Movie revenues are always an interesting topic to review. A while back movie director Brett Ratner claimed that Rotten Tomatoes was the “ruin of the movie business”. To validate the claim, we have constructed a database that includes 14762 scores from Rotten Tomatoes and IMDB audience scores for over 4000 movies released between 1914 and 2019. Nonetheless, many of the movies in the dataset were released between 1900 and 2020.

We found that the CSV file, contains all others equal, each percentage point a movie is rated “Fresh” Rotten Tomatoes correlated a with audience review. In addition, the revenues across the world were not added a correlation with the IMDB with their 100 point scale associated with the audience review we used a 1-10 point system on the CSV file.

We looked at the overall correlation between reviews and revenues but did not control for other factors that affect revenues, like the movie genre, rating, and other factors that affect revenues. We can validate that critics are likely not the key factor determining revenues but cannot pinpoint whether reviews have increasing outcomes.

The data used was collected from https://www.kaggle.com/orgesleka/imdbmovies/data#) and critics dataset (https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-

critics-datasets) sources, scraped rottentomatoes.com for its “tomatometer” scores. Rotten Tomatoe’s shows that the tomatometer “represents the percentage of qualified critic reviews that are positive for a given film or tv show.” IMDB data include score, genre, and rating and come from the “IMDB 5000 Movie Database” hosted on Kaggle.com and compiled by user Greg. The IMDB ratings are the average score, which can range from 1 to 10, submitted by users of IMDB.com. Movie revenues are available at “[The Numbers](http://www.the-numbers.com/movie/budgets/all).” Budget and release date were available on both the IMDB and Numbers datasets. Budget and revenue are not provided at this time.

We can make a table showing statistically significant and economically important results on the ratings coefficient with the use of this tools. Hypothetically we could show: “*The results suggest that a one percentage point increase in a movie’s tomatometer score is associated with a $1.12 million increase in worldwide revenues. Hence, with no other changes, a movie rated 60 percent “fresh” would earn about $11.2 million more than a movie rated 50 percent “fresh.*” However, we only used the audience rating. Comparably, all else equal, we would review for example*: “a movie rated 6.0 on IMDB would earn about $38 million more than a movie rated 5.0. Given that mean worldwide revenues are about $116 million”, or we can add other data for comparison as demographics, revenue, etc.”*

* Extract:

The data sources that we used for this project are the IMDB Movies dataset

(https://www.kaggle.com/orgesleka/imdbmovies/data#) and the Rotten tomatoes movie

and critics dataset(https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-

critics-datasets).

Both datasets were originally formatted as CSV files.

* Transform:

We are using two CSV files to compare movie ratings between IMDB and Rotten

Tomatoes.

The two datasets use different values for their rating systems. The rotten tomatoes

dataset uses a rating system from 1-100 while the IMDB dataset uses a rating

system of 1-10. We converted the IMDB rating system to a 1-100 value to be able to

compare with rotten tomatoes.

* Load:

We chose MongoDB for our final database to store the IMDB and Rotten tomatoes

datasets

The code:

import pandas as pd  
import json  
from pymongo import MongoClientdef CleanData(rawImdbDF, rawRottenDF):  
 imdbTrimDF = rawImdbDF[['title','year', 'imdbRating', 'ratingCount']]  
 rottenTrimDF = rawRottenDF[['movie\_title','in\_theaters\_date', 'audience\_rating','audience\_count']] imdbTrimDF.dropna(inplace=True)  
 rottenTrimDF.dropna(inplace=True) imdbTrimDF['title'] = imdbTrimDF['title'].str.split('(').str[0]  
 rottenTrimDF['in\_theaters\_date'] = rottenTrimDF['in\_theaters\_date'].str[0:4]  
 rottenTrimDF['in\_theaters\_date'] = rottenTrimDF['in\_theaters\_date'].astype(int)  
 rottenTrimDF['audience\_rating'] = rottenTrimDF['audience\_rating'].astype(float)  
 rottenTrimDF['audience\_rating'] = rottenTrimDF['audience\_rating'].div(10)  
 rottenTrimDF['audience\_count'] = rottenTrimDF['audience\_count'].astype(int)  
 return imdbTrimDF, rottenTrimDFdef LoadCsv():  
 #build file location string to read data  
 imdbFile = "Resources/Sources/imdb.csv"  
 rottenFile = "Resources/Sources/rotten\_tomatoes\_movies.csv" #load movie csv to data frames  
 imdbDF = pd.read\_csv(imdbFile, error\_bad\_lines=False)  
 rottenDF = pd.read\_csv(rottenFile, error\_bad\_lines=False)  
 return imdbDF, rottenDFdef MongoDBInit():  
 #Create Connection to Mongo DB and create the DB  
 client = MongoClient(port=27017)  
 db = client.MovieAnalysisDB  
 return dbdef MongoCollection(mongoDB):  
 imdbCollection = mongoDB['IMDB']  
 rottenCollection = mongoDB['Rotten']  
 return imdbCollection, rottenCollectiondef MongoInsert(imdbCol, rottenCol, cleanImdbDF, cleanRottenDF):  
 result = cleanImdbDF.to\_json()  
 imdbDict = json.loads(result)  
 imdbCol.insert\_one(imdbDict) result = cleanRottenDF.to\_json()  
 rottenDict = json.loads(result)  
 rottenCol.insert\_one(rottenDict)rawImdbDF, rawRottenDF = LoadCsv()cleanImdbDF, cleanRottenDF = CleanData(rawImdbDF, rawRottenDF)mongoDB = MongoDBInit()imdbCol, rottenCol = MongoCollection(mongoDB)MongoInsert(imdbCol, rottenCol, cleanImdbDF, cleanRottenDF)