Investigate_a_Dataset

November 24, 2018

```
<img src="Source/header.jpg"  alt="Hedaar" height="400" width="800">
```

Project: Investigate a Dataset : TMDb movie data

0.1 Table of Contents

Introduction
Projects questions
Data Wrangling
Exploratory Data Analysis
Conclusions
Introduction
Project Overview :

This project focus on data analysis life cycle. From gathering the data to communicate the result. This analysis uses many helpful Python libraries such as pandas, NumPy, and Matblotlib to make the analysis easier.

About the dataset :

```
In [2]: # Data geathering
        ## 1) Import the packages.
        import numpy as np  # useful for many scientific computing in Python
        import pandas as pd # primary data structure library


        import matplotlib.pyplot as plt #load this library for ploting
        %matplotlib inline
```

```
In [3]: # 2) Reading the data file.
        Movies = pd.read_csv('Data/tmdb-movies.csv')
```

```
In [94]: # Data assessing
         # 1) Retrieve the first 5 rows to understand the data.
         Movies.head()
```

```
Out[94]:        id    imdb_id  popularity     budget     revenue  \
         0  135397  tt0369610   32.985763  150000000  1513528810
         1   76341  tt1392190   28.419936  150000000   378436354
```

```
2  262500  tt2908446   13.112507  110000000   295238201
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360

               original_title  \
0               Jurassic World
1           Mad Max: Fury Road
2                     Insurgent
3  Star Wars: The Force Awakens
4                     Furious 7

                                                cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...

                                      homepage          director  \
0                   http://www.jurassicworld.com/   Colin Trevorrow
1                     http://www.madmaxmovie.com/     George Miller
2    http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod...       J.J. Abrams
4                         http://www.furious7.com/         James Wan

                       tagline      ...       \
0             The park is open.      ...
1            What a Lovely Day.      ...
2    One Choice Can Destroy You      ...
3  Every generation has a story.     ...
4           Vengeance Hits Home      ...

                                          overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137

                                    genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                     Action|Crime|Thriller

                              production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
```

```
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3           Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

In [95]:
```
# 2) Retrieve the columns names to decide what is the columns I need in the project.
names = Movies.columns.values
names
```

Out[95]:
```
array(['id', 'imdb_id', 'popularity', 'budget', 'revenue',
       'original_title', 'cast', 'homepage', 'director', 'tagline',
       'keywords', 'overview', 'runtime', 'genres', 'production_companies',
       'release_date', 'vote_count', 'vote_average', 'release_year',
       'budget_adj', 'revenue_adj'], dtype=object)
```

In [4]:
```
Movies['revenue'].max()
```

Out[4]:
```
2781505847
```

In [96]:
```
# 3) Identify the shape of the dataset.
Movies.shape
```

Out[96]:
```
(10866, 21)
```

In [97]:
```
# 4)Identify the column type to decide if there are any changes required.
Movies.dtypes
```

Out[97]:
```
id                      int64
imdb_id                object
popularity            float64
budget                  int64
revenue                 int64
original_title         object
cast                   object
homepage               object
director               object
tagline                object
keywords               object
overview               object
runtime                 int64
```

```
genres                  object
production_companies    object
release_date            object
vote_count               int64
vote_average           float64
release_year             int64
budget_adj             float64
revenue_adj            float64
dtype: object
```

In [98]:
```
# 5) Unique values in each row.
Movies.nunique()
```

Out[98]:
```
id                     10865
imdb_id                10855
popularity             10814
budget                   557
revenue                 4702
original_title         10571
cast                   10719
homepage                2896
director                5067
tagline                 7997
keywords                8804
overview               10847
runtime                  247
genres                  2039
production_companies    7445
release_date            5909
vote_count              1289
vote_average              72
release_year              56
budget_adj              2614
revenue_adj             4840
dtype: int64
```

In [100]:
```
# 6) Calculate the null values in each column.
Movies.isnull().sum()
```

Out[100]:
```
id                        0
imdb_id                  10
popularity                0
budget                    0
revenue                   0
original_title            0
cast                     76
homepage               7930
director                 44
tagline                2824
```

```
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

In [4]:
```
# Data cleaning
#1) Q1: What are the columns I need in the project.
#I will drop unneeded column in the dataset.
Movies.drop(['id','imdb_id','budget','cast','homepage','production_companies','director'
                        ,'tagline','tagline','keywords','overview','runtime'
                       ,'release_date','vote_count','budget_adj','revenue_adj'],axis=1,in
Movies.head()
```

Out[4]:

	popularity	revenue	original_title	genres	vote_average	release_year
0	32.985763	1513528810	Jurassic World	Action|Adventure|Science Fiction|Thriller	6.5	2015
1	28.419936	378436354	Mad Max: Fury Road	Action|Adventure|Science Fiction|Thriller	7.1	2015
2	13.112507	295238201	Insurgent	Adventure|Science Fiction|Thriller	6.3	2015
3	11.173104	2068178225	Star Wars: The Force Awakens	Action|Adventure|Science Fiction|Fantasy	7.5	2015
4	9.335014	1506249360	Furious 7	Action|Crime|Thriller	7.3	2015

In [5]:
```
#Q2: Are there any changes required in these columns name?
# Yes I will change original_title by (movie_name) , release year by (year)
Movies.rename(columns={"original_title": "movie_name", "release_year": "year"}, inplace=
Movies.head()
```

Out[5]:

	popularity	revenue	movie_name	genres	vote_average	year
0	32.985763	1513528810	Jurassic World	Action|Adventure|Science Fiction|Thriller	6.5	2015
1	28.419936	378436354	Mad Max: Fury Road	Action|Adventure|Science Fiction|Thriller	7.1	2015
2	13.112507	295238201	Insurgent	Adventure|Science Fiction|Thriller	6.3	2015
3	11.173104	2068178225	Star Wars: The Force Awakens	Action|Adventure|Science Fiction|Fantasy	7.5	2015
4	9.335014	1506249360	Furious 7	Action|Crime|Thriller	7.3	2015

```
1  Action|Adventure|Science Fiction|Thriller          7.1  2015
2         Adventure|Science Fiction|Thriller          6.3  2015
3   Action|Adventure|Science Fiction|Fantasy          7.5  2015
4                     Action|Crime|Thriller          7.3  2015
```

In [103]:
```
#Q3: Are the data type of this column correct?
#a.Identify the column type to decide if there are any changes required.
Movies.dtypes

# There are no changes required here
```

Out[103]:
```
popularity      float64
revenue           int64
movie_name       object
genres           object
vote_average    float64
year              int64
dtype: object
```

In [104]:
```
#Q4: Are there a missing data in this column? If yes, how many?
Movies.isnull().sum()

#I will leave missing data processing at the end of the data cleaning process
```

Out[104]:
```
popularity       0
revenue          0
movie_name       0
genres          23
vote_average     0
year             0
dtype: int64
```

In [105]:
```
#Q5: Are there any duplicate rows? if yes, how many?
print(sum(Movies.duplicated()))

#drop it
Movies.drop_duplicates(inplace=True)
```

```
1
```

In [106]:
```
#Q6) Are there any outliers?
#Statistical description
Movies.describe()
```

Out[106]:

	popularity	revenue	vote_average	year
count	10865.000000	1.086500e+04	10865.000000	10865.000000
mean	0.646446	3.982690e+07	5.975012	2001.321859
std	1.000231	1.170083e+08	0.935138	12.813260

```
min         0.000065  0.000000e+00       1.500000    1960.000000
25%         0.207575  0.000000e+00       5.400000    1995.000000
50%         0.383831  0.000000e+00       6.000000    2006.000000
75%         0.713857  2.400000e+07       6.600000    2011.000000
max        32.985763  2.781506e+09       9.200000    2015.000000
```

The Popularity column contains a gap between the minimum and the maximum value. First of all, I thought there are outliers values. So, I asked myself many questions:1) What does the popularity of movies depend on?2) Are there any boundaries of its values?

Then I searched on the official website of the dataset and I found the answers.

1) What does the popularity of movies depend on?The popularity defined by the behavior of the users. Taking in account the number of ratings a movie received, the number of favorites and number of watched list additions all for the previous day. It also uses the part of the previous days score to help popularity trending and finally, boosts scores a tad if the newer a release date is. https://www.themoviedb.org/talk/5141d424760ee34da71431b0

2) Are there any boundaries of its values?The lower boundary is 0.0, and the upper is essentially infinity. https://www.themoviedb.org/talk/58b2df5392514177a8003e50

There are no outliers values in popularity

One of the issues with handling missing data when the column is not empty but does not contain the value. For example, there are a large number of zeros (6016) in the revenue. I thought maybe these Movies have legal problems that don't allow it to distribution. So, I take a row for the test.

```
In [107]: Movies['revenue'].loc[Movies['revenue'] == 0].count()

Out[107]: 6016
```

There is a Survivor movie I watched it at the cinema and I liked it. So there are no legal problems here. or maybe there but this not alone affects in the missing data.

```
In [108]: zero = Movies.loc[Movies['revenue'] == 0]
          zero.head()

Out[108]:     popularity  revenue                      movie_name  \
          48    2.932340        0                       Wild Card
          67    2.331636        0                        Survivor
          74    2.165433        0          Mythica: The Darkspore
          75    2.141506        0  Me and Earl and the Dying Girl
          92    1.876037        0        Mythica: The Necromancer

                              genres  vote_average  year
          48      Thriller|Crime|Drama           5.3  2015
          67     Crime|Thriller|Action           5.4  2015
          74  Action|Adventure|Fantasy           5.1  2015
          75              Comedy|Drama           7.7  2015
          92  Fantasy|Action|Adventure           5.4  2015
```

Find Helpful dataset:

I found the data set contain more than 40000 movies so this will increase the chance of finding the missing values in my dataset. data source (https://www.kaggle.com/rounakbanik/the-movies-dataset#movies_metadata.csv)

Read the dataset:

In [109]:
```
# 1) Read another dataset named Movies2
Movies2 = pd.read_csv('Data/movies_metadata.csv')
Movies2.head()
```

```
/opt/conda/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2785: DtypeWarning: Colu
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[109]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_date
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	1995-10-30
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	1995-12-15
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	...	1995-12-22
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	...	1995-12-22
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	...	1995-02-10

```
       revenue runtime                                     spoken_languages  \
0  373554033.0    81.0           [{'iso_639_1': 'en', 'name': 'English'}]
1  262797249.0   104.0  [{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2          0.0   101.0           [{'iso_639_1': 'en', 'name': 'English'}]
3   81452156.0   127.0           [{'iso_639_1': 'en', 'name': 'English'}]
4   76578911.0   106.0           [{'iso_639_1': 'en', 'name': 'English'}]

     status                                            tagline  \
0  Released                                                NaN
1  Released          Roll the dice and unleash the excitement!
2  Released  Still Yelling. Still Fighting. Still Ready for...
3  Released  Friends are the people who let you be yourself...
4  Released  Just When His World Is Back To Normal... He's ...

                         title  video vote_average vote_count
0                    Toy Story  False          7.7     5415.0
1                      Jumanji  False          6.9     2413.0
2             Grumpier Old Men  False          6.5       92.0
3            Waiting to Exhale  False          6.1       34.0
4  Father of the Bride Part II  False          5.7      173.0

[5 rows x 24 columns]
```

In [110]:
```
#Manager function
def fill_miss_data(set1,col1,col2,set2,col3,col4):
    # Call prepare_sets function and send to it parameters
    #(Movies,movie_name,production_companies,Movies2,original_title,production_compani
    Null_set,Filled_set = prepare_sets(set1,col1,col2,set2,col3,col4)

    # Check if the filled_set have missing value. if any drop it
    Check_missing_zero(Filled_set)

    #Fill missing data
    fill_missing(set1,col1,col2,Filled_set,col3,col4)
```

In [111]:
```
def prepare_sets(set1,col1,col2,set2,col3,col4):

    #this is the useful link ( https://www.youtube.com/watch?v=2AFGPdNn4FM )
    #1) Create the data frame containing Movie name and production companies from both
    # My set -> Movie_subset
    Movie_subset = pd.DataFrame(set1, columns = [col1 , col2])

    # Another set -> Movie2_set
    Movie2_set = pd.DataFrame(Movies2, columns = [col3 , col4])


    #2) Create the data frame containing the missing values rows.
```

```
        Null_set = Movie_subset[Movie_subset[col2].isnull()]


        #3) Create a boolean series that contain a true and false value of this condition
        #( movie2 move name [isin] the null set)
        # the purpose of this step is making a filter to use it in the large dataset
        boole_series = Movie2_set[col3].isin(Null_set[col1])


        #4) Extract the data that match boolean series = true
        Filled_set = Movie2_set[boole_series]
        return Null_set,Filled_set
```

In [112]:
```
def Check_missing_zero(dataset):

    # There is a problem here the null values dos'nt counted so I will set 0s as nan
    dataset['revenue'].replace({0: np.nan},inplace=True)


    # Drop duplicated
    drop_set = dataset.dropna(subset=['revenue'], how='all',inplace=True)
    num = dataset['revenue'].isnull().sum()


    return drop_set
```

In [113]:
```
def fill_missing(set1,col1,col2,set2,col3,col4):
    #Fill missing data
    for i,value in set2[col3].iteritems():
        for g,p in set1[col1].iteritems():
             if value == p:
                    set1[col2].loc[g] = set2[col4].loc[i]

    return print("The data set missing value successfully filled")
```

In [114]:
```
print("  ------------------------------------------")
print("|         Handle Revenue Missing Value     |")
print("  ------------------------------------------")

# #Replace 0s by NaN
Movies['revenue'].replace({0: np.nan},inplace=True)


# Fill missing data in revenue column
number_befor = Movies['revenue'].isnull().sum()
fill_miss_data(Movies,'movie_name','revenue',Movies2,'original_title','revenue')

number_after = Movies['revenue'].isnull().sum()
```

```
          print("The missing values before filling =",number_befor
                ,'\n The missing values after filling = ',number_after
                ,'\n The filled rows ',number_befor-number_after)

  ------------------------------------------
|         Handle Revenue Missing Value     |
  ------------------------------------------


/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self._setitem_with_indexer(indexer, value)


The data set missing value successfully filled
The missing values before filling = 6016
 The missing values after filling =  5600
 The filled rows  416


In [115]: # After this process, I will check how many rows have missing data to decide about the
          Movies.isnull().sum()

Out[115]: popularity         0
          revenue         5600
          movie_name         0
          genres            23
          vote_average       0
          year               0
          dtype: int64

In [116]: # I decided to use the mean approach because I don't want to lose the data in my analy
          #Calculate the mean
          mean = Movies['revenue'].mean()
          print("Mean is : \n",mean)

          #Replace 0s by mean
          Movies['revenue'].replace({np.nan: mean},inplace=True)

Mean is :
 87069004.6784


In [117]: #Check if it works
          total_missing = Movies.isnull().values.ravel().sum()
          print(10866 - total_missing)
          print(Movies.isnull().sum())
```

```
10843
popularity       0
revenue          0
movie_name       0
genres          23
vote_average     0
year             0
dtype: int64
```

```
In [118]: # The null values equal to 868 now I will just drop it
          Movies.dropna(inplace=True)
          print(Movies.isnull().sum())
```

```
popularity      0
revenue         0
movie_name      0
genres          0
vote_average    0
year            0
dtype: int64
```

```
In [119]: #Final shape of my data
          Movies.shape
```

```
Out[119]: (10842, 6)
```

Exploratory Data Analysis

> **Tip**: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

During the exploration, I face a new problem. There are duplicate movies. The chart below shows this problem. So I will return to the cleaning step to handle this problem. I decided to keep the first row and drop the others

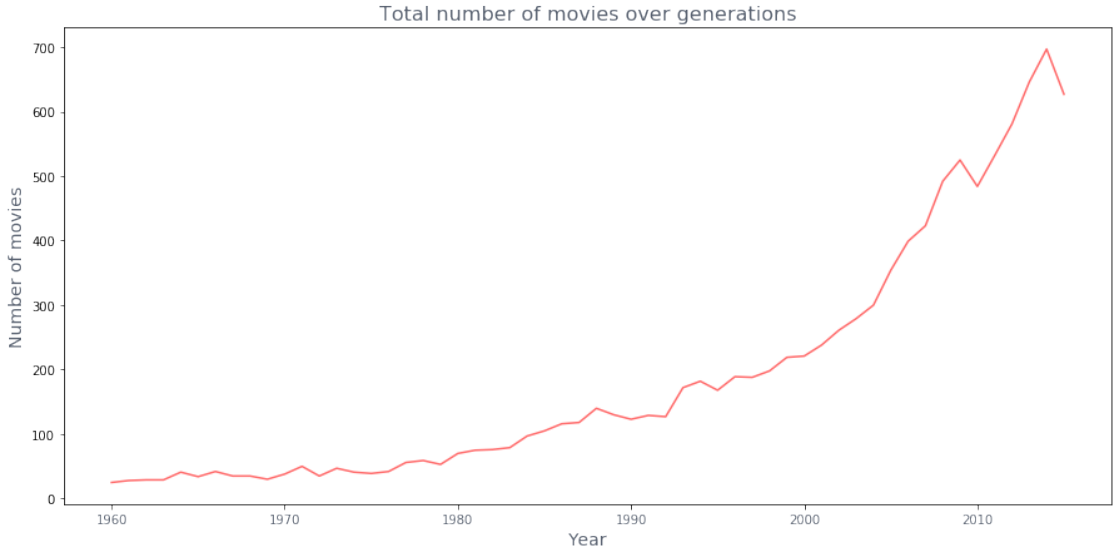
```
In [120]: #Cleaning duplication of the Movies name
          # I decided to keep the first row and drop the others
          Movies.drop_duplicates(subset='movie_name', keep="first",inplace=True)

          #test if it works
          Movies.sort_values(by='revenue',ascending=False).head()
```

```
Out[120]:       popularity       revenue                    movie_name  \
          1386    9.432768  2.781506e+09                        Avatar
```

```
3       11.173104  2.068178e+09  Star Wars: The Force Awakens
4361     7.637767  1.519558e+09                   The Avengers
0       32.985763  1.513529e+09                 Jurassic World
4        9.335014  1.506249e+09                      Furious 7

                                              genres  vote_average  year
1386   Action|Adventure|Fantasy|Science Fiction           7.1  2009
3      Action|Adventure|Science Fiction|Fantasy           7.5  2015
4361           Science Fiction|Action|Adventure           7.3  2012
0     Action|Adventure|Science Fiction|Thriller           6.5  2015
4                        Action|Crime|Thriller           7.3  2015
```

General questions:1. What is the trend of the total number of movies over the generations?2. What's the revenue trend over the generations?

```
In [121]: # Plot total number of movies by generations
          Movies_per_year = Movies.groupby(['year']).count()['movie_name']

          plt.figure(figsize=(15,7))
          plt.plot(Movies_per_year,color="#ff6b6b")
          plt.title('Total number of movies over generations',fontsize=16,color='#57606f')
          plt.xlabel('Year',fontsize=14,color='#57606f')
          plt.ylabel('Number of movies',fontsize=14,color='#57606f')
          plt.xticks(color='#57606f')
          plt.show()
```



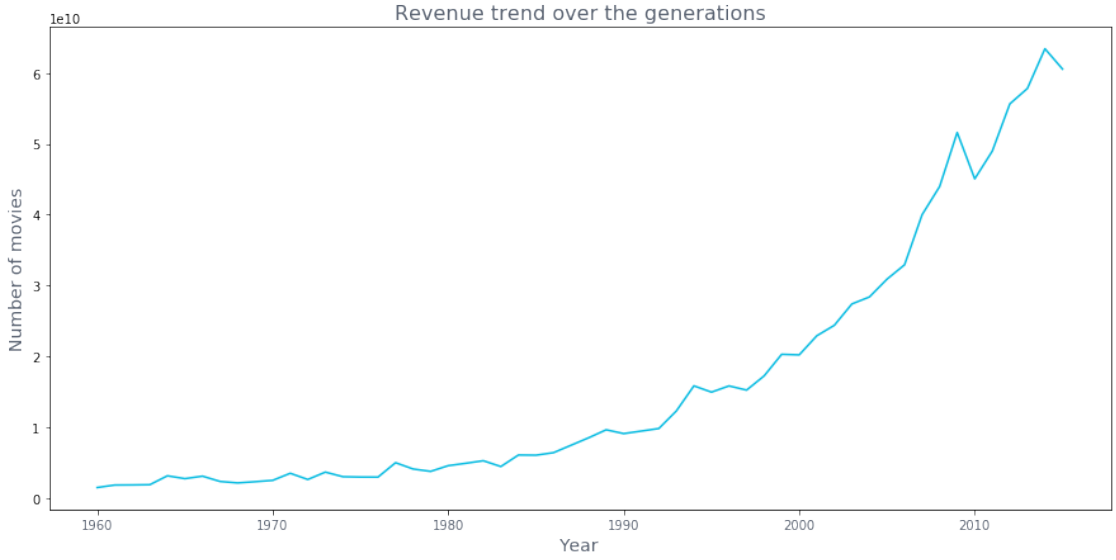
The number of movies increases by generations. I think this relationship is logical. Almost all industries are improving over the years. The reason for this is the technology revolution which is inflated flexibility of work and quality and creates new channels. In the past, movies need a great

deal of effort to produce the black, white and silent film (this amazing movie have a story about it https://www.themoviedb.org/movie/44826-hugo?language=en-US). Now, you can see a lot of movies across your tablet set on your bed.

There is an important point in the chart there is a decrease in the number of films in 2010. One of the possible reasons is the dataset have not enough data this year.

What's the revenue trend over the generations?

```
In [122]: #What's the revenue trend over the ?
          revenue_year = Movies.groupby(['year']).sum()['revenue']

          plt.figure(figsize=(15,7))
          plt.plot(revenue_year,color='#0abde3')
          plt.title('Revenue trend over the generations',fontsize=16,color='#57606f')
          plt.xlabel('Year',fontsize=14,color='#57606f')
          plt.ylabel('Number of movies',fontsize=14,color='#57606f')
          plt.xticks(color='#57606f')
          plt.show()
```



Increase revenues over generations. This relationship is also logical. Because the number of movies also increases.

What are properties that associated with high revenue Movies?1. What are the top 20 Movies based on revenue?2. High movies revenue Vs popularity.3. High revenue Movies Vs voting rate.

```
In [7]: Movies['revenue'].max()

Out[7]: 2781505847
```

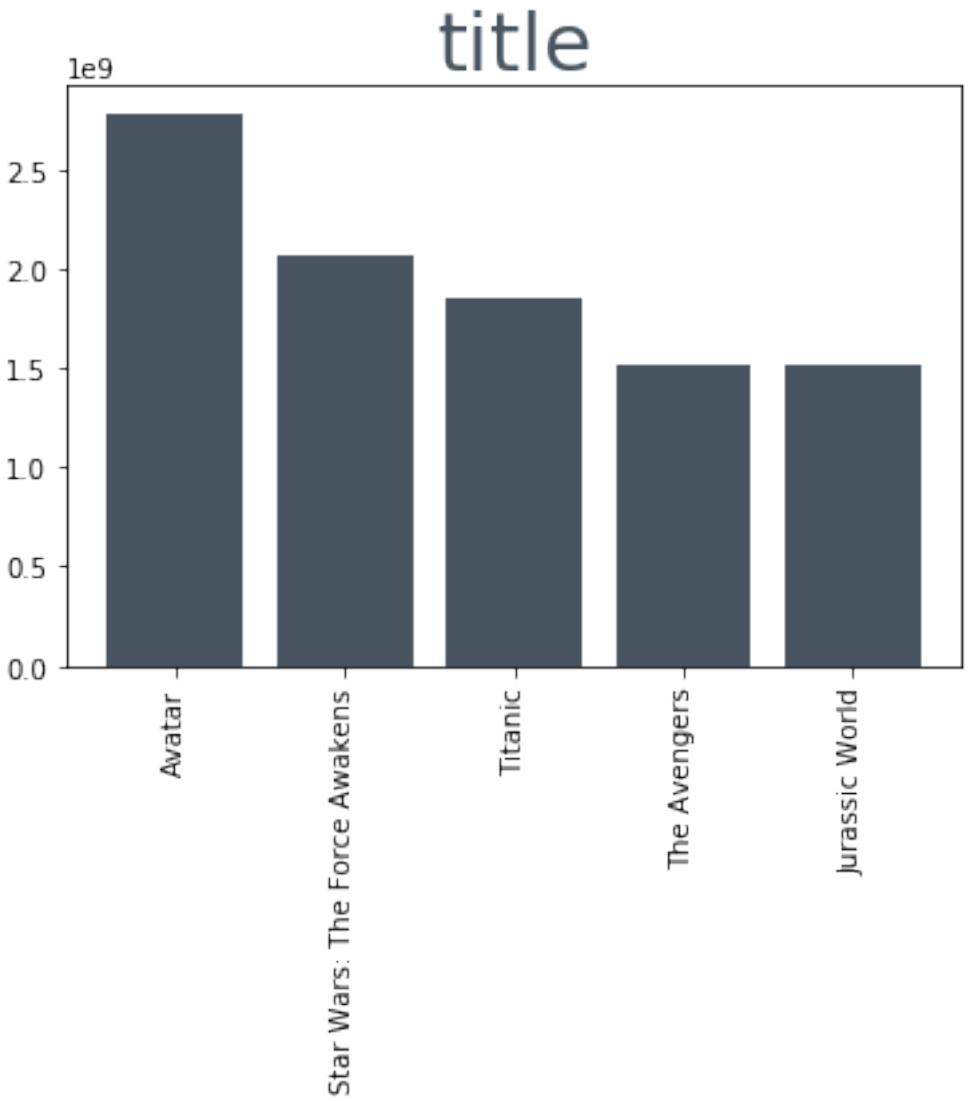
In [73]:
```
#What is the top 20 movies based on revenue?
# useful link (http://cmdlinetips.com/2018/02/how-to-sort-pandas-dataframe-by-columns-a

#Subset of top 20 Movies based on revenue
Revenue_top = Movies.sort_values(by='revenue',ascending=False).head()
Revenue_top.head()
a = Revenue_top[['original_title', 'revenue']]
a
```

Out[73]:

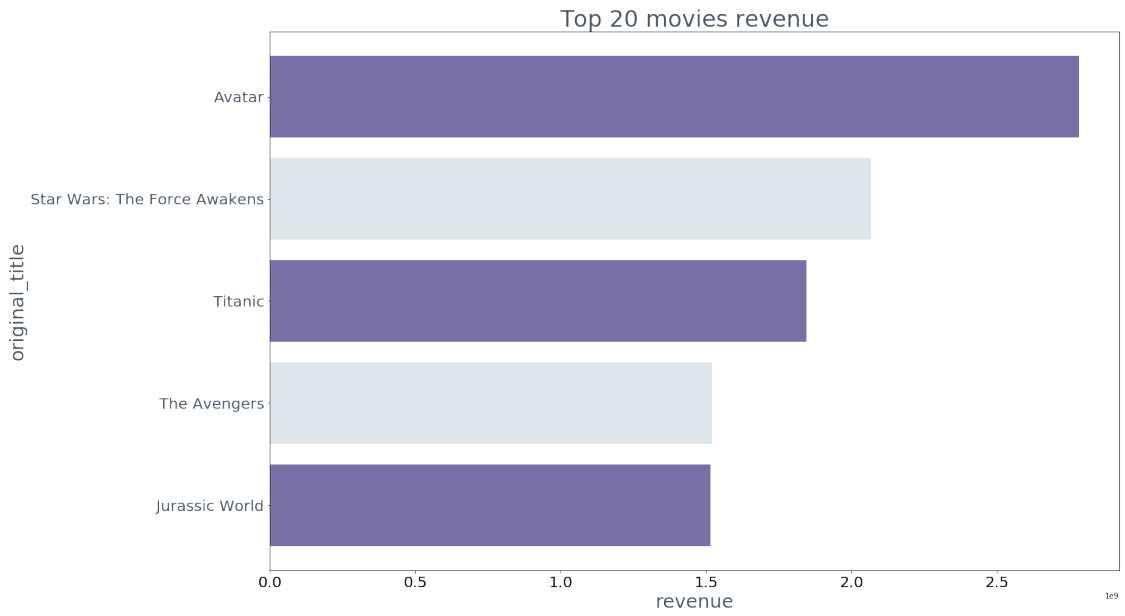
	original_title	revenue
1386	Avatar	2781505847
3	Star Wars: The Force Awakens	2068178225
5231	Titanic	1845034188
4361	The Avengers	1519557910
0	Jurassic World	1513528810

In [61]:
```
# This function plot a bar graph by sending columns and the figure colors
def plot_bar(dataset,col1,col2,title,color1,color2):

    #sort values
    df_c = dataset.sort_values(col2)

    #Figure size
    plt.figure(figsize=(20,13))

    #Bar plot
    x = df_c[col2]
    y = range(len(df_c[col2]))
    plt.barh(y,x, align='center',
            color=[color1,color2])

    #Bar plot properties
    plt.title(title,fontsize=30,color='#485460')

    #Labels
    plt.xlabel(col2,fontsize=25,color='#485460')
    plt.ylabel(col1,fontsize=25,color='#485460')
    plt.yticks(np.arange(len(df_c[col1])),df_c[col1],color='#485460',fontsize=20)
    plt.xticks(fontsize=20)
    plt.show()
```

In [81]:
```
x = range(len(a['original_title']))
y = a['revenue']
plt.bar(x,y, align='center',color=['#485460'])
plt.title("title",fontsize=30,color='#485460')

plt.xticks(np.arange(len(a['original_title'])),a['original_title'],rotation=90)
plt.show()
```



In [63]:
```
# Plot a bar chart display top 20 Movies name
# you send the dataset,column1(y),column2(x),title of the graph,color1,color2

# I use 2 colors to make the graph more readable
plot_bar(a,'original_title','revenue','Top 20 movies revenue','#786fa6','#dfe6e9')
```



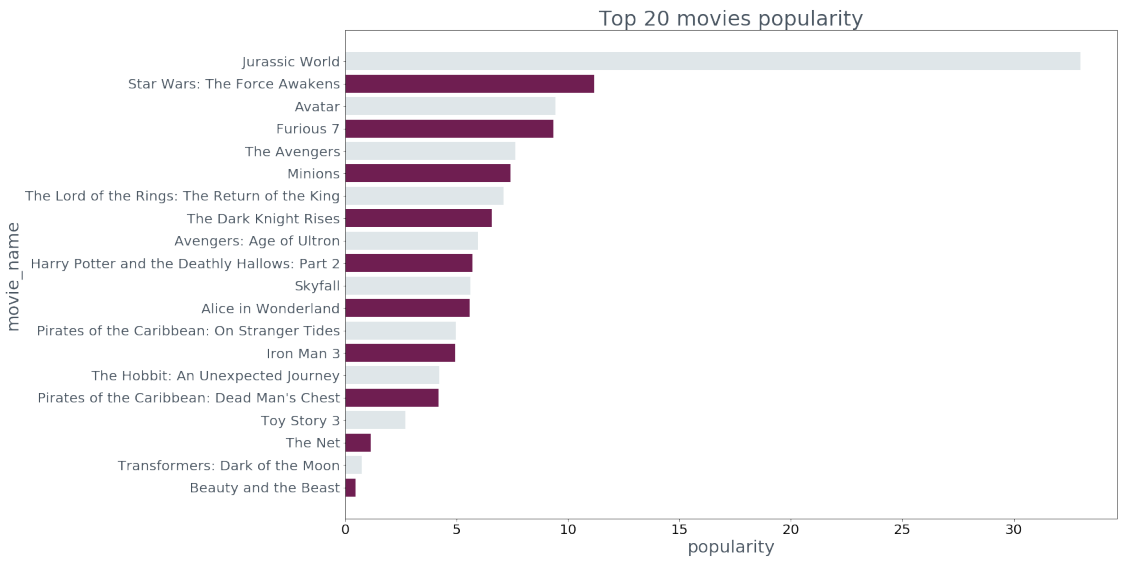
Avatar movie has the most revenue. The movie made extensive use of new motion capture filming techniques and was released for traditional viewing, 3D viewing (using the RealD 3D, Dolby 3D, XpanD 3D, and IMAX 3D formats). https://en.wikipedia.org/wiki/Avatar_(2009_film)

Star Wars: The Force Awakens
It comes after Avatar with remarkably amount between it and the rest

- The Avengers
- Jurassic World
- Farios7

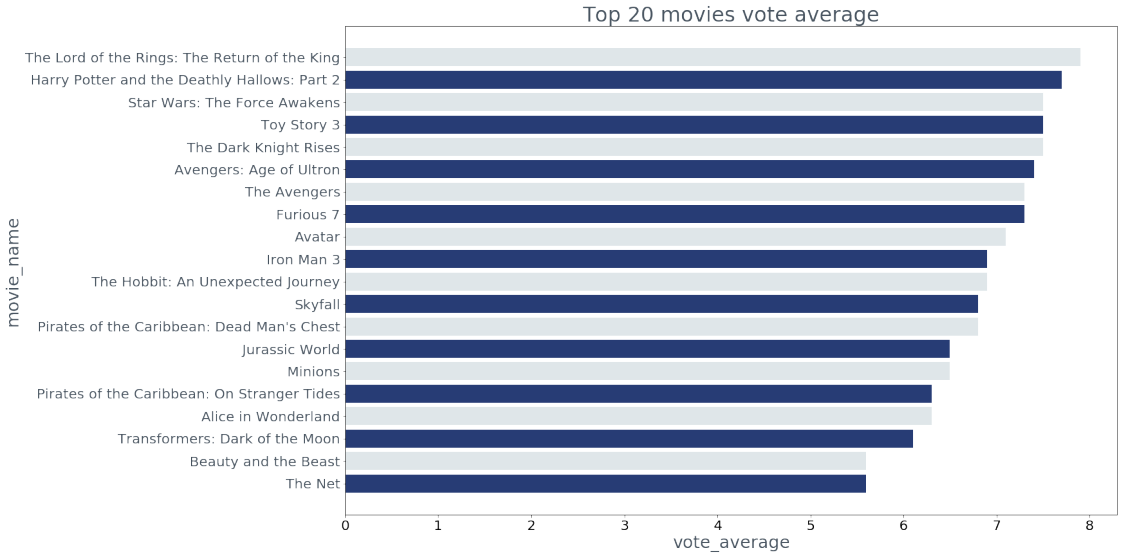
Revenue is close to each other.

```
In [126]: #High movies revenue Vs popularity
          plot_bar(Revenue_top,'movie_name','popularity','Top 20 movies popularity','#6F1E51','#
```



Revenue is not necessarily affected by popularity. Because the site calculates its own audience. There are multi-website that does the same thing as IMDB and rotten tomatoes. Also what if the movie was added after the show by the long time the user maybe don't visit the page or favorite it. If this scenario doesn't appear maybe the users favorite the movie as a part of there memory or a preferable movie.

```
In [127]: #High revenue Movies Vs voting rate.
          plot_bar(Revenue_top,'movie_name','vote_average','Top 20 movies vote average','#273c75
```



The average votes for the best 20 movies are higher than 5 points which is more than the median point that mean good scores. In general, the opinions of website users matches the real

audience on cinema. These matches create the relation between voting rate and revenue. but this not necessary in all cases.

The lord of the rings: the return of the king have the highest voting rate with almost 8 points.

- Beauty and the beast
- The net

Have the minimum average vote value, and almost equals each other.

Which genres are the most popular over the generations?

In [6]:
```
#Q1) Which genres are the most popular over the generations??

        # Extract genres list
        # This is a useful links
        # https://stackoverflow.com/questions/41190422/pandas-split-a-column-on-delimiter-and-ge
        # https://stackoverflow.com/questions/19392226/attributeerror-dataframe-object-has-no-at
        # https://medium.com/@rtjeannier/pandas-101-cont-9d061cb73bfc


        split_genrese = Movies['genres'].str.split('|', expand=True)
        genres = pd.unique(split_genrese.stack())

        year = Movies['year'].unique()
        year.sort(axis=0)
        print(year)

        # Count the genres frequancy
        count = pd.value_counts(split_genrese.values.flatten())

        # Greate now I should sum the popularity for each genres grouped it by year
        genre_popularity = pd.DataFrame({'genre': genres,'2010':0,'2011':0,'2012':0,'2013':0,'20
        print(type(genre_popularity))
        print(genres)
```

```
[1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974
 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004
 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015]
<class 'pandas.core.frame.DataFrame'>
['Action' 'Adventure' 'Science Fiction' 'Thriller' 'Fantasy' 'Crime'
 'Western' 'Drama' 'Family' 'Animation' 'Comedy' 'Mystery' 'Romance' 'War'
 'History' 'Music' 'Horror' 'Documentary' 'TV Movie' 'Foreign']
```

In [129]:
```
# Useful link (https://medium.com/@rtjeannier/pandas-101-cont-9d061cb73bfc)

          #Fill the popularity fo each genry
```

```
genre_popularity = pd.DataFrame({'genre': genres,1960:0,1970:0,1980:0,1990:0,2000:0,20
```

```
for i,value in Movies['genres'].iteritems():
    for g,p in genre_popularity['genre'].iteritems():
        if value == p:
            if  (Movies['year'].loc[i] >= 1960) and (Movies['year'].loc[i] <= 1969):
                genre_popularity[1960].loc[g] += Movies['popularity'].loc[i]

            elif  (Movies['year'].loc[i] >= 1970) and (Movies['year'].loc[i] <= 1979):
                genre_popularity[1970].loc[g] += Movies['popularity'].loc[i]

            elif  (Movies['year'].loc[i] >= 1980) and (Movies['year'].loc[i] <= 1989):
                genre_popularity[1980].loc[g] += Movies['popularity'].loc[i]

            elif  (Movies['year'].loc[i] >= 1990) and (Movies['year'].loc[i] <= 1999):
                genre_popularity[1990].loc[g] += Movies['popularity'].loc[i]

            elif  (Movies['year'].loc[i] >= 2000) and (Movies['year'].loc[i] <= 2009):
                genre_popularity[2000].loc[g] += Movies['popularity'].loc[i]

            elif  (Movies['year'].loc[i] >= 2010) and (Movies['year'].loc[i] <= 2019):
                genre_popularity[2010].loc[g] += Movies['popularity'].loc[i]
```

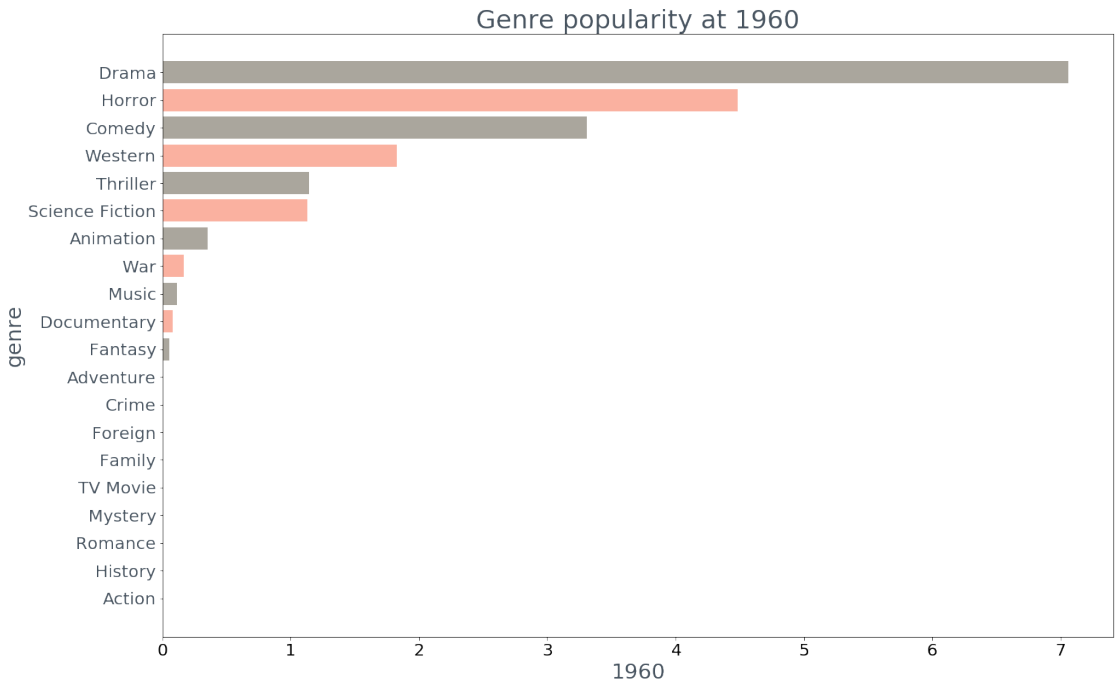
```
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self._setitem_with_indexer(indexer, value)
```

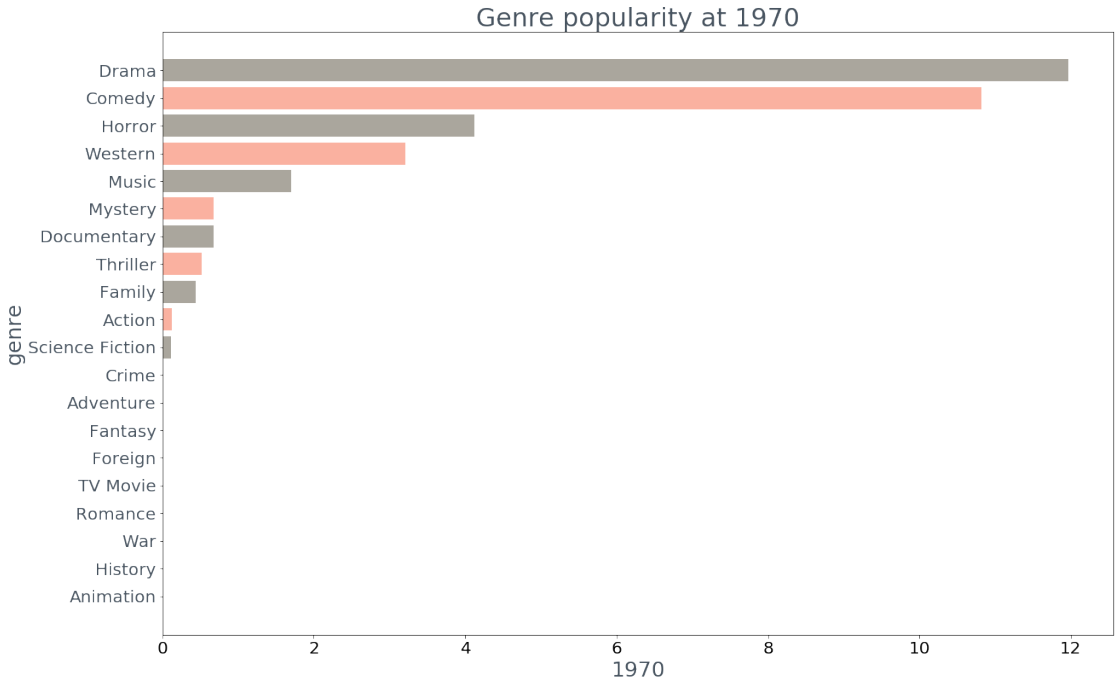
```
In [130]: genre_popularity.head()
```

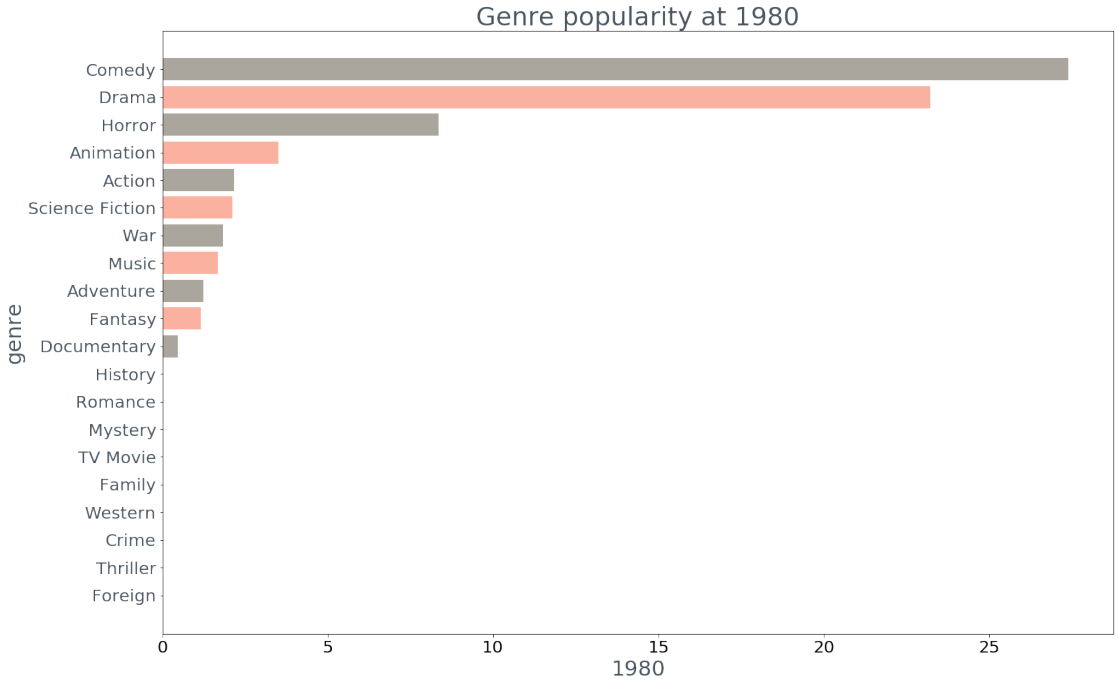
Out[130]:

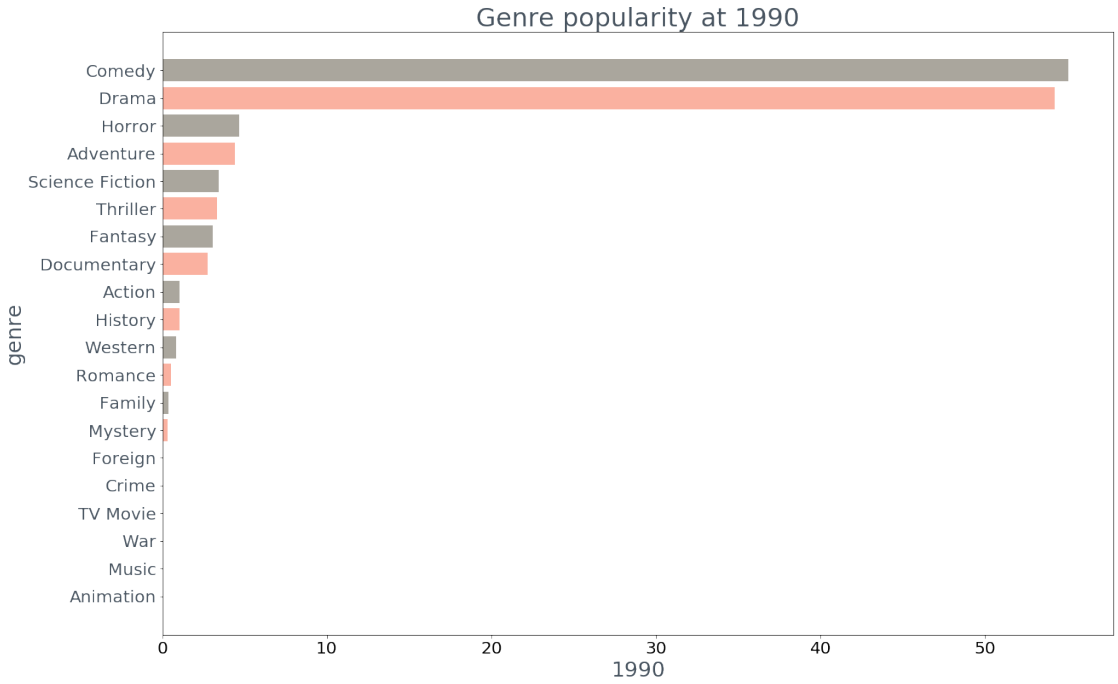
	genre	1960	1970	1980	1990	2000	2010
0	Action	0.000000	0.126723	2.182794	1.063002	14.602510	19.294962
1	Adventure	0.000000	0.000000	1.247333	4.422967	3.083300	5.535303
2	Science Fiction	1.131402	0.114062	2.120826	3.447578	2.228401	3.591147
3	Thriller	1.143560	0.522601	0.000000	3.326565	7.227164	40.541297
4	Fantasy	0.057243	0.000000	1.178524	3.062771	0.475826	0.911367

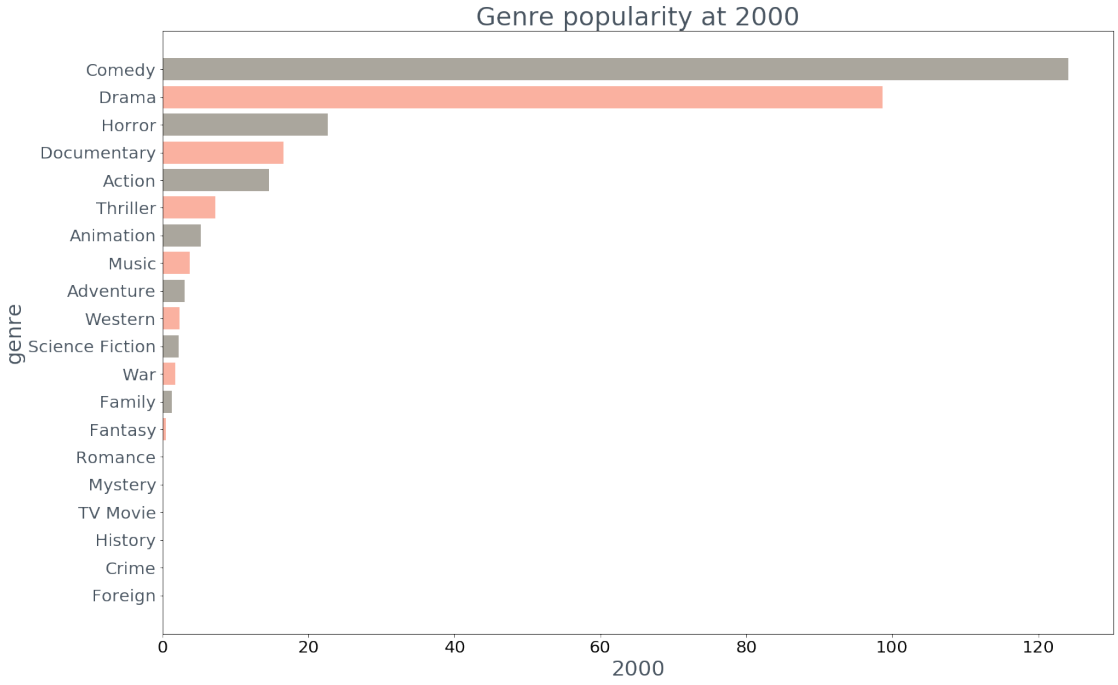
In [133]:
```
#Frist I will take the column name
x_ax = list(genre_popularity.columns.values)

# Then remove genre
x_ax.remove("genre")


#for loop to plot

title = "Genre popularity at "
for x in x_ax:
    plot_bar(genre_popularity,'genre',x,title+str(x),'#fab1a0','#aaa69d')
```

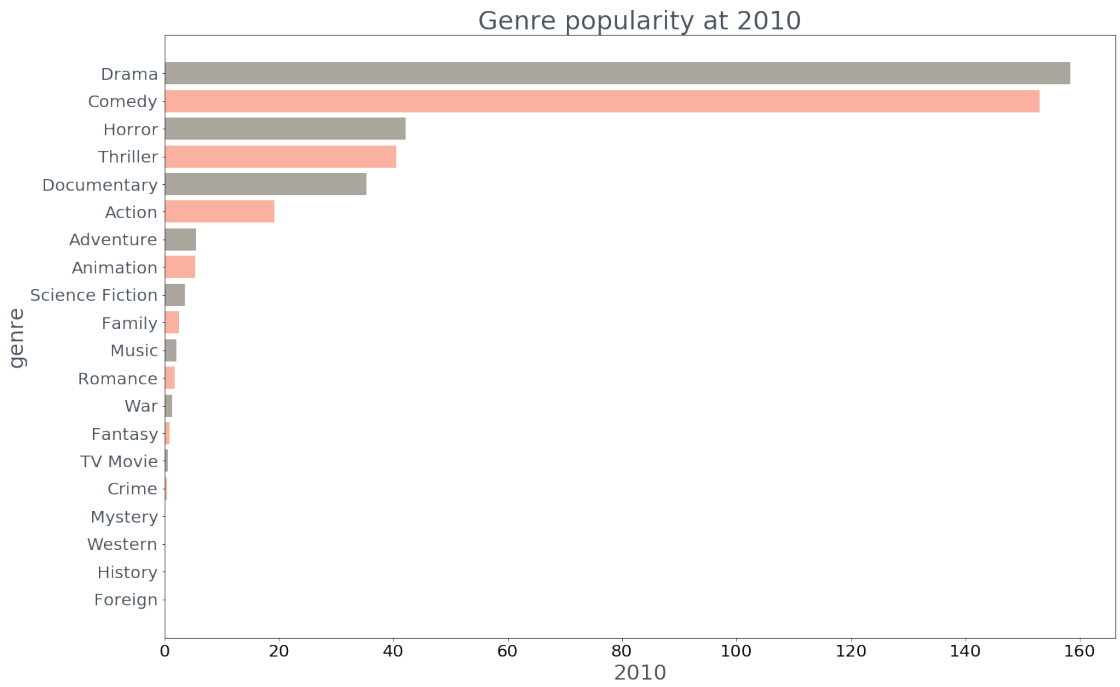



Genre popularity at 1970
Drama
Comedy
Horror
Western
Music
Mystery
Documentary
Thriller
Family
Action
Science Fiction
Crime
Adventure
Fantasy
Foreign
TV Movie
Romance
War
History
Animation
genre
0
2
4
6
8
10
12
1970


Genre popularity at 1980
Comedy
Drama
Horror
Animation
Action
Science Fiction
War
Music
Adventure
Fantasy
Documentary
History
Romance
Mystery
TV Movie
Family
Western
Crime
Thriller
Foreign
genre
0
5
10
15
20
25
1980


Genre popularity at 1990
Comedy
Drama
Horror
Adventure
Science Fiction
Thriller
Fantasy
Documentary
Action
History
Western
Romance
Family
Mystery
Foreign
Crime
TV Movie
War
Music
Animation
genre
0
10
20
30
40
50
1990


Genre popularity at 2000
Comedy
Drama
Horror
Documentary
Action
Thriller
Animation
Music
Adventure
Western
Science Fiction
War
Family
Fantasy
Romance
Mystery
TV Movie
History
Crime
Foreign
genre
0
20
40
60
80
100
120
2000



There are changes in genes over generations. The best three genes during these generations were drama, comedy, and horror with a different arrangement from one to the other. Drama movies dominate the 60s,70s, and 80s. While the comedy dominates the 90s and 2000s.

The comedy movies gain popularity during the generations. in the 60s it was the third then it became second in the 70s and 80s until it reached the first place in 90s and 2000s. But it fell in the 2010s again. But why? remember that 2010 affected by some reason and the number of movies decreased so maybe this effect on it.

The horror movies placed at third after the 70s and still there until 2010s.

Conclusions

```
In [132]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[132]: 0
```

```
 <img src="Source/footer.jpg"  alt="Hedaar" height="400" width="800">
```

```
In [ ]:
```