Team name: Movie Pruners
Zachary Chang
Hugo Tessier
Jameson Toper
Ellis Wright

Phase 1

For the first phase of the project we chose to use Python in conjunction with Pandas and Numpy to process and join the data. A presentation of the steps involved in finer granularity than presented here is contained in the Jupyter notebook along with this submission (you can view the notebook without downloading Jupyter by opening the *phase_1_import.html* file in your browser or view the raw python code in *phase_1_import.py*). The input data was processed in three distinct steps: import, filter, & merge. The import step included enumerating over each of the tsv files and importing them into a Pandas Dataframe using the “read_csv” function with a tab as the delimiter. If the file contained a column that had data separated by commas then for each row the data was split on the “,” character into a Python list. The Dataframe was then “exploded” on that column (using the corresponding method name in Pandas). This created a new row for each value in the list with all other columns identical.

Filtering was performed to reduce the total number of rows in the dataset before merging to minimize join time. The *title.akas.tsv* file contained information about local titles for other countries, this was filtered to only include US titles. This reduced the table by 11,728,136 entries. The *title.basics.tsv* file contained information on media other than movies, this was filtered to include only the “movie” and “tvMovie” categories as we are only interested in movies. This reduced the table by 11,979,416 entries. The *name.basics.tsv* file contained information about staff who worked on each movie, including non-actors. This table was filtered to only include “actor” and “actress” in the *primaryProfession* column. This reduced the table by 18,251,612 entries.

Merging the tables first required that the column names be updated so they were consistent for joining. The *title.akas* table had the *titleId* column renamed to

tconst and the *name.basics* table had the *knownForTitles* column renamed to *tconst* as well. Each table had its index reset to the “tconst” column for ease of use. In order to generate the combined table an inner join was performed iteratively over the tables in the following order: *name.basics*, *title.basics*, *title.akas*, and *title.ratings.* The tables were joined on the *tconst* column. The newly created table was then written to a file in tsv format.

Phase 2

The dataset obtained in Phase 1 is smaller than the one expected as we merged the *name.basics* table -which contains the information on the actors- with an inner join. Thus, we only kept the movies which had at least one famous actor playing in it.

For the second phase of the project, we begin to analyse the resulting dataset by creating the data dictionary (Figure 1).

Variable	Unique Entries	Total Entries	Data Type	Data Classification	Number of Empty Values	Range
tconst	142461	9465447	String	Nominal	0	
nconst	1312824	9465447	String	Nominal	0	
primaryName	1221954	9465447	String	Nominal	0	
birthYear	189	2550732	float64	Scalar	6914715	12.0 - 2018.0
deathYear	119	826698	float64	Scalar	8638749	1569.0 - 2021.0
primaryProfession	39	9465447	String	Nominal	0	
titleType	2	9465447	String	Nominal	0	
primaryTitle	125298	9465447	String	Nominal	0	
originalTitle	130378	9465447	String	Nominal	0	
isAdult	2	9465447	int64	Scalar	0	0 - 1
startYear	117	9465308	float64	Scalar	139	1906.0 - 2022.0
endYear	0	0	float64	Scalar	9465447	
runtimeMinutes	337	9288532	float64	Scalar	176915	4.0 - 5220.0
genres	28	9448891	String	Nominal	16556	
ordering	79	9465447	int64	Ordinal	0	1 - 108
title	149274	9465447	String	Nominal	0	
region	1	9465447	String	Nominal	0	
language	5	168462	String	Nominal	9296985	
types	13	7962363	String	Nominal	1503084	
attributes	99	1105031	String	Nominal	8360416	
isOriginalTitle	2	9465447	int64	Scalar	0	0 - 1
averageRating	91	9465447	float64	Scalar	0	1.0 - 10.0
numVotes	16410	9465447	int64	Scalar	0	5 - 2455388

Figure 1: data dictionary of the resulting dataset of phase 1

We automatically created the dictionary via a Python script *create_data_dict.py* using pandas. We set na values as *'\\N'* for the import because it was the way pandas exported missing values after phase 1. The dictionary summaries information on the 23 variables:

- **Unique entries:** unique values of the variable - computed with *pandas.DataFrame.nunique*.
- **Total entries:** number of values of the variable - computed with *pandas.DataFrame.count* (excluding missing values).
- **Data Type:** Type of the variable - computed with *pandas.DataFrame.dtype*.
- **Data Classification:** category of the variable (scalar, nominal, or ordinal value) - set manually.
- **Number of empty values:** number of empty values of the variable - computed with *pandas.DataFrame.isna*.
- **Range:** minimum and maximum values of the variable - computed with *min* and *max* functions.

We study the values obtained in the dictionary. “birthYear”, “deathYear”, and “startYear” are considered as float variables and need to be set as integers later on as they correspond to years. “IsAdult” and “isOriginalTitle” are integer variables between 0 and 1, those correspond to boolean variables and could be then considered as categorical. We observe that the “endYear” variable is entirely empty, “birthYear”, “deathYear”, “language”, and “attributes” contain a lot of empty values (>80,000,000). We can also notice that “titleType” and “region” have only 2 and 1 unique entries respectively - it is due to the filtering made on *US movies* during the first phase. Concerning the “primaryProfession” variable, considering all the other professions of actresses and actors lead to 39 different unique entries.

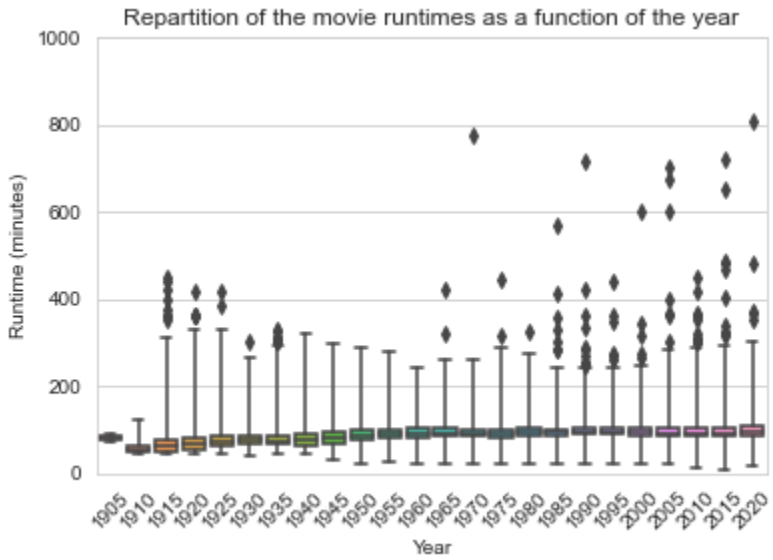
Finally, we begin to observe values that seem to be outliers such as the minimum of “BirthYear” which is *12* or the minimum “runtimeMinutes” of the movies which is *4 minutes*. Those will be studied more granularly in the next phase.

Phase 3

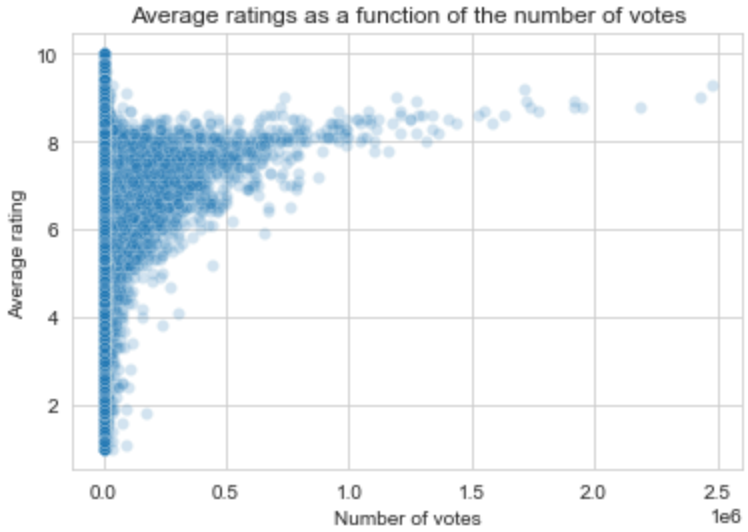
Variable	Unique Entries	Total Entries	Data Type	Data Classification	Number of Empty Values	Range	Median	Mean	Mode	Min	Max	Outliers
tconst	142461	9465447	String	Nominal	0							
nconst	1312824	9465447	String	Nominal	0							
primaryName	1221954	9465447	String	Nominal	0							
birthYear	189	2550732	float64	Scalar	6914715	12.0 - 2018.0	1959.0	1952.567790344105	1970.0	12.0	2018.0	{12.0, 21.0, 1827.0, 1828.0, 1829.0, 1830.0, 1831.0, 1834.0, 1835.0, 1837.0, 1838.0, 1839.0, 1841.0, 1842.0, 1843.0, 1844.0, 1845.0, 1846.0, 1847.0, 1848.0, 1849.0, 1850.0, 1851.0, 1852.0, 1853.0, 1854.0, 1855.0, 1856.0, 1857.0, 1858.0, 1859.0, 1860.0, 1861.0, 1862.0, 1863.0, 1864.0, 1865.0, 1866.0}
deathYear	119	826698	float64	Scalar	8638749	1569.0 - 2021.0	1998.0	1992.987573454877	2020.0	1569.0	2021.0	{1920.0, 1921.0, 1922.0, 1923.0, 1924.0, 1925.0, 1926.0, 1569.0, 1886.0, 1890.0, 1895.0, 1900.0, 1904.0, 1905.0, 1908.0, 1910.0, 1912.0, 1913.0, 1914.0, 1915.0, 1916.0, 1917.0, 1918.0, 1919.0}
primaryProfession	39	9465447	String	Nominal	0							
titleType	2	9465447	String	Nominal	0							
primaryTitle	125298	9465447	String	Nominal	0							
originalTitle	130378	9465447	String	Nominal	0							
isAdult	2	9465447	int64	Scalar	0	0 - 1	0.0	0.0065564785265819999	0	0	1	
startYear	117	9465308	float64	Scalar	139	1906.0 - 2022.0	2008.0	2001.031816608609	2016.0	1906.0	2022.0	{1920.0, 1921.0, 1922.0, 1923.0, 1924.0, 1925.0, 1926.0, 1927.0, 1928.0, 1929.0, 1930.0, 1931.0, 1932.0, 1933.0, 1934.0, 1935.0, 1936.0, 1937.0, 1938.0, 1939.0, 1940.0, 19410, 1942.0, 1906.0, 1907.0, 1908.0, 1909.0, 1910.0, 1911.0, 1912.0, 1913.0, 1914.0, 1915.0, 1916.0, 1917.0, 1918.0, 1919.0}
endYear	0	0	float64	Scalar	9465447	nan - nan	nan	nan		nan	nan	set{}
runtimeMinutes	337	9288532	float64	Scalar	176915	4.0 - 5220.0	100.0	105.1536777824526	90.0	4.0	5220.0	{4.0, 6.0, 7.0, 100, 13.0, 15.0, 20.0, 209.0, 720.0, 566.0, 1234.0, 600.0, 5220.0, 653.0, 675.0, 700.0, 191.0, 192.0, 193.0, 194.0, 195.0, 196.0, 197.0, 198.0, 199.0, 2000, 201.0, 202.0, 203.0, 204.0, 205.0, 206.0, 207.0, 208.0, 715.0, 2100, 211.0, 212.0, 213.0, 214.0, 215.0, 216.0, 217.0, 218.0, 219.0, 220.0, 2210, 222.0, 223.0, 224.0, 225.0, 226.0, 227.0, 228.0, 229.0, 230.0, 231.0, 232.0, 233.0, 234.0, 235.0, 236.0, 237.0, 238.0, 239.0, 240.0, 241.0, 242.0, 243.0, 244.0, 245.0, 246.0, 247.0, 248.0, 250.0, 251.0, 252.0, 253.0, 2540, 255.0, 256.0, 257.0, 258.0, 259.0, 260.0, 261.0, 262.0, 263.0, 264.0, 265.0, 266.0, 267.0, 268.0, 269.0, 270.0, 271.0, 776.0, 273.0, 272.0, 275.0, 278.0, 279.0, 280.0, 281.0, 282.0, 283.0, 284.0, 285.0, 287.0, 288.0, 289.0, 290.0, 292.0, 293.0, 294.0, 295.0, 808.0, 298.0, 299.0, 300.0, 302.0, 303.0, 305.0, 306.0, 308.0, 310.0, 3150, 317.0, 319.0, 320.0, 321.0, 323.0, 324.0, 328.0, 330.0, 338.0, 345.0, 348.0, 350.0, 353.0, 354.0, 357.0, 360.0, 364.0, 366.0, 368.0, 369.0, 374.0, 382.0, 398.0, 400.0, 1428.0, 404.0, 410.0, 416.0, 417.0, 418.0, 420.0, 421.0, 422.0, 439.0, 440.0, 442.0, 4500, 467.0, 1500.0, 480.0, 485.0}
genres	28	9448891	String	Nominal	16556							
ordering	79	9465447	int64	Ordinal	0	1 - 108	7.0	12.270208263804129	1	1	108	{51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 75, 76, 85, 102, 104,
title	149274	9465447	String	Nominal	0							
region	1	9465447	String	Nominal	0							
language	5	168462	String	Nominal	9296985							
types	13	7962363	String	Nominal	1503084							
attributes	99	1105031	String	Nominal	8360416							
isOriginalTitle	2	9465447	int64	Scalar	0	0 - 1	0.0	4.722439415697959e-05	0	0	1	
averageRating	91	9465447	float64	Scalar	0	1.0 - 10.0	6.3	6.133128736550957	6.6	1.0	10.0	{1.1, 2.0, 2.1, 1.8, 1.2, 1.4, 1.9, 1.0, 1.6, 1.5, 1.7, 1.3}
numVotes	16410	9465447	int64	Scalar	0	5 - 2455388	3425.0	93004.13702934473	11	5	2455388	{904196, 1042959, 972822, 986649, 1934364, 1288223, 820779, 1191470, 1241138, 945204, 783931, 881723, 788542, 809535, 1273927, 879176, 792650, 1065049, 1705053, 1064032, 794210, 1199717, 939110, 2411126, 779898, 945802, 1237131, 1058967, 897699, 799916, 923822, 818874, 1026237, 1262787, 793802, 1259216, 816852, 1747669, 775896, 1902811, 1109215, 848630, 1328378, 940300, 980753, 1043218, 1090323, 981781, 1699106, 1573670, 897834, 883507, 924469, 931637, 980801, 869702, 1153866, 779089, 1356629, 2455388, 2163044, 1607526, 1066347, 1726320, 1898353, 1202035, 1089907, 983925, 951675, 1039749, 1102218, 1304978, 788384, 985505, 781225, 1541547, 784303, 955316, 1180087, 918455, 789954, 914884, 906195, 1166292, 774611, 1421779, 1327578, 1078239, 1031146,

First, we updated the data dictionary by adding information on the scalar variables such as descriptive statistics (mean, median, mode, min, max) and the outliers.

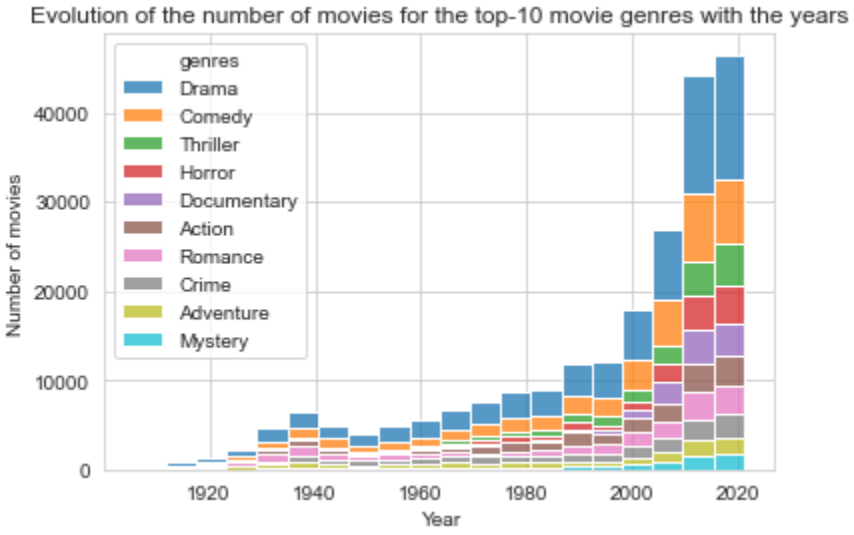
Then, for this phase, the main goal was to generate visual representations of the dataset. This is intended to highlight interesting aspects of the dataset, while also locating any outliers or dangerous areas in the data. The first set of charts were created to investigate the connection between two attributes of the data. The graphs for “Runtime versus Year” and “Rating versus # of Votes” show a strong correlation between their two respective variables. Movie lengths began fairly consistent, potentially due to some limitation of the time, and after 1965, movie durations have many more instances of long runtimes and outliers. And for votes, there is a correlation that people are more likely to vote if they are going to give a high rating.



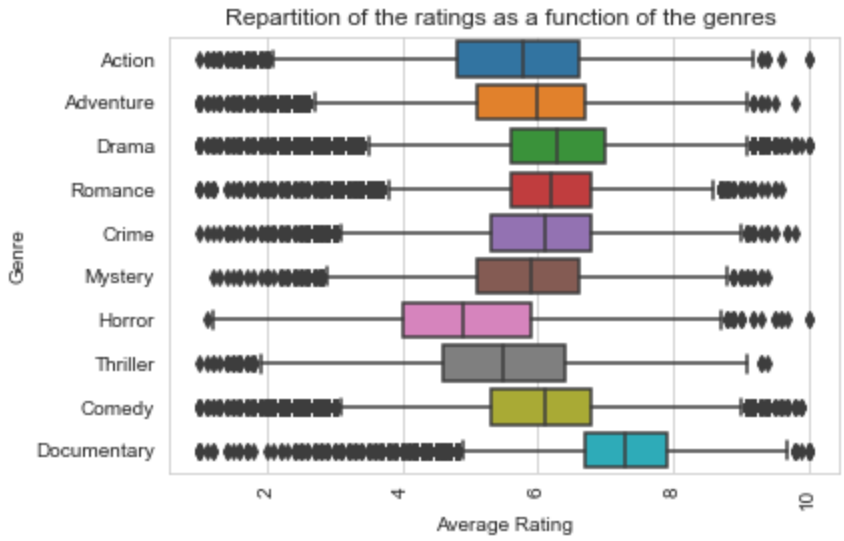
The box and whiskers plot was chosen for the “Rating versus Year” as it gives an indication of where the majority of ratings rank while also showing the outliers. If this was a bar chart, the information would be condensed into a single datapoint for each bar, and important insights about the data would be lost.



The same goes for the “Rating versus # of Votes”, the scatter plot works well with a large mass of data that have small variations. Looking for clusters of points will help pick out trends in the ratings or discover a recording error.

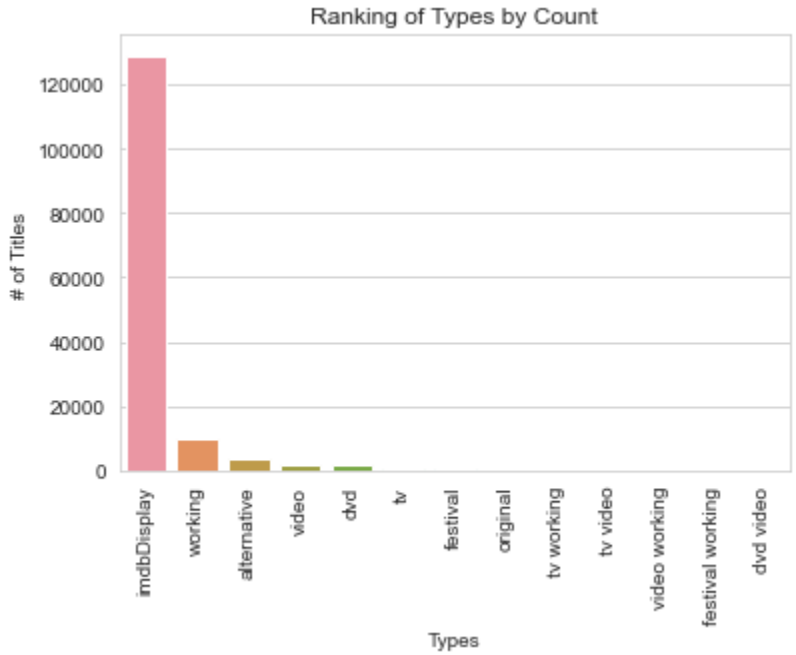


The two evolution graphs show the scale at which movies are released each successive year. The graph of genres points out that dramas have become the most popular genre of movie, but the graphs are mainly rising as a result of the sheer volume of movies released in recent years.



The chart that describes ratings based on the genre of movie also shows interesting details such as the horror movies having a consistent wide range of ratings, and documentaries tend to have higher ratings with some low outliers.

There are graphs for non-numerical as well, which is best suited to be represented by a bar graph with heights as the total counts of each unique value; a countplot. The graph which shows the ranking of types of movies only had 13 unique values to consider, so the number of occurrences are plotted to best describe the column.



Because these variables have no numerical value, the most informative way to chart them is by how often they appear in the dataset. But because there are a limited number of options, one downside is that getting numerical results from the charts is tough. The bars on the chart are barely even visible in some circumstances, so having a numerical chart breakdown would also be helpful.