do we have 200,000 uni
length(unique(ratings$tconst))                                       # Check: Do we have 200,000 uni
```

```
# Here, we found out that all three datasets have exactly 200,000 unique ID values. That's good.

all(unique(basics$tconst) == unique(crew$tconst))                    # Check: Are the first column's
all(unique(basics$tconst) == unique(ratings$tconst))                 # Check: Are the first column's

# Here, we found out that the datasets basics and crew have the exact same first column - both their ro
# Here, we found out that the datasets basics and ratings do NOT have the exact same first column - one
```

Here's the big takeaway from this diagnosis:

- All three datasets have exactly 200,000 unique ID values — however, the **ratings** dataset does NOT cover ALL the same films.
- Instead, the basics and crew datasets talk about the exact same 200,000 films.
- Instead, the ratings dataset is a loner that brings up movies that aren't shared with those two datasets. (Most of them are in common, but not all.)

So according to our final merged dataset's dimensions, we can say that out of all 200,000 films listed in each dataset, only 139602 of them are shared between all three datasets.

- The remaining 60398 films are films that the ratings dataset doesn't have with the basics and crew dataset.
- The ratings dataset talks about 60398 other films instead.

So what does this mean moving forward?

- It means that before we analyze any data, we'll have to make the decision to only analyze the films all three datasets share (i.e. only 139602 out of 200,000 films).
- If we choose to bluntly include ALL movies in all three datasets, we'll just end up with a final merged DataFrame that contains NA's for the films that don't have reviews.
- If considering the reviews of a movie are important, then we should stick with just analyzing the 139,602 out of 200,000 films (because we won't have reviews for the rest).
- If considering as many films as possible is important, then we should include all 200,000 films (even if they obviously won't have reviews attached to them).

As a final check, here is the last diagnosis to confirm these findings about the differences between the basics/crew dataset and the ratings dataset.

```
difference.1 <- setdiff(basics$tconst, crew$tconst)                # A vector that lists all the elemen
length(difference.1)                                               # This should be zero because both b

difference.2 <- setdiff(basics$tconst, ratings$tconst)             # A vector that lists all the elemen
length(difference.2)                                               # This should be 60398 because basic

difference.3 <- setdiff(ratings$tconst, basics$tconst)             # A vector that lists all the elemen
length(difference.3)                                               # This should be 60398 because ratin

shared.amount1 <-intersect(basics$tconst, crew$tconst)             # A vector that lists all the elemen
length(shared.amount1)                                             # This should be 200,000. (It is!)

shared.amount2 <-intersect(basics$tconst, ratings$tconst)          # A vector that lists all the elemen
length(shared.amount2)                                             # This should be 139602. (It is!)
```