Final Project Submission

Please fill out:

- Student name: Zeth Abney
- Student pace: flex (20 week)
- Scheduled project review date/time: TBD
- Instructor name: Matt Carr
- Blog post URL: TBA

MS Studios Analysis for Creative Direction and Timline

- some photo here!

Overview

This project seeks to discover the most productive creative direction and production timeline for the newly established Microsoft Studios. The analyses on productivity for a given title or genre use measures of financial profitablity (profit/loss ratio and gross profit) and measures of public appeal (average viewer rating and a popularity score), and it explores both of these measures in the context of release date and genre. This analysis is intended to aid decision makers of MS Studios in determing the most profitable and/or appealing creative direction and production timeline.

Business problem

To date, MS Studios has released no original titles, so there is no data on profit or consumer feedback available regarding MS Studios branded content.

With that in mind, it follows that MS Studios has no brand recognition in the industry, in other words the public has no broad association with the studio/brand and a certain style or genre (e.g. Spiderman's association to Sony Pictures). So in order to explore which genre would present the most potential earning or garner the most attention, this analyses investigates which genres show the the greatest profitability and public appeal.

For the same reasons, MS Studios has set no precedent for a release cycle, meaning the public does not know what time of year to expect another battery of new-releases like they do for a service such as Netflix or Hulu. As a newcomer in the space, it is probably best to follow suit with what time of year the most profitable content is released, or what time of year seems to get the most attention. So the analyses seeks to determine what is best time of year to make a new release in terms of profitablity and/or public appeal.

Hopefully, this analyses will give a hint as to what *genre of film to produce and when to release it*.

Data Understanding

This analysis makes use of data collected from thenumbers.com (tn) for measures of profitability, and tmdb (The Movie Database) for analyses of public appeal, along with an imdb dataset (Internet Movied Database) used to map genre to a given record in either the tn or tmdb datasets.

The initial dataset available had 11 csv/tsv files. These files for passed as tables into a sqlite databes named 'testBase.db' in the 'setup_testBase.ipynb' journal, then testBase was explored in 'first_round_eda.ipynb' where the tables needed for analyses were explored and selected. Then using the 'setup_hollywood.ipynb' journal, the desired tables were queried from testBase into pandas dataframes, which were then cleaned and then passed as a table into the sqlite databses 'hollywood.db'. This journal queries the data from the hollywood database.

In [1]:
```
import sqlite3
import seaborn as sns
import pandas as pd
import numpy as np
from datetime import date
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
conn = sqlite3.connect('hollywood.db')
cur = conn.cursor()
```

Selecting the data measuring financial performance and performing secondary cleaning

- The dataset on measures of financial performance comes from thenumbers.com, the table is linked with another table from the an imdb data set. The imdb data is joined on to provide the genre(s) for each title.
- gross_profit is an engineered feature, representing the difference in revenue from expenses, acquired by substracting 'international_boxoffice' from 'production_budget'. It is impoprtant to note this does not (apparently) include marketing expenses, and it is using global revenue (not only domestic).
- 'PL_ratio' is also an engineered feature, it is the profit/loss ratio calculated by the dividing 'gross_profit' by 'production_budget'.
- 'release_date' and 'release_quarter' are engineered features acquired by using the datetime module inline with pandas on the 'release' column.

In [3]:

```
budgets_qry = """
SELECT DISTINCT
    movie AS title,
    genres,
    release_date AS release,
    production_budget,
    domestic_gross AS USA_boxoffice,
    worldwide_gross AS international_boxoffice
FROM tn_movie_budgets
JOIN imdb_title_basics AS itb
    ON itb.primary_title = title
WHERE ((USA_boxoffice & international_boxoffice) > 0)
;
"""
budgRev_df = pd.read_sql(budgets_qry,conn)

budgRev_df['gross_profit'] = (budgRev_df['international_boxoffice'] - budgRev
budgRev_df['PL_ratio'] = (round(budgRev_df['gross_profit'] / budgRev_df['proc

budgRev_df['genres'] = [li.split(",") for li in budgRev_df['genres']]

quarters = [pd.Timestamp(date).quarter for date in budgRev_df['release']] #ge
budgRev_df['release_quarter'] = quarters
months = [pd.Timestamp(date).month for date in budgRev_df['release']] # get m
budgRev_df['release_month'] = months

budgRev_df = budgRev_df.sort_values('gross_profit', ascending=False).reset_ir
top_financials_df = budgRev_df
# .iloc[:700,1:]

top_financials_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2711 entries, 0 to 2710
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   index                    2711 non-null   int64  
 1   title                    2711 non-null   object 
 2   genres                   2711 non-null   object 
 3   release                  2711 non-null   object 
 4   production_budget        2711 non-null   int64  
 5   USA_boxoffice            2711 non-null   int64  
 6   international_boxoffice  2711 non-null   int64  
 7   gross_profit             2711 non-null   int64  
 8   PL_ratio                 2711 non-null   float64
 9   release_quarter          2711 non-null   int64  
 10  release_month            2711 non-null   int64  
dtypes: float64(1), int64(7), object(3)
memory usage: 233.1+ KB
```

Selecting data to measure public appeal and performing secondary data cleaning

- The dataset on measures of public appeal is acquired from The Movie Database (tmdb) and joined with the same imdb dataset to provide the genre(s) for each title.
- 'release_month' and 'release_quarter' are engineered the same way they are in top_financials_df above.
- according to the tmdb API docs the popularity score is an engineered feature (by tmdb) and for movies is determined by considering:
  - Number of votes for the day
  - Number of views for the day
  - Number of users who marked it as a "favourite" for the day
  - Number of users who added it to their "watchlist" for the day
  - Release date
  - Number of total votes
  - Previous days score (due to the nature of this project it is unknown what on what date this data was aquired)

In [4]:

```
reviews_qry = """
SELECT DISTINCT
    tbm.title,
    itb.genres,
    tbm.release_date AS release,
    tbm.popularity,
    tbm.vote_average AS average_rating,
    tbm.vote_count
FROM tmdb_movies AS tbm
JOIN imdb_title_basics AS itb ON tbm.title = itb.primary_title
WHERE (popularity > 1)
ORDER BY popularity DESC
;
"""
#LIMIT 3200
reviews_df = pd.read_sql(reviews_qry,conn)

reviews_df['genres'] = [li.split(",") for li in reviews_df['genres']]

quarters = [pd.Timestamp(date).quarter for date in reviews_df['release']] #ge
reviews_df['release_quarter'] = quarters
months = [pd.Timestamp(date).month for date in reviews_df['release']] # get m
reviews_df['release_month'] = months

public_appeal_df = reviews_df
public_appeal_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12725 entries, 0 to 12724
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   title            12725 non-null  object 
 1   genres           12725 non-null  object 
 2   release          12725 non-null  object 
 3   popularity       12725 non-null  float64
 4   average_rating   12725 non-null  float64
 5   vote_count       12725 non-null  int64  
 6   release_quarter  12725 non-null  int64  
 7   release_month    12725 non-null  int64  
dtypes: float64(2), int64(3), object(3)
memory usage: 795.4+ KB
```

Aggregating and grouping the data for plotting

In [5]:

```
df = top_financials_df

# create dataframes grouping financial measures by month and quarter.
quant_by_month = df.groupby('release_month').mean().reset_index()
quant_by_quarter = df.groupby('release_quarter').mean().reset_index()

#round off data for readibility and clarity
quant_by_month['PL_ratio'] = [round(pl,2) for pl in quant_by_month.PL_ratio]
quant_by_month['gross_profit'] = [round(pro,0) for pro in quant_by_month.gros

quant_by_quarter['PL_ratio'] = [round(pl,2) for pl in quant_by_quarter.PL_rat
quant_by_quarter['gross_profit'] = [round(pro,0) for pro in quant_by_quarter.
quant_by_quarter['release_month'] = [round(month,0) for month in quant_by_qua

# asign idiomatic variables to cleaned dataframes
quant_month_means = quant_by_month[['release_month','gross_profit','PL_ratio'
quant_quarter_means = quant_by_quarter[['release_quarter','gross_profit','PL_

################################################################################

# create dataframes grouping financial measures by genre.
pre_explode_quants = df.explode('genres')
pre_explode_quants = pre_explode_quants[pre_explode_quants.genres != 'News']
pre_explode_quants = pre_explode_quants[pre_explode_quants.genres != 'Game-Sh
explode_quants = pre_explode_quants.groupby('genres').mean().reset_index()

#round off data for readibility and clarity
explode_quants['PL_ratio'] = [round(pl,2) for pl in explode_quants.PL_ratio]
explode_quants['gross_profit'] = [round(pro,0) for pro in explode_quants.gros
quarters = [round(quarter,0) for quarter in explode_quants['release_quarter']
explode_quants['release_quarter'] = quarters
months = [round(month,0) for month in explode_quants['release_month']] # roun
explode_quants['release_month'] = months

# asign idiomatic variable to cleaned dataframes
quant_genre_means = explode_quants[['genres','gross_profit','PL_ratio','relea

# creating and reversing list for the sake of a plot latter
quant_genre_list = list(set(quant_genre_means['genres']))
quant_genre_list.reverse()
```

Creating tables of the financial measures (pl ratio and gross profit) for each genre and month of the year. Each tables contain the set of all values for a given feature within the given genre. dataframes are asigned variable names according to the respective genre.

In [6]:
```
all_profits_list = []

for genre in quant_genre_list:
    if genre == 'Sci-Fi':
        tableName = 'SciFi_profs'
        vars()[tableName] = pre_explode_quants[pre_explode_quants['genres'] =
        all_profits_list.append(tableName)
    else:
        tableName = f'{genre}_profs'
        vars()[tableName] = pre_explode_quants[pre_explode_quants['genres'] =
        all_profits_list.append(tableName)

prof_tabs =[
Horror_profs,
Action_profs,
Animation_profs,
War_profs,
Western_profs,
Mystery_profs,
Crime_profs,
SciFi_profs,
Fantasy_profs,
History_profs,
Family_profs,
Drama_profs,
Romance_profs,
Music_profs,
Musical_profs,
Documentary_profs,
Comedy_profs,
Adventure_profs,
Biography_profs,
Thriller_profs,
Sport_profs]
profit_tuples = tuple(zip(all_profits_list,prof_tabs))
```

In [7]:
```
# Same idea as above but for months of the year instead of genres
monthly_quants_list = []

months = [(1,'January'),(2,'February'),(3,'March'),(4,'April'),(5,'May'),(6,'

for t in months:
    vars()[t[1]] = budgRev_df[budgRev_df['release_month']==t[0]][['gross_prof
    monthly_quants_list.append(t[1])
# print(monthly_quants_list)

monthly_quants = [(1,January),(2,February),(3,March),(4,April),(5,May),(6,Jur
```

Uncomment this variables to see their content. Everything useful from the above two cells can be accesed here.

In [8]:

```
# pre_explode_quants #top_financials_df exploded by genre
# explode_quants #pre_explode_quants grouped by genre

# quant_month_means # averages for the month
# quant_quarter_means # averages for the quarter
# quant_genre_means # averages for the genre
# quant_genre_list # list of genres with profitability data

# all_profits_list # list of the names of the dfs instantiated by the forloop
# prof_tabs #list of genre/profitability dfs
# profit_tuples

# monthly_quants_list
# monthly_quants
```

repeating the process for measures of public appeal

In [9]:

```
df = public_appeal_df

# create a dataframe grouping appeal measures by month and quarter
qual_by_month = df.groupby('release_month').mean().reset_index()
qual_by_quarter = df.groupby('release_quarter').mean().reset_index()

#round off data for readibility and clarity
qual_by_month['popularity'] = [round(num,2) for num in qual_by_month.populari
qual_by_month['average_rating'] = [round(num,2) for num in qual_by_month.aver
qual_by_month['vote_count'] = [round(num,0) for num in qual_by_month.vote_cou
qual_by_month['release_month'] = [round(month,0) for month in qual_by_month.r

qual_by_quarter['popularity'] = [round(num,2) for num in qual_by_quarter.popu
qual_by_quarter['average_rating'] = [round(num,2) for num in qual_by_quarter.
qual_by_quarter['vote_count'] = [round(num,0) for num in qual_by_quarter.vote
qual_by_quarter['release_month'] = [round(month,0) for month in qual_by_quart

#############################################################################

# group qualitative measures by genre
pre_exp_quals = df.explode('genres')
pre_exp_quals = pre_exp_quals[pre_exp_quals.genres != 'News'] #irrelevent ger
pre_exp_quals = pre_exp_quals[pre_exp_quals.genres != 'Game-Show'] #irrelever
exp_quals = pre_exp_quals.groupby('genres').mean().reset_index()

#round off data for readibility and clarity
exp_quals['popularity'] = [round(pop,2) for pop in exp_quals.popularity]
exp_quals['average_rating'] = [round(rat,2) for rat in exp_quals.average_rati
exp_quals['vote_count'] = [round(count,0) for count in exp_quals.vote_count]
exp_quals['release_quarter'] = [round(q,0) for q in exp_quals.release_quarter
exp_quals['release_month'] = [round(m) for m in exp_quals.release_month]

quals_by_genre = exp_quals #set more idiomatic variable name

# creating and reversing list for the sake of a plot latter
qual_genre_list = list(quals_by_genre['genres'])
qual_genre_list.reverse()
```

In [10]:

```
# essentially the function as the forloop for financial measures but
# using measures of appeal data.
all_appeal_list = []

for genre in qual_genre_list:
    if genre == 'Sci-Fi':
        tableName = 'SciFi_appeal'
        vars()[tableName] = pre_exp_quals[pre_exp_quals['genres']==genre][['a
        all_appeal_list.append(tableName)
    else:
        tableName = f'{genre}_appeal'
        vars()[tableName] = pre_exp_quals[pre_exp_quals['genres']==genre][['a
        all_appeal_list.append(tableName)

appeal_tabs = [
War_appeal,
Thriller_appeal,
Sport_appeal,
SciFi_appeal,
Romance_appeal,
Mystery_appeal,
Musical_appeal,
Music_appeal,
Horror_appeal,
History_appeal,
Fantasy_appeal,
Family_appeal,
Drama_appeal,
Documentary_appeal,
Crime_appeal,
Comedy_appeal,
Biography_appeal,
Animation_appeal,
Adventure_appeal,
Action_appeal]

appeal_tuples = tuple(zip(all_appeal_list,appeal_tabs))
```

In [11]:

```
# same idea as above but for months instead of genres
monthly_appeals_list = []

months = [(1,'January'),(2,'February'),(3,'March'),(4,'April'),(5,'May'),(6,'

for t in months:
    vars()[t[1]] = public_appeal_df[public_appeal_df['release_month']==t[0]][
    monthly_appeals_list.append(t[1])
print(monthly_appeals_list)

monthly_appeals = [(1,January),(2,February),(3,March),(4,April),(5,May),(6,Ju
```

```
['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
'September', 'October', 'November', 'December']
```

Uncomment this variables to see their content. Everything useful from the above two cells can be accesed here.

In [12]:

```
# public_appeal_df
# pre_exp_quals #(public_appeal_df exploded by genre)
# exp_quals (pre_exp_quals grouped by genre)

# qual_by_month # averages by month
# qual_by_quarter # averages by quarter
# quals_by_genre # averages by genre
# qual_genre_list # list of genre with public appeal data

# all_appeal_list # list of genre based dataframes instantiated by for loop
# appeal_tabs #list of the actual dataframes from the for loop
# appeal_tuples

# monthly_appeals_list
# monthly_appeals
```

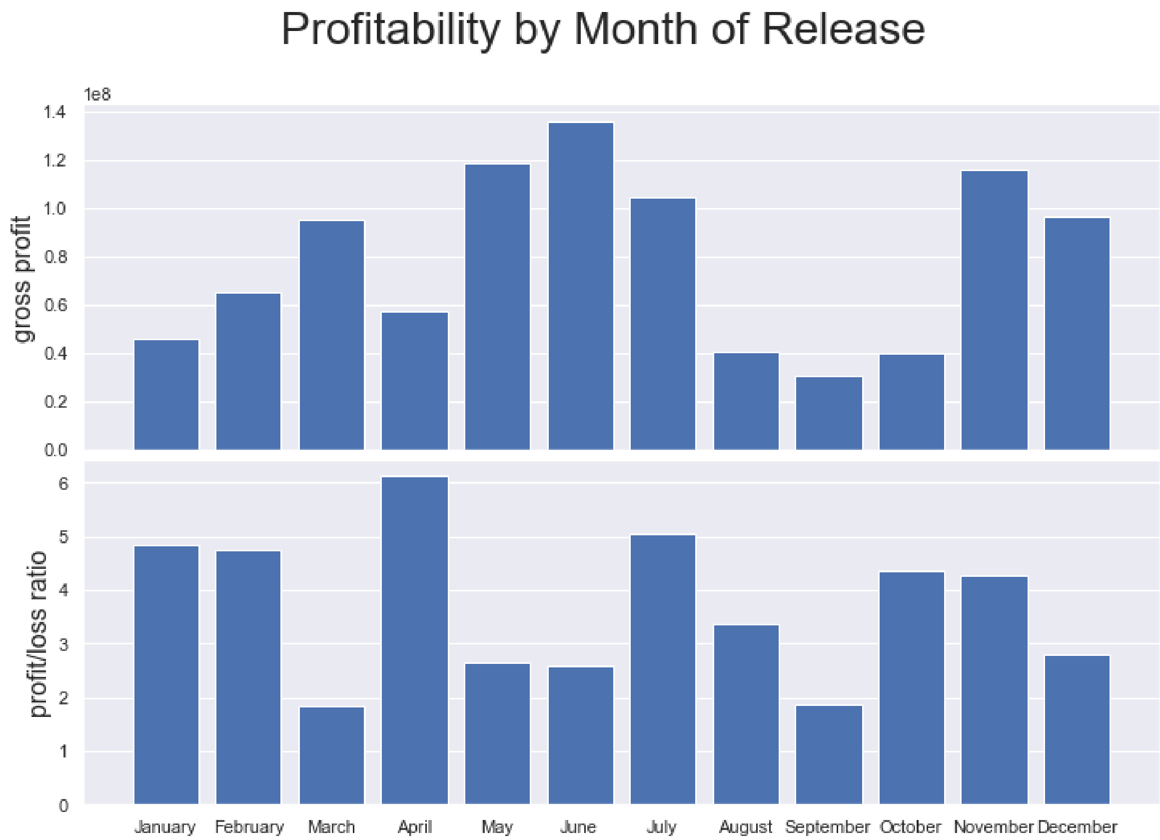
Descriptive analysis and visualization

- Average profitability and average public appeal based on the release date
- Statistical behavior of profitability and public appeal by genre
- Correlation of gross profit to ROI (pl ratio) and popularity to average rating

Typical profitability and appeal based on the month of release

In [31]:

```
months = ['January','February','March','April','May','June','July','August','

fig, (ax1, ax2) = plt.subplots(2,1)
plt.subplots_adjust(hspace=.03)

###########################################################################
# ax1_cmap = mpl.colors.LinearSegmentedColormap.from_list('Greens', ['darksea
# ax1_norm = plt.Normalize(quant_month_means['gross_profit'].min(), quant_mon
# ax1_colors = ax1_cmap(ax1_norm(quant_month_means['gross_profit']))

ax1.set_ylabel("gross profit",fontsize=16)
ax1.set_xticks([])
ax1.bar(months,quant_month_means['gross_profit'])
ax1.grid(axis='x')

###########################################################################

# ax2_cmap = mpl.colors.LinearSegmentedColormap.from_list('Greens', ['darksea
# ax2_norm = plt.Normalize(quant_month_means['PL_ratio'].min(), quant_month_m
# ax2_colors = ax2_cmap(ax2_norm(quant_month_means['PL_ratio']))


ax2.set_ylabel("profit/loss ratio",fontsize=16)
ax2.bar(months,quant_month_means['PL_ratio'],alpha=1)
ax2.grid(axis='x')



fig.canvas.draw()
fig.suptitle("Profitability by Month of Release", fontsize=28)
fig.set_figheight(8)
fig.set_figwidth(12)
sns.set_style()

plt.show()
```



In [14]:

```
# uncomment here to see descriptive analysis of either plot
# quant_month_means['gross_profit'].describe()
# quant_month_means['PL_ratio'].describe()
```

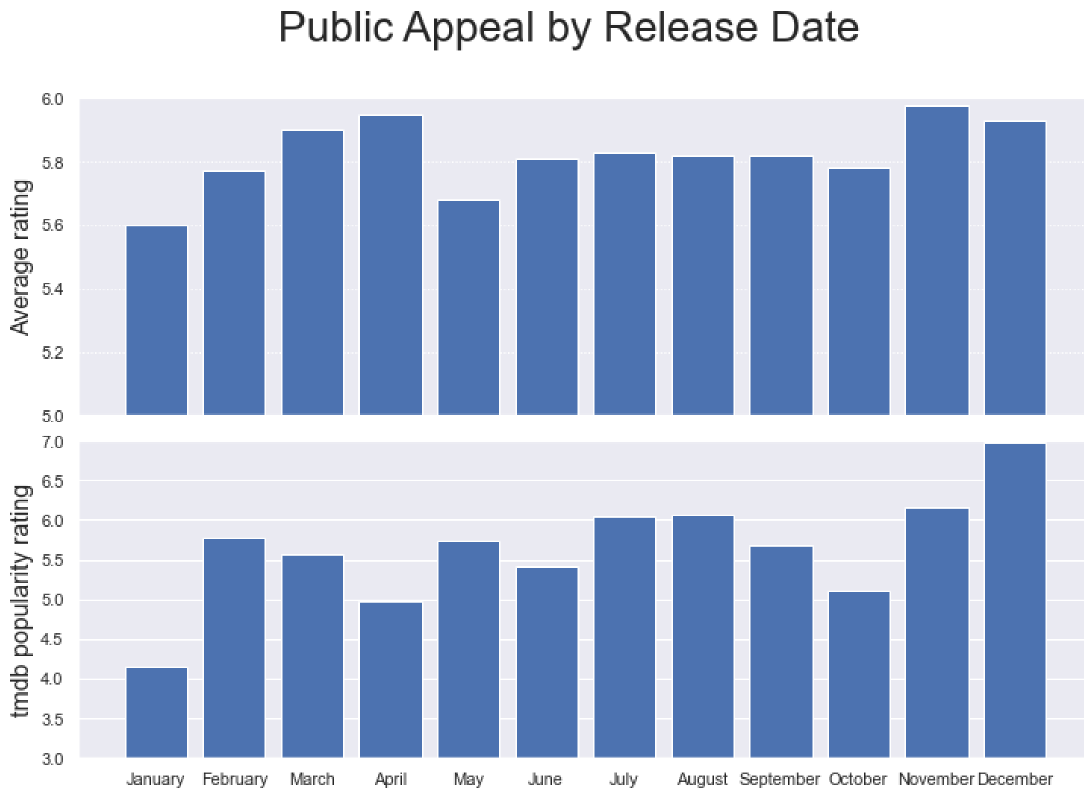
Curiously, ROI and gross profit seem to follow (roughly) opposite trends. This is probably one of the most informative visualizations in this analysis. Apprently, months where the gross profit is greatest, ROI is the leanest.

whichever plot is used to determine a target release date should be chosen based on the business goals for the film:

- if generating cashflow as a whole is the priority I suggest setting a production timeline according to the plot showing P/L raitos to ensure that however great or small the budget is, every dollar will return by several factors
- otherwise if it is more important to develop a brand (Marvel's MCU for example) then I recommend setting a production timeline according to the plot showing gross profits. Showing a significant gross profit even with an expensive budget would appear more attractive to investors and lenders, providing robust financial resources to build a brand or series over time.

In [30]:

```
months = ['January','February','March','April','May','June','July','August','

fig, (ax1, ax2) = plt.subplots(2,1)
fig.canvas.draw()
fig.suptitle("Public Appeal by Release Date", fontsize=28)

fig.set_figheight(8)
fig.set_figwidth(12)
plt.subplots_adjust(hspace=.08)

# ax1_cmap = mpl.colors.LinearSegmentedColormap.from_list('Wistia', ['yellow'
# ax1_norm = plt.Normalize(qual_by_month['average_rating'].min(), qual_by_mon
# ax1_colors = ax1_cmap(ax1_norm(qual_by_month['average_rating']))

ax1.set_ylabel("Average rating",fontsize=16)
ax1.set_xticks([])
ax1.bar(months,qual_by_month['average_rating'],alpha=1)
ax1.grid(axis='y',ls='dotted',zorder=0.0)
ax1.set_ylim(5,6)
sns.set_theme()

###########################################################################

# ax2_cmap = mpl.colors.LinearSegmentedColormap.from_list('Wistia', ['yellow'
# ax2_norm = plt.Normalize(qual_by_month['popularity'].min(), qual_by_month['
# ax2_colors = ax2_cmap(ax2_norm(qual_by_month['popularity']))

ax2.set_ylabel("tmdb popularity rating",fontsize=16)
ax2.bar(months,qual_by_month['popularity'],alpha=1)
ax2.grid(axis='x')
ax2.set_ylim(3,7)

sns.set_theme()

plt.show()
```



This figure suggest the best times of year to release similarly to the profitability figure above. Please take note however, of the x axis tick labels and see that range spanned by the values in either table is not incredibly large. So the early summer and mid winter is apparently the most appealing time of year to release a movie. However, it does not make that big of a difference, so sacrificing production quality to meet a deadline based on this figure is probably not *the right hill to die on*, so-to-speak, but it does somewhat inform how much attention and favor a film might gain depending on when it is debuted.

Analyzing the figures together, depending on the studios branding and earnings goals for a film, it appears that a good release date will fall within the holiday stretch from November through January, or several weeks in the peak of summer from May through August.

Statistical behaviour of profits and public appeal by genre

There is an overwhelming amount of genres represented in the dataset, so its probably best to focus in an a some subset of the data. It is also important to note that a single title might be present in multiple genres. For example, most action movies are also represented in adventure. Also, some genres hold as little as two total records. To make the visualizations more informative, the genres with the *greatest volume* of data points are extracted from the initial data. To do this, a data series is created cointaining the lengths of each genre's dataframe, sumary statistics are gathered and then the fourth quartile (of volume of data)is selected for plotting.

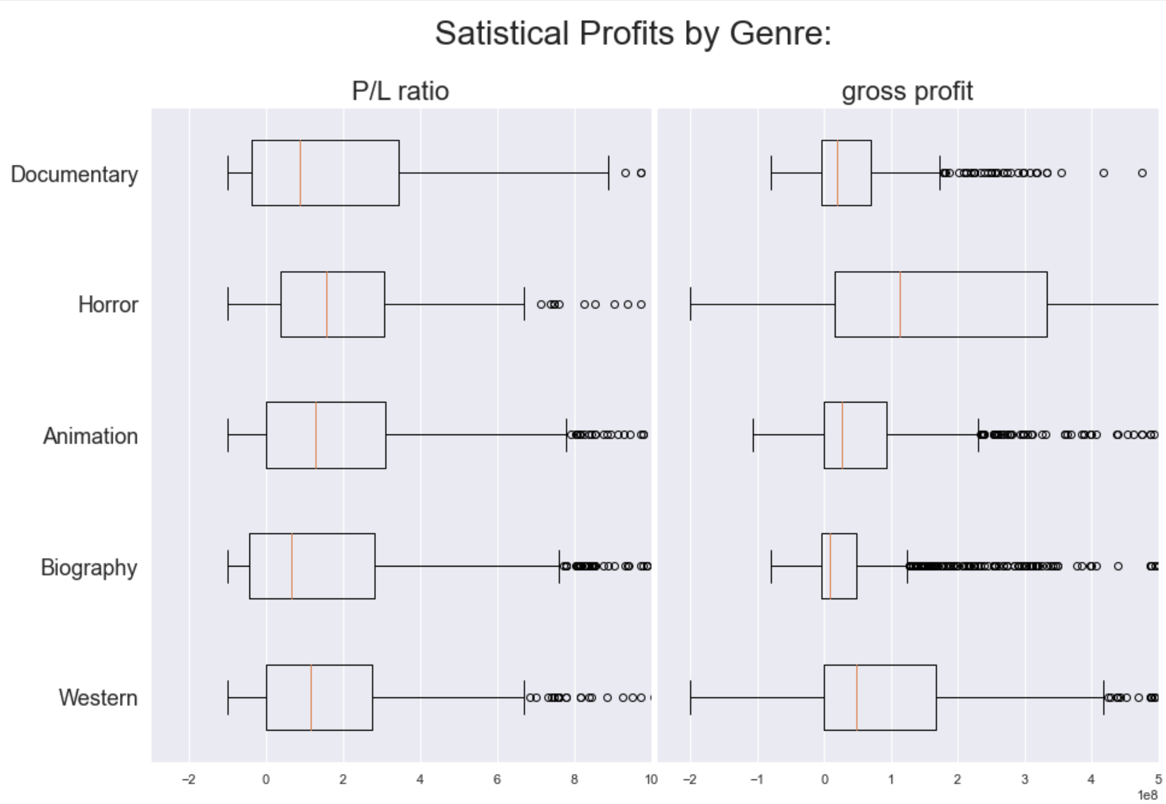
In [16]:
```
# There is a large variance with the amount of data available for a given gen
# here we are filtering outliers with minimal data available.
genre_profit_sizes = pd.DataFrame([(genre,len(tab)) for genre,tab in profit_t
genre_profit_sizes.describe()
# filtering for the third and fourth quartile of dfs based on volume of data
populous_profits = [t for t in profit_tuples if len(t[1]) > 341]
len(populous_profits)
```

Out[16]: 5

In [17]:
```
# Repeat this process for appeal data
genre_appeal_sizes = pd.DataFrame([(genre,len(tab)) for genre,tab in appeal_t
genre_appeal_sizes.describe()
# filtering for the third and fourth quartile of dfs based on volume of data
populous_appeals = [t for t in appeal_tuples if len(t[1]) >= 1882]
len(populous_appeals)
```

Out[17]: 5

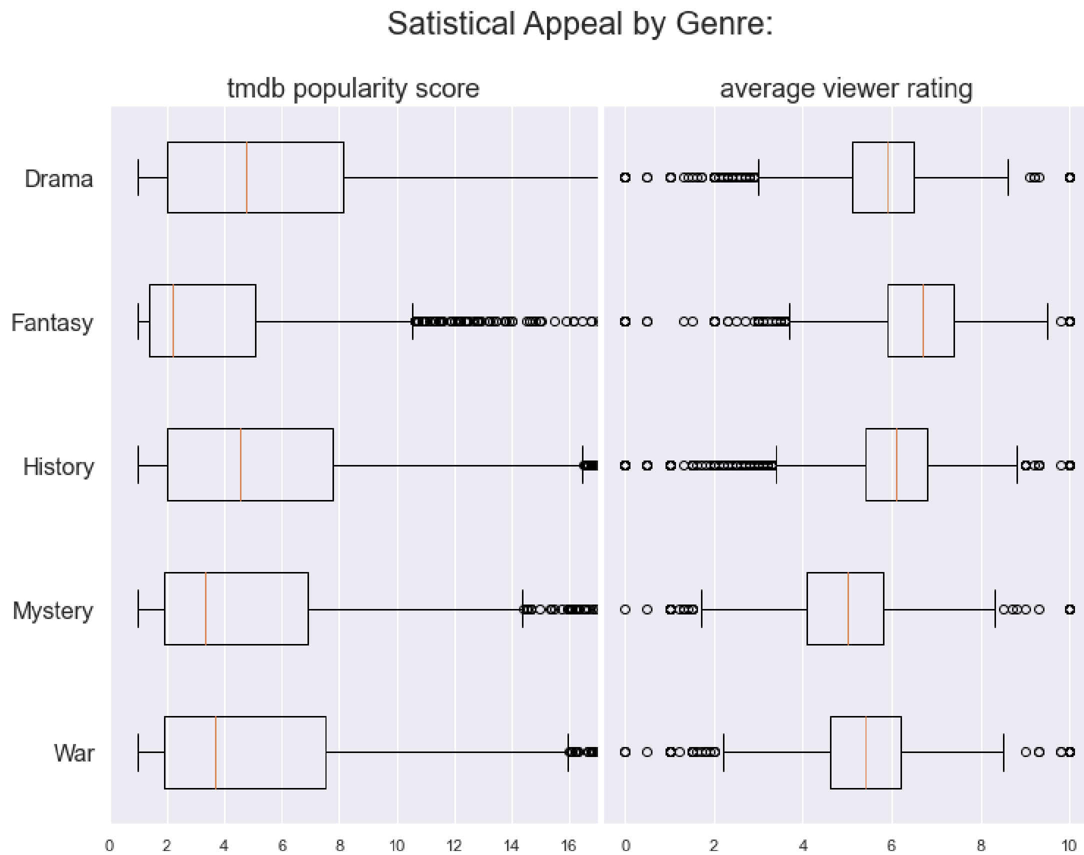
In [18]:
```
fig, (ax1, ax2) = plt.subplots(1,2)
gross_profits = [t[1]['gross_profit'] for t in populous_profits]
pl_ratios = [t[1]['PL_ratio'] for t in populous_profits]

fig.suptitle('Satistical Profits by Genre: ',fontsize=28)
plt.subplots_adjust(wspace=.01)
fig.set_figheight(12)
fig.set_figwidth(20)

ax1.boxplot(pl_ratios,vert=False)
ax1.set_yticklabels([t[0][:-6] for t in populous_profits],fontsize=18)
ax1.set_xlim(-3,10)

ax1.set_title("P/L ratio",fontsize=22)
ax1.grid(axis='y')

################################################################################

ax2.boxplot(gross_profits,vert=False)
ax2.set_yticklabels([],fontsize=20)
ax2.set_xlim(-250000000,500000000)

ax2.set_title("gross profit",fontsize=22)
ax2.grid(axis='y')

fig.set_figheight(10)
fig.set_figwidth(15)

sns.set_theme()

plt.show()
```



If getting in the green is the priority, thrillers is the bottom line.

- for prioritizing gross profit, consider horror first followed by war.
- for prioritizing ROI, horror is again the clear leader with war right behind.
- Mystery deserves an honerable mention for the greatest maximum.

In [32]:
```
fig, (ax1, ax2) = plt.subplots(1,2)
popularities = [t[1]['popularity'] for t in populous_appeals]
avg_ratings = [t[1]['average_rating'] for t in populous_appeals]

fig.suptitle('Satistical Appeal by Genre: ',fontsize=22)
plt.subplots_adjust(wspace=.01)
fig.set_figheight(12)
fig.set_figwidth(20)

ax1.boxplot(popularities,vert=False)
ax1.set_yticklabels([t[0][:-7] for t in populous_appeals],fontsize=16)
ax1.set_xlim(0,17)

ax1.set_title("tmdb popularity score",fontsize=18)
ax1.grid(axis='y')

#############################################################################

ax2.boxplot(avg_ratings,vert=False)
ax2.set_yticklabels([],fontsize=18)

ax2.set_title("average viewer rating",fontsize=18)
ax2.grid(axis='y')

fig.set_figheight(9)
fig.set_figwidth(12)

sns.set_style("whitegrid")

plt.show()
```



If developing a brand is the goal, it depends on the time-scale you're looking at....

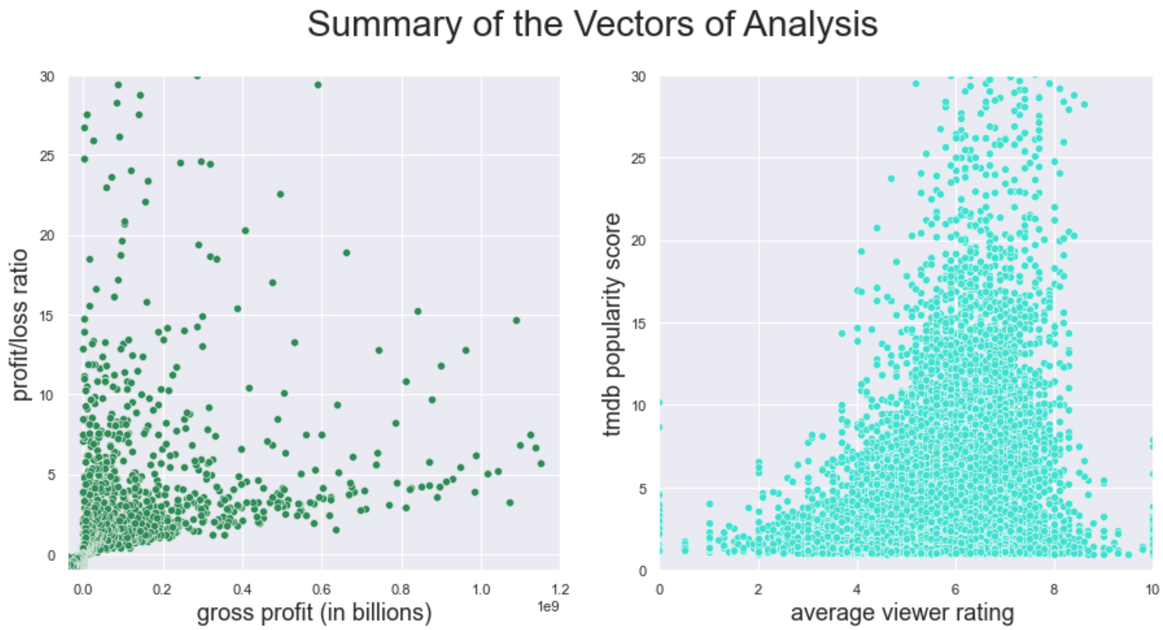
- if the objective is to create a brand or creative universe (e.g. MCU), the popularity score is the best metric because it measures public appeal on a variety of vectors some of which are measured over time (e.g. frequency of title in a search engine is updated regularly). So it can be infered from the chart above that -in order- *drama, history, and war* tend to be better at maintaining attention in the long run.
- However, -in order- fantasy, history and drama clearly show generally greater average viewer ratings, which basically means in those are the genres most likely to have a great public appeal at time of release, but not necessarily maintain a fanbase over time.

Drama and history both occur in the top 3 genres based on the public appeal measures, meaning they will probably be well received either way. Fantasy is the obvious leader in terms of viewer ratings so this genre should be strongly considered for the sake of making a great first impression as MS Studios enters this space.

In [28]:

```
fig,ax = plt.subplots(1,2)
# ax1 = plt.subplot2grid((2, 2), (0, 0))
ax2 = plt.subplot2grid((1,2), (0,0)) #
ax3 = plt.subplot2grid((1,2),(0,1)) #
# ax4 = plt.subplot2grid((2, 2), (1, 1))

# for t in populous_appeals:
#     g = sns.regplot(data=t[1],x='average_rating',y='popularity',
#                       label=t[0][:-7],ax=ax1)

for t in populous_profits:
    k = sns.scatterplot(data=t[1],y='PL_ratio',x='gross_profit',
                      label=t[0][:-6],ax=ax2,color='seagreen')

for t in monthly_appeals:
    p = sns.scatterplot(data=t[1],x='average_rating',y='popularity',
                   label=months[t[0]-1],ax=ax3,color='turquoise')

# for t in monthly_quants:
#     q = sns.regplot(data=t[1],y='PL_ratio',x='gross_profit',
#                    label=months[t[0]-1],ax=ax4)

fig.set_figheight(7)
fig.set_figwidth(15)
fig.suptitle('Summary of the Vectors of Analysis',fontsize=28)

# ax1.set_title("average viewer rating(x) vs tmdb popularity score(y)",fontsi
# ax1.set_xlabel('average viewer rating ',fontsize=18)
# ax1.set_ylabel('tmdb popularity score',fontsize=18)
# ax1.set_ylim(0,30)
# ax1.set_xlim(0,10)

# ax2.set_title("public appeal",fontsize=22)
ax2.set_xlabel('gross profit (in billions)',fontsize=18)
ax2.set_ylabel('profit/loss ratio',fontsize=18)
ax2.set_ylim(-1,30)
ax2.set_xlim(-40000000,1200000000)
ax2.legend().set_visible(False)

# ax3.set_title("profitability",fontsize=22)
ax3.set_xlabel('average viewer rating ',fontsize=18)
ax3.set_ylabel('tmdb popularity score',fontsize=18)
ax3.set_ylim(0,30)
ax3.set_xlim(0,10)
ax3.legend().set_visible(False)

# ax4.set_title("gross profit(x) vs profit/loss ratio(y)",fontsize=26)
# ax4.set_xlabel('gross profit (in billions)',fontsize=18)
# ax4.set_ylabel('profit/loss ratio',fontsize=18)
# ax4.set_ylim(-1,30)
# ax4.set_xlim(-40000000,1200000000)

# ax1.legend(loc='upper left',prop={'size':15})
# ax2.legend(loc='upper right',prop={'size':15})
# ax3.legend(loc='upper left',prop={'size':15})
# ax4.legend(loc='upper right',prop={'size':15})
```

```
sns.set_theme()

plt.show()
```



We see here that regardless of of wether we analyze in regard to the time of year to release or genre to release in, regardless of by which feature we measure performance, the data behaves consistently across all vectors of analyses. So loosely speaking it doesn't really matter which measure is used, so long as one is used consistently and with respect to its featuresm it will probably inform good decision making. The *"which"* is only relevant so far as there is a specific business goal in mind.

Conclusion

The data suggest that in order to maximize financial productivity, it is best to produce a thriller, western or animation and release mid-summer or mid-winter. To narrow further consider the specific financial needs of the studio. When prioritizing gross profit, consider horror or war, or if ROI is the main concern, while it returns marginaly less on average than horror and war, mystery shows the highest potential returns.

In regard to branding, if there is a long-term vision to develop a series or creative universe like the MCU or Star Wars, the tmdb popularity score is probably a more indicated meausrument as it considers how much a film is *talked about* in addition to viewer ratings, and tracks several metrics over time, so popularity is an indication of how lasting of an impression a film makes in additio to how well its received upon release making it a useful guide when the primary concern is long-term creative development. According to the analyses of popularity December is by far the best month to release a film in, july and august rank well also; drama and history or likely to build the greatest popularity, war is a strong consideration as well.

In the case that the goal is make a real entrance into the industry with a jaw-dropping first impression, being ambivalent to wether the movie will be talked about years later, average rating is the clear metric to use as it is a direct measurement of how viewers rated a movie. The analyses of average ratings sugest that the best time of year to release is April or November, and the most appealing genres to viewers is -in order- fantasy, history and drama.

Next steps

This analyses is a broad assesment of the topics at hand, it is agnostic to the year that a title was released, and the medium that it was released on (i.e. we are not filtering for streaming vs theatres), and considers only release date and not the length of time spent on production. These factors shoud be explored in future analyses before finalizing any executive level decisions.