DATA 607, Project Two: Data Transformation - Movie Ratings

Kevin Kirby

2024-09-29

Overview

This is one of three distinct files created for project two of the Fall 2024 edition of DATA 607. This project asked me to pick three datasets posted by other students in the class and use it in a tidying and review exercise. The below is based on a movie ratings dataset posted by AL. I added the file to my CUNY GCP bucket and set it for public download.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5

## -- 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
```

```
library(readr)

movie_data_al <- read_lines("https://storage.googleapis.com/data_science_masters_files/2024_fall/data_6
```

Tidying the data

The data was a bit of a mess at the start, with the biggest issue being that some rows of data sat in one cell together and others were split across more than one.

I first did the data cleanup as I would do it and then I did it as asked for in the discussion board prompt.

```
header <- strsplit(movie_data_al[1], ";")[[1]]
aspiring_start <- movie_data_al[-1]
```

```r
split_data <- lapply(aspiring_start, function(line) {
  strsplit(line, ";")[[1]]
})

max_length <- max(sapply(split_data, length))
split_data_uniform <- lapply(split_data, function(row) {
  length(row) <- max_length
  return(row)
})

tidy_movie_data <- do.call(rbind, split_data_uniform) %>%
  as.data.frame(stringsAsFactors = FALSE)

colnames(tidy_movie_data) <- header %>%
  tolower() %>%
  gsub("([a-z])([A-Z])", "\\1_\\2", .) %>%
  gsub(" ", "_", .) %>%
  tolower()

tidy_movie_data <- tidy_movie_data %>%
  rename(
    release_date = releasedate,
    avg_rating = avgrating
  )
```

Discussion board tidy The prompt said: “I would also replace the 1s and 0s system of denoting a films category, by just simply typing out the categories that apply to each film - ie if a film is”action and adventure” just simply add those descriptors and create a system for working this into our analysis. This would greatly reduce the number of columns and the size of the file, thus making it much easier to work with. Five columns (name, release date, category, avg rating, num of views) is much easier to work with than the current format.From there, this would make subsetting the data and drawing meaningful insights on popularity, popularity by category, highest rating, average rating etc much easier.”

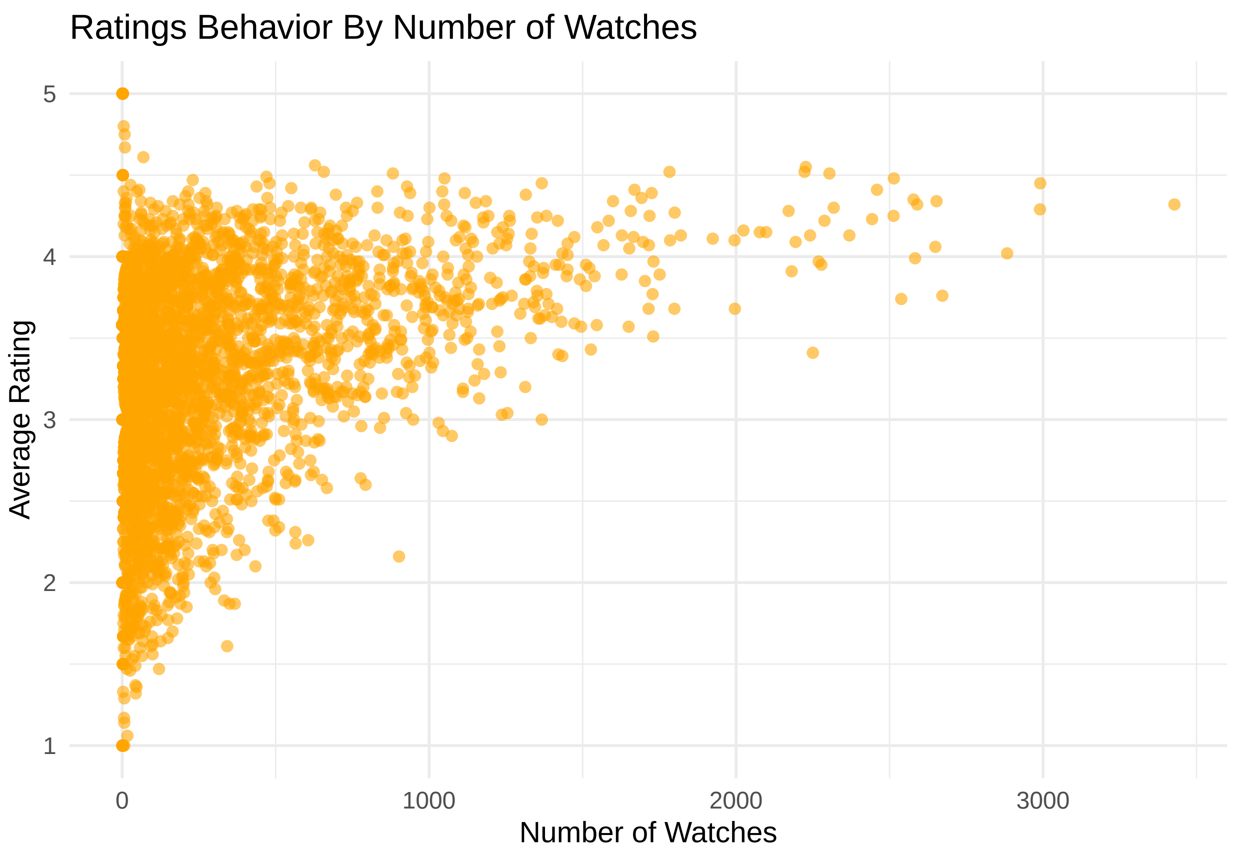
The below takes the category names and makes them a comma separated list in a single cell. Unfortunately, this did make further analysis more difficult as I would have had to parse the comma separated values to try and separate things out for analysis. I was able to make, without further deviation from the prompt, a chart showing average ratings as a function of the number of watches.

```r
tidy_movie_al <- tidy_movie_data %>%
  rowwise() %>%
  mutate(
    category = paste(
      c(
        if (action == 1) "action",
        if (adventure == 1) "adventure",
        if (children == 1) "children",
        if (comedy == 1) "comedy",
        if (crime == 1) "crime",
        if (documentary == 1) "documentary",
        if (drama == 1) "drama",
        if (fantasy == 1) "fantasy",
        if (noir == 1) "noir",
        if (horror == 1) "horror",
        if (musical == 1) "musical",
        if (mystery == 1) "mystery",
        if (romance == 1) "romance",
        if (scifi == 1) "scifi",
        if (thriller == 1) "thriller",
        if (war == 1) "war",
        if (western == 1) "western"
      ), collapse = ", "
    )
  ) %>%
  ungroup()

tidy_movie_al <- tidy_movie_al %>%
  select(name, release_date, category, avg_rating, watches)
```

```r
tidy_movie_al$watches <- as.integer(tidy_movie_al$watches)
tidy_movie_al$avg_rating <- as.numeric(tidy_movie_al$avg_rating)

ggplot(tidy_movie_al, aes(x = watches, y = avg_rating)) +
  geom_point(alpha = 0.6, color = "orange") +
  ggtitle("Ratings Behavior By Number of Watches") +
  xlab("Number of Watches") +
  ylab("Average Rating") +
  theme_minimal() +
  scale_x_continuous(guide = guide_axis(check.overlap = TRUE)) +
  scale_y_continuous(guide = guide_axis(check.overlap = TRUE))
```



Conclusion

I understand the desire to want to have as few columns as possible and, if done right, it's a good headspace to be in. For this specific dataset, placing the genres in a single cell created a hierarchical mismatch that made follow on analysis a bit more challenging. To do a by-genre analysis, you need a distinct value for a movie and a genre, even if a movie belongs to multiple genres.