Driving Miss *Data*

An Analysis of Best Picture Nominees (1927 - 2010)

The Team

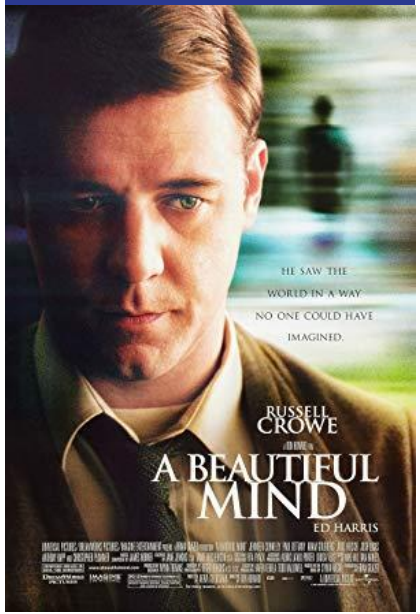
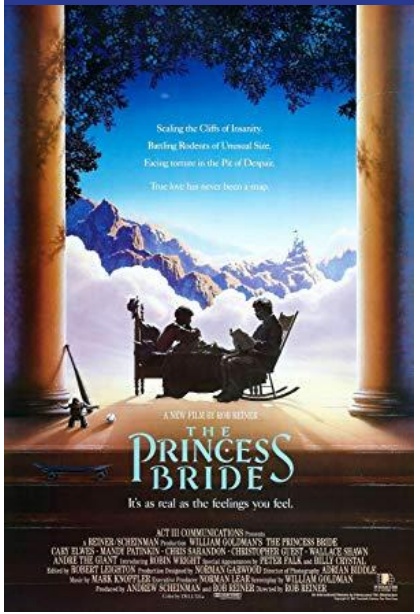
K. Staley Sharples	Kellen Park	Eric Litz	Bryce Davis
			

Image Source: http://www.omdbapi.com/

Core Challenge

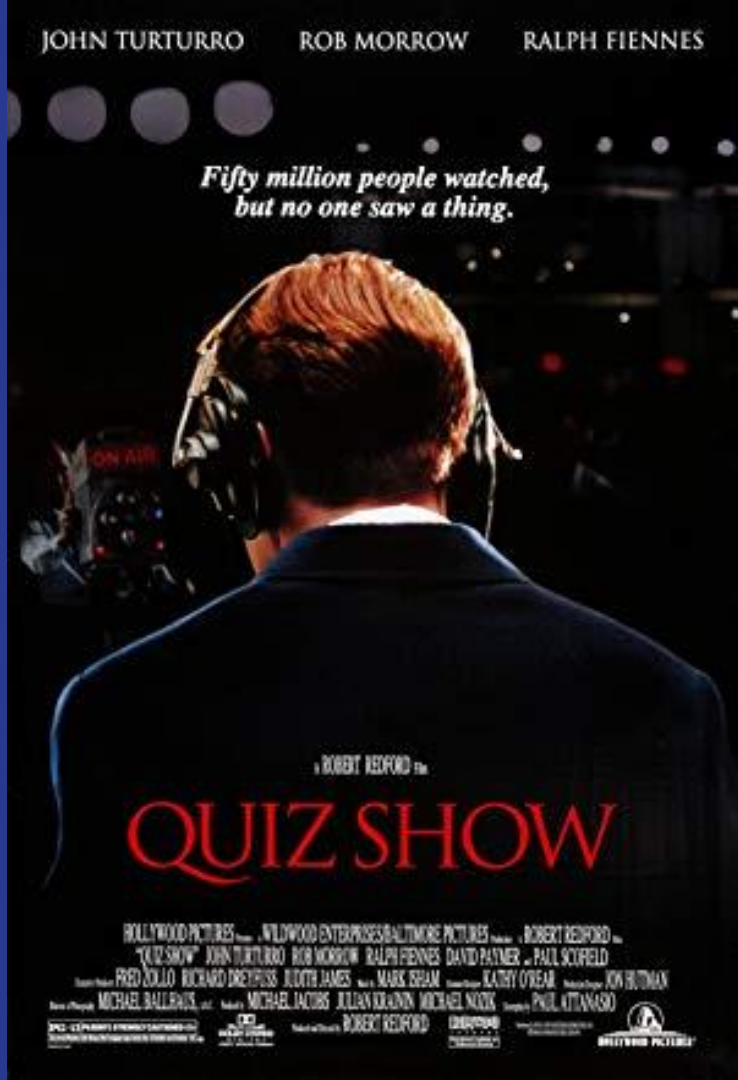
Core Challenge

What trends can be identified in the **‘Best Picture’ Oscar nomination category** over the span of over 80+ years’ worth of data?

Core Challenge Questions

1. What is the count of movie ratings (R, PG, G, etc.) by winning and losing ‘Best Picture’ nominees?
2. What is the count of movie rating (R, PG, G, etc.) by season?
3. Is there a correlation between IMDB rating and MetaCritic score?
4. Is there a trend in ‘Best Picture’ nominees by season?
5. Is there a trend in genre by winning and nominated movies?
6. What is the average IMDB rating by season?
7. Is there a significant difference between the average of IMDB Ratings for different seasons?

Our Approach



Our Approach

- Outlined our hypotheses:
  - If a movie is ‘Rated R’ and released during the Fall, then it is more likely to be a ‘Best Picture’ nominee.
  - If a movie is released during Fall or Winter it is more likely to have a higher average IMDB rating, due to the assumption that the majority of ‘Best Picture’ winners are released during these seasons.
  - If a movie is a drama or action movie it is more likely to be nominated
  - Null hypothesis for t test: there is no significant difference between the mean of fall nominees and winter nominees
- Observed data from 1 api and 2 csv exports (omdbapi.com/, aggdata.com/awards/oscar, and kaggle.com/rounakbanik/the-movies-dataset)

Our Approach

- Made sense of the csv containing movie award data
  - Had to set encoding to ‘latin-1’
- Initial file contained over 10,000 records, so decided to narrow our scope to just ‘Best Picture’ (~400 records)

```
# create a path to academy awards csv and read it into a pandas dataframe
awards_csv = "academy_awards_data_2.csv"
awards_df = pd.read_csv(awards_csv, usecols = ['Nominee', 'Year', 'Category', 'Won?'], encoding = 'latin-1')

# create another dataframe that only includes nominees in the Best Picture category
award_data = awards_df.loc[awards_df["Category"] == "Best Picture", :]

# create list of best picture nominees
best_picture_noms = award_data["Nominee"]

# print(best_picture_noms)
# print(best_picture_noms[77])
# award_data.head()
```

Our Approach

```
# print the corresponding number for each movie
movie_number = 1

# empty lists for holding movie data
box_office = []
genre = []
meta_score = []
imdb_rating = []
title = []
poster_url = []
rated = []
release_date = []
studio = []

best_picture_noms = award_data["Nominee"]
base_url = "http://www.omdbapi.com/?"
```

Our Approach

- Similar to our homework, wrote a for loop to pull data from each corresponding movie
- Was not sure of API limitations, so added in “time.sleep(1.01)” to slowly pull data from the API

```
# print statement as each movie is processed
print(f"Beginning Data Retrieval")
print(f"-----------------------------")

# loop through the movies in the best picture noms dataframe
for movie in best_picture_noms:

    params = {
    "apikey" : omdb_key,
    "t" : movie
    }

    # try statement for each potential movie
    try:
        omdb_data_raw = requests.get(base_url, params=params)
        omdb_data = omdb_data_raw.json()
        box_office.append(omdb_data["BoxOffice"])
        genre.append(omdb_data["Genre"])
        meta_score.append(omdb_data["Metascore"])
        imdb_rating.append(omdb_data["imdbRating"])
        title.append(omdb_data["Title"])
        poster_url.append(omdb_data["Poster"])
        rated.append(omdb_data["Rated"])
        release_date.append(omdb_data["Released"])
        studio.append(omdb_data["Production"])
        print_title = omdb_data["Title"]

        print(f"Processing Record {movie_number} | {print_title}")
        print(omdb_data_raw.url)

        # increase movie number by one each loop
        movie_number = movie_number + 1

#         to avoid 60 rpm api limit i'm waiting just over 1 second per loop
#         https://www.pythoncentral.io/pythons-time-sleep-pause-wait-sleep-stop-your-code/)
        time.sleep(1.01)

    # skip if no movie is found or if data is missing
    except:
        print("Data missing or movie not found. Skipping...")
    continue

print(f"-----------------------------")
print(f"Data Retrieval Complete")
print(f"-----------------------------")
```

Our Approach

```
# converting filtered api data into dataframe
filtered_omdb_data_df= pd.DataFrame ({
    "Title": title,
    "Genre": genre,
    "Meta_Score": meta_score,
    "imdb_Rating": imdb_rating,
    "Box_Office" : box_office,
    "Rated" : rated,
    "Studio" : studio,
    "Release_Date" : release_date,
    "Poster_URL" : poster_url
})

# coverting dataframe into csv-- this step isn't necessary, but did it so i'm not constantly dealing with the api directly
filtered_omdb_data_df.to_csv('filtered_omdb_data.csv', index=False)
# filtered_omdb_data_df.head()
```

Our Approach

```
# created a path to the filteredd api csv and read it into a pandas dataframe
filtered_omdb_csv = "filtered_omdb_data.csv"
filtered_omdb_csv_df = pd.read_csv(filtered_omdb_csv)
# filtered_omdb_csv_df.count()
# award_data.count()

# merged the filtered api data csv and awards data csv into a single dataset
merged_movie_data_df = pd.merge(filtered_omdb_csv_df, award_data, left_on="Title", right_on="Nominee")
# merged_movie_data_df.to_csv('merged_movie_data.csv', index=False)
# merged_movie_data_df.head()

# import Eric's csv that adds a seasons column to "merged_movie_data_df"
seasons_omdb_csv = "Seasons_Movie_Data_2.csv"
seasons_omdb_csv_df = pd.read_csv(seasons_omdb_csv)
# seasons_omdb_csv_df.head()
```

Our Approach

```
#Separate seasons into bins with the months that each contain inside
winter = ['Dec', 'Jan', 'Feb']
spring = ['Mar', 'Apr', 'May']
summer = ['Jun', 'Jul', 'Aug']
fall = ['Sep', 'Oct', 'Nov']

#iterate through rows and retrieve second word from 'Release Date'
for index, movie in movie_data_df.iterrows():

    try:
        release_date = movie['Release_Date']
        month = release_date.split(' ')[1]

#Print corresponding bin into 'Seasons' column
        if month in winter:
            movie_data_df.at[index, 'Seasons'] = 'Winter'
        elif month in spring:
            movie_data_df.at[index, 'Seasons'] = 'Spring'
        elif month in summer:
            movie_data_df.at[index, 'Seasons'] = 'Summer'
        elif month in fall:
            movie_data_df.at[index, 'Seasons'] = 'Fall'
    except:
        movie_data_df.at[index, 'Seasons'] = 'Unknown'
```

Our Approach

- What is the count of movie ratings (R, PG, G, etc.) by winning and losing ‘Best Picture’ nominees?

```
# clean up Rated category.
seasons_omdb_csv_df['Rated'] = seasons_omdb_csv_df['Rated'].replace(
    {'NOT RATED': 'Not Rated', 'PASSED': 'Passed', 'UNRATED': 'Not Rated', 'Unrated': 'Not Rated', 'APPROVED': 'Approved'})
```

```
# create dataframe for winning and losing nominees
winning_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Won?"] == "YES"]
# winning_noms

losing_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Won?"] == "NO"]
# losing_noms
```

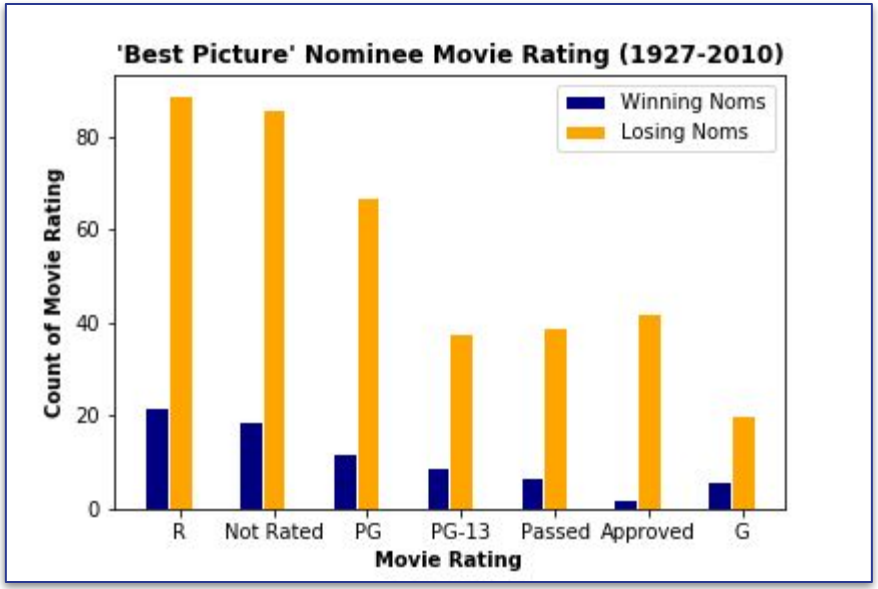
Our Approach

- What is the count of movie ratings (R, PG, G, etc.) by winning and losing ‘Best Picture’ nominees?

```
# set width of bar
barWidth = 0.25

# set height of bar
height_winning_rated = [22, 19, 12, 9, 7, 2, 6]
height_losing_rated = [89, 86, 67, 38, 39, 42, 20]


# Set position of bar on X axis
r1 = np.arange(len(height_winning_rated))
r2 = [x + barWidth for x in r1]

# Make the plot
plt.bar(r1, height_winning_rated, color='navy', width=barWidth, edgecolor='white', label='Winning Noms')
plt.bar(r2, height_losing_rated, color='orange', width=barWidth, edgecolor='white', label='Losing Noms')

# Add xticks on the middle of the group bars
plt.title("'Best Picture' Nominee Movie Rating (1927-2010)", fontweight='bold')
plt.xlabel('Movie Rating', fontweight='bold')
plt.ylabel("Count of Movie Rating", fontweight='bold')

plt.xticks([r + barWidth for r in range(len(height_winning_rated))], ['R', 'Not Rated', 'PG', 'PG-13', 'Passed', 'Approved', 'G']

#create legend, show graphic, and push to .png
plt.legend()
plt.savefig("count_of_rated_grouped.png")
plt.show()
```

Our Approach

- What is the count of movie ratings (R, PG, G, etc.) by winning and losing ‘Best Picture’ nominees?



Our Approach

- What is the count of movie rating (R, PG, G, etc.) by season?

```
winter_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Seasons"] == "Winter"]
spring_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Seasons"] == "Spring"]
summer_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Seasons"] == "Summer"]
fall_noms = seasons_omdb_csv_df[seasons_omdb_csv_df["Seasons"] == "Fall"]

winter_rated = winter_noms['Rated'].value_counts()
# print(winter_rated)

spring_rated = spring_noms['Rated'].value_counts()
# print(spring_rated)

summer_rated = summer_noms['Rated'].value_counts()
# print(summer_rated)

fall_rated = fall_noms['Rated'].value_counts()
# print(fall_rated)

# rated_as_index_ = merged_movie_data_df.set_index('Rated').groupby(['Rated'])
# rated_as_index.head()
```

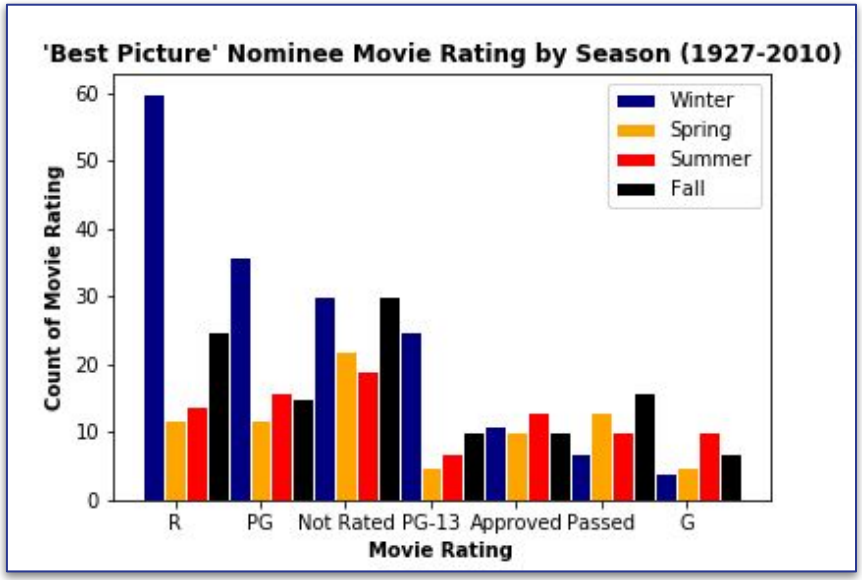
Our Approach

- What is the count of movie rating (R, PG, G, etc.) by season?

```
# set width of bar
barWidth = 0.25

# set height of bar
height_winter_rated = [60, 36, 30, 25, 11, 7, 4]
height_spring_rated = [12, 12, 22, 5, 10, 13, 5]
height_summer_rated = [14, 16, 19, 7, 13, 10, 10]
height_fall_rated = [25, 15, 30, 10, 10, 16, 7]


# Set position of bar on X axis
r1 = np.arange(len(height_winter_rated))
r2 = [x + barWidth for x in r1]
r3 = [x + barWidth for x in r2]
r4 = [x + barWidth for x in r3]

# Make the plot
plt.bar(r1, height_winter_rated, color='navy', width=barWidth, edgecolor='white', label='Winter')
plt.bar(r2, height_spring_rated, color='orange', width=barWidth, edgecolor='white', label='Spring')
plt.bar(r3, height_summer_rated, color='red', width=barWidth, edgecolor='white', label='Summer')
plt.bar(r4, height_fall_rated, color='black', width=barWidth, edgecolor='white', label='Fall')


# Add xticks on the middle of the group bars
plt.title("'Best Picture' Nominee Movie Rating by Season (1927-2010)", fontweight='bold')
plt.xlabel('Movie Rating', fontweight='bold')
plt.ylabel("Count of Movie Rating", fontweight='bold')

plt.xticks([r + barWidth for r in range(len(height_winter_rated))], ['R', 'PG', 'Not Rated', 'PG-13', 'Approved', 'Passed', 'G']

#create legend, show graphic, and push to .png
plt.legend()
plt.savefig("count_of_rated_by_season.png")
plt.show()
```

Our Approach

- What is the count of movie rating (R, PG, G, etc.) by season?



Our Approach

- What is the average IMDB rating of Best Picture award-winning movies by season?

In [2]:
```
import pandas as pd
import requests
import json
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
titles_csv = "movie_titles.csv"
ratings_csv = "merged_movie_data.csv"
omdb_csv = "filtered_omdb_data.csv"
awards_csv = "academy_awards_data_2.csv"
seasons_csv = "Seasons_Movie_data.csv"
```

In [6]:
```
seasons_df = pd.read_csv(seasons_csv, usecols = ['Title', 'imdb_Rating', 'Nominee', 'Won?', 'Seasons'], encoding = 'lat
```

In [11]:
```
winning_noms = seasons_df[seasons_df["Won?"] == "YES"]
winning_noms.head()
```

Out[11]:

	Title	imdb_Rating	Nominee	Won?	Seasons
4	The King's Speech	8.0	The King's Speech	YES	Winter
14	The Hurt Locker	7.6	The Hurt Locker	YES	Summer
23	Slumdog Millionaire	8.0	Slumdog Millionaire	YES	Winter
27	No Country for Old Men	8.1	No Country for Old Men	YES	Fall
30	The Departed	8.5	The Departed	YES	Fall

In [14]:
```
fall_movies = winning_noms[winning_noms['Seasons'] == 'Fall']
spring_movies = winning_noms[winning_noms['Seasons'] == 'Spring']
summer_movies = winning_noms[winning_noms['Seasons'] == 'Summer']
winter_movies = winning_noms[winning_noms['Seasons'] == 'Winter']
```

In [15]:
```
fall_movies.head()
```

Our Approach

- What is the average IMDB rating of Best Picture award-winning movies by season?

Out[15]:

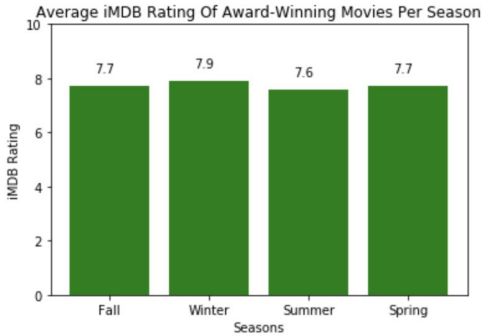
	Title	imdb_Rating	Nominee	Won?	Seasons
27	No Country for Old Men	8.1	No Country for Old Men	YES	Fall
30	The Departed	8.5	The Departed	YES	Fall
67	American Beauty	8.4	American Beauty	YES	Fall
140	Amadeus	8.3	Amadeus	YES	Fall
204	The French Connection	7.8	The French Connection	YES	Fall

```
In [19]: fall_movies_imdb = fall_movies['imdb_Rating']
         spring_movies_imdb = spring_movies['imdb_Rating']
         summer_movies_imdb = summer_movies['imdb_Rating']
         winter_movies_imdb = winter_movies['imdb_Rating']

In [21]: fall_movies_imdb.mean()
Out[21]: 7.735714285714286

In [22]: spring_movies_imdb.mean()
Out[22]: 7.741176470588235

In [23]: summer_movies_imdb.mean()
Out[23]: 7.590909090909091

In [24]: winter_movies_imdb.mean()
Out[24]: 7.905882352941177
```

Our Approach

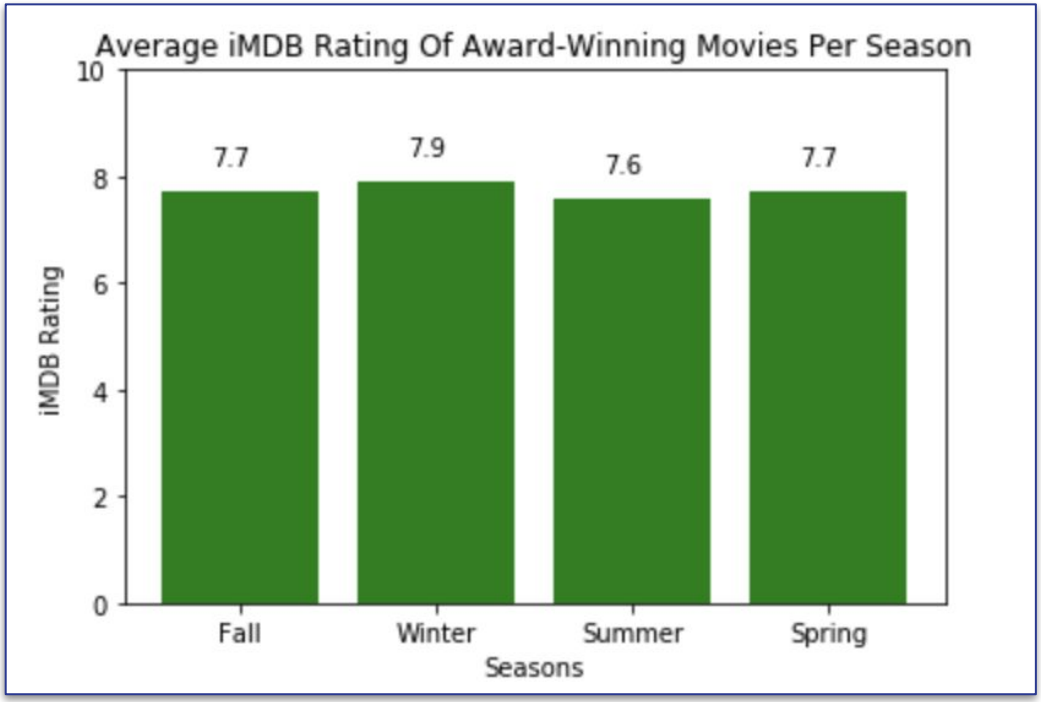
- What is the average IMDB rating of Best Picture award-winning movies by season?

In [39]:
```
x_values = ['Fall','Winter','Summer','Spring']
y_values = [7.7, 7.9, 7.6, 7.7]
plt.ylim([0,10])
bars = plt.bar(x_values, y_values, color = 'g', alpha = 1.0, align="center")
plt.title('Average iMDB Rating Of Award-Winning Movies Per Season')
plt.xlabel('Seasons')
plt.ylabel('iMDB Rating')
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + .25, yval + 0.5, yval)
plt.show()
```



Our Approach

- What is the average IMDB rating of Best Picture award-winning movies by season?



Our Approach

- Is there a trend in genre by winning and nominated movies?
- Most of the movies we had on the list had multiple genres. We needed to figure out how we wanted to analyze this.
- We decided to count the amount of times a genre was used within our data set. We did this by doing a word count of the genre column.

```
import csv
from collections import Counter
from collections import defaultdict

word_counts = {}
with open('merged_movie_data.csv', 'r') as csvfile:
  reader = csv.reader(csvfile)
  next(reader)
  for row in reader:
        csv_words = row[1].split(", ")
        for word in csv_words:
            if word in word_counts:
                word_counts[word] += 1;
            else:
                word_counts[word] = 1;

word_counts
#this is for all of the movies
```

Our Approach

- Next we took this information and made it into a bar chart.

```
 # Create an array that contains the number of users each language has
genre = ['Drama', 'Adventure', 'Biography', 'Thriller', 'Comedy', 'Musical',
         'Romance', 'Crime', 'Sci-Fi', 'War', 'History', 'Fantasy', 'Mystery',
         'Music', 'Action', 'Family', 'Western', 'Animation', 'Sport', 'Film-Noir', 'Horror']
count = [413, 49, 89, 57, 100, 36, 203, 54, 4, 56, 65, 25, 35, 21, 24, 25, 17, 4, 17, 10, 3]
```

```
plt.bar(genre, count, color='r', alpha=1.0, align="center")
plt.xticks(rotation=90)

plt.title("Genre Count Of Best Picture Nominees")
plt.xlabel("Genre")
plt.ylabel("Count of Genre of Nominees")
```

Our Approach

- Next we did the same process, but for only the winning movies

```
# create a path to csv and read it into a pandas dataframe
movies_csv = 'merged_movie_data.csv'
movies_df = pd.read_csv(movies_csv, usecols = ['Title', 'Genre', 'Meta_Score', 'imdb_Rating',
                                               'Box_Office', 'Rated', 'Studio', 'Release_Date',
                                               'Poster_URL', 'Plot', 'Year', 'Category', 'Nominee',
                                               'Won?'], encoding = 'latin-1')

# create another dataframe that only includes winners of the Best Picture category
movies_data = movies_df.loc[movies_df['Won?'] == 'YES', :]

# create list of best picture nominees
best_picture_wins = movies_data["Won?"]

movies_data.head()
```

```
# coverting dataframe into csv
movies_data.to_csv('best_picture_winners.csv', index=False)
```

```
winner_word_counts = {}
with open('best_picture_winners.csv', 'r') as csvfile:
  reader = csv.reader(csvfile)
  next(reader)
  for row in reader:
       csv_words = row[1].split(", ")
       for word in csv_words:
           if word in winner_word_counts:
               winner_word_counts[word] += 1;
           else:
               winner_word_counts[word] = 1;

winner_word_counts
```

```
genre_winners = ['Drama', 'Adventure', 'Biography', 'Thriller', 'Comedy', 'Musical',
                 'Romance', 'Crime', 'Sci-Fi', 'War', 'History', 'Fantasy', 'Mystery',
                 'Music', 'Action', 'Family', 'Western', 'Animation', 'Sport', 'Film-Noir', 'Horror']
count_winners = [72, 9, 17, 10, 12, 9, 29, 11, 0, 15, 14, 1, 2, 2, 3, 6, 3, 0, 3, 1, 0]
```

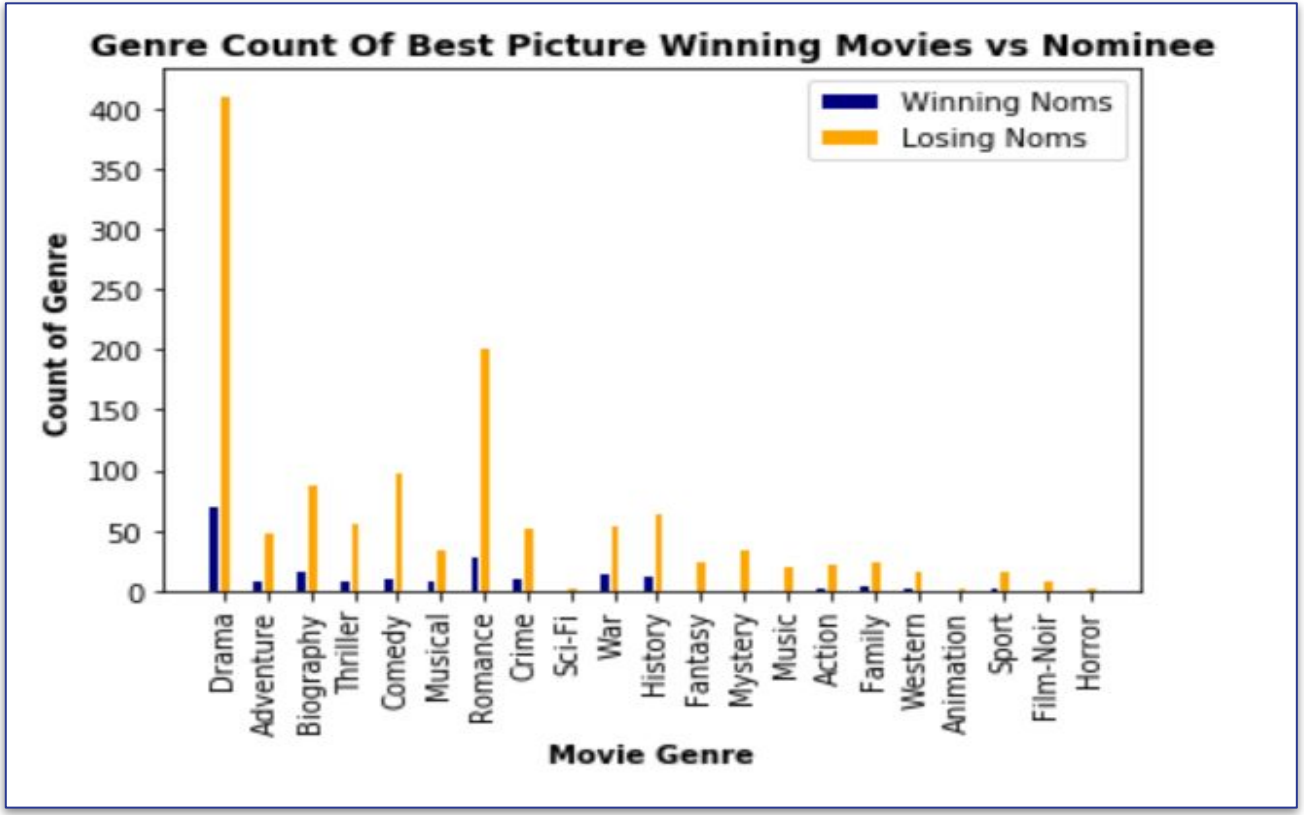
```
plt.bar(genre, count, color='r', alpha=1.0, align="center")
plt.xticks(rotation=90)

plt.title("Genre Count Of Best Picture Winning Movies")
plt.xlabel("Genre")
plt.ylabel("Count of Genre of Winning Movies")
```

Our Approach

- Next we put this data together.

```
# set width of bar
barWidth = 0.25

# set height of bar
nominee_genre_count = [413, 49, 89, 57, 100, 36, 203, 54, 4, 56, 65, 25, 35, 21, 24, 25, 17, 4, 17, 10, 3]
winner_genre_count = [72, 9, 17, 10, 12, 9, 29, 11, 0, 15, 14, 1, 2, 2, 3, 6, 3, 0, 3, 1, 0]


# Set position of bar on X axis
r1 = np.arange(len(nominee_genre_count))
r2 = [x + barWidth for x in r1]

# Make the plot
plt.bar(r1, winner_genre_count, color='navy', width=barWidth, edgecolor='white', label='Winning Noms')
plt.bar(r2, nominee_genre_count, color='orange', width=barWidth, edgecolor='white', label='Losing Noms')


# Add xticks on the middle of the group bars
plt.title("Genre Count Of Best Picture Winning Movies vs Nominee", fontweight='bold')
plt.xlabel('Movie Genre', fontweight='bold')
plt.ylabel("Count of Genre", fontweight='bold')

plt.xticks([r + barWidth for r in range(len(nominee_genre_count))], ['Drama', 'Adventure', 'Biography',
                                                                     'Thriller', 'Comedy', 'Musical', 'Romance',
                                                                     'Crime', 'Sci-Fi', 'War', 'History', 'Fantasy',
                                                                     'Mystery', 'Music', 'Action', 'Family', 'Western',
                                                                     'Animation', 'Sport', 'Film-Noir', 'Horror'],
           rotation=90)


#create legend, show graphic
plt.legend()
plt.show()
```

Our Approach



Our Approach

- Then we found a different way of analyzing this information, by doing word clouds

```
merged_movie_data_df = pd.read_csv("merged_movie_data.csv")


# creating wordcloud (https://www.datacamp.com/community/tutorials/wordcloud-python)
word_cloud = WordCloud(max_font_size=75, max_words=100,
                       background_color="white").generate(' '.join(merged_movie_data_df['Genre']))


# generate plot
plt.title("Genre Wordcloud")
plt.imshow(word_cloud)
plt.axis("off")
plt.show()
```

Our Approach

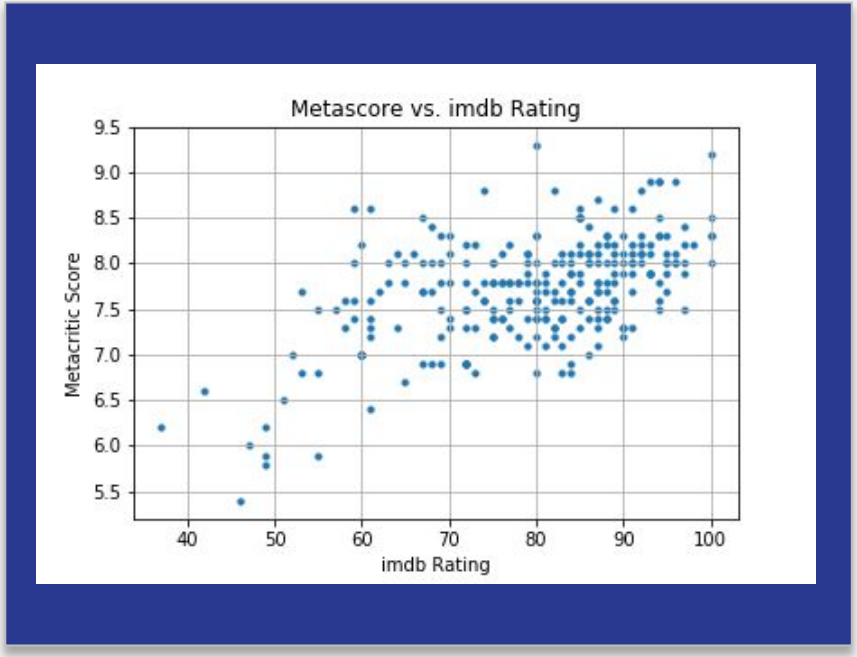
Our Approach

- What is the relationship between imdb rating and metacritic score?

```
plt.scatter(movie_data_df['Meta_Score'], movie_data_df['imdb_Rating'], marker='o', s=10)

plt.title('Metascore vs. imdb Rating')
plt.ylabel('Metacritic Score')
plt.xlabel('imdb Rating')
plt.grid(True)

plt.savefig('Metascore vs. imdb.png')

plt.show
```

Our Approach

- What is the relationship between imdb rating and metacritic score?



Our Approach

- Is there a significant difference between the means of imdb ratings for different seasons?

```
fall_movies = movie_data_df[movie_data_df['Season'] == 'Fall']
spring_movies = movie_data_df[movie_data_df['Season'] == 'Spring']
summer_movies = movie_data_df[movie_data_df['Season'] == 'Summer']
winter_movies = movie_data_df[movie_data_df['Season'] == 'Winter']

fall_movies_imdb = fall_movies['imdb_Rating']
spring_movies_imdb = spring_movies['imdb_Rating']
summer_movies_imdb = summer_movies['imdb_Rating']
winter_movies_imdb = winter_movies['imdb_Rating']
```

```
stats.ttest_ind(fall_movies_imdb, winter_movies_imdb, equal_var=False)
```

```
Ttest_indResult(statistic=-1.7278963018931661, pvalue=0.08561486632333437)
```

Key Findings/Wrap-Up

Key Findings

1. ‘Rated R’ movies in the Drama/Romance category released in the Winter are more likely to win ‘Best Picture’

2. Horror, Sci-Fi, and Animation are less likely to receive a ‘Best Picture’ nomination

3. Time of year the movie was released did not have an impact on IMDB rating

4. From a TTest, there is not a significant difference in the means of the IMDB ratings across seasons

Wrap-up

- Some difficulties we ran into:
  - Reading data from API into a dataframe
  - Creating new column to contain Seasons
  - Most movies are in multiple genres, causing issues in how to analyze this data

Ask a Question or Three.