UCSD-Data-Science-MicroMasters (/github/mpothier/UCSD-Data-Science-MicroMasters/tree/master)
/ 1_Python_for_Data_Science (/github/mpothier/UCSD-Data-Science-MicroMasters/tree/master/1_Python_for_Data_Science)
/
Mini-Project_Movie-Genres (/github/mpothier/UCSD-Data-Science-MicroMasters/tree/master/1_Python_for_Data_Science/Mini-Project_Movie-Genres)

# Exploring Movie Genres and Ratings Over Time

### Questions to Investigate:

- How has the popularity of genres evolved over time?
- More specifically, what has been the percentage distribution among movies throughout the years?
- Does genre popularity correlate to particular decades or identifiable cultural eras?

This initial approach will only seek correlations between the percent *frequency* of movies of each genre in each year.
*****n.b. some results may appear skewed due to movies being tagged with multiple genres, as well as limited movie data in certain years*****

Additionally, we can explore how movies through the years in each genre have been rated:

- Do the the *quality* of movies of each genre (determined by user ratings) also display any trends throughout the years?
  e.g. Were there any stretches of years where comedies were more poorly-rated? Were Sci-Fi movies more highly acclaimed during any decade? What was happening in the world at that time that might have had an influence?

In [1]:

```
# Import necessary libraries

import pandas as pd
import numpy as np
import plotly
import plotly.graph_objs as go
from plotly.offline import iplot as plot, init_notebook_mode
init_notebook_mode(connected=True)
```

In [2]:

```
# Import CSV source data files

movies_file = r'C:\Users\Mark Pothier\Documents\Code\data_science_tutorials\edX - UC San Diego\Python for Data Science
movies = pd.read_csv(movies_file)
```

In [3]:

```
# Extract the year from the Title column, add as new column

movies_years = movies.copy(deep=True)
movies_years['Year'] = movies_years['title'].str.extract('.*\((.*)\).*')
movies_years = movies_years[movies_years['Year'].str.contains('^....$', na=False)]
movies_years['Year'] = movies_years['Year'].astype(np.int64)
movies_years.head()
```

Out[3]:

| | movieId | title | genres | Year |
|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1995 |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy | 1995 |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance | 1995 |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance | 1995 |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy | 1995 |

In [4]:

```
# Split apart the strings in genres, in order to extract a list of unique genres across the entire dataset

movie_genres = movies[['movieId','genres']].copy(deep=True)
movie_genres = movie_genres['genres'].str.split('|', expand=True)

unique_genres = list(pd.unique(movie_genres.values.ravel('K'))[:-2])
unique_genres
```

Out[4]:

```
['Adventure',
 'Comedy',
 'Action',
 'Drama',
 'Crime',
 'Children',
 'Mystery',
 'Documentary',
 'Animation',
 'Thriller',
 'Horror',
 'Fantasy',
 'Western',
 'Film-Noir',
 'Romance',
 'War',
 'Sci-Fi',
 'Musical',
 'IMAX']
```

In [5]:

```
# For each genre, create a new column in the DataFrame
# Each cell is evaluated as 'True' or 'False' depending the genre's inclusion in the original 'genres' string

movies_expanded = movies_years.copy(deep=True)

for value in unique_genres:
    movies_expanded[value] = movies_expanded['genres'].str.contains(value)
movies_expanded.head()
```

Out[5]:

| | movieId | title | genres | Year | Adventure | Comedy | Action | Drama | Crime | Children |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy | 1995 | True | True | False | False | False | True |
| **1** | 2 | Jumanji (1995) | Adventure\|Children\|Fantasy | 1995 | True | False | False | False | False | True |
| **2** | 3 | Grumpier Old Men (1995) | Comedy\|Romance | 1995 | False | True | False | False | False | False |
| **3** | 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance | 1995 | False | True | False | True | False | False |
| **4** | 5 | Father of the Bride Part II (1995) | Comedy | 1995 | False | True | False | False | False | False |

5 rows × 23 columns

In [6]:

```
# Group the DataFrame by 'Year', summing the True/False values of each genre as the rows are collapsed

movies_yearly_genre_counts = movies_expanded.groupby(['Year'])[unique_genres].sum().astype(np.int64)
movies_yearly_genre_counts['TOTAL'] = movies_yearly_genre_counts.sum(axis=1)
movies_yearly_genre_counts.tail()
```

Out[6]:

| Year | Adventure | Comedy | Action | Drama | Crime | Children | Mystery | Documentary | Animation | Thriller | Horror | Fantasy | Wes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2011** | 78 | 287 | 132 | 471 | 74 | 46 | 47 | 182 | 49 | 181 | 86 | 52 | |
| **2012** | 67 | 303 | 127 | 439 | 92 | 31 | 42 | 187 | 52 | 167 | 110 | 43 | |
| **2013** | 77 | 260 | 122 | 437 | 82 | 33 | 45 | 193 | 49 | 178 | 90 | 47 | |
| **2014** | 60 | 221 | 111 | 315 | 81 | 33 | 32 | 102 | 33 | 123 | 67 | 35 | |
| **2015** | 16 | 37 | 21 | 46 | 10 | 8 | 6 | 15 | 6 | 23 | 8 | 5 | |

## To reiterate an important point from the introduction:

Many movies were tagged with **multiple** genres**. The values being plotted correspond with the number of genre tags**, and *not* the number of movies. In other words, # genre tags > # movies**.**

*This may not an ideal way to process the data (it would be better to assign a single, primary genre to each movie; however, this information is not readily distinguishable), so we will have to accept that movies with greater complexity (i.e. more genre tags) are creating multiple genre hits.*

In [7]:

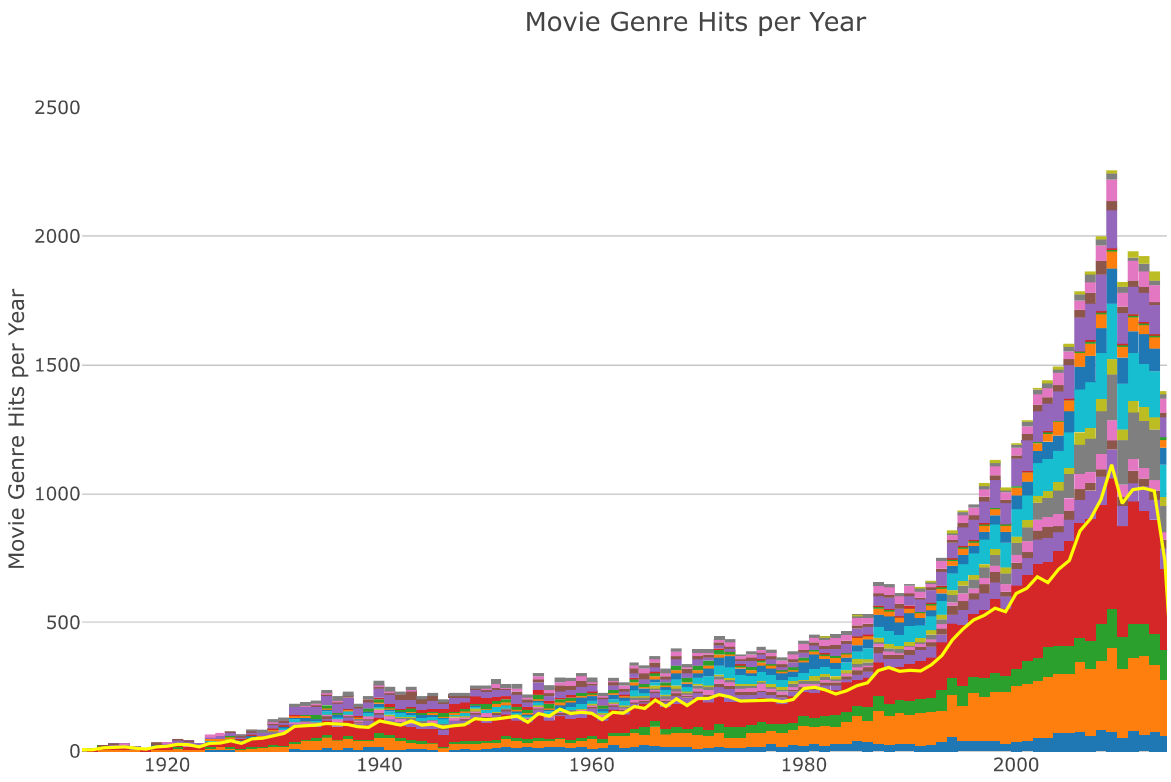
```
# To help clarify this last point, let's also get the data for number of movies per year,
# to be plotted on top of the stacked bars as a line graph

movies_per_year = movies_years.copy(deep=True)
movies_per_year = movies_per_year.groupby(['Year'], as_index=False)['movieId'].count()
movies_per_year.tail()
```

Out[7]:

| | Year | movieId |
|---|---|---|
| **113** | 2011 | 1016 |
| **114** | 2012 | 1022 |
| **115** | 2013 | 1011 |
| **116** | 2014 | 740 |
| **117** | 2015 | 120 |

In [8]:

```
# First, plot a stacked bar chart depicting the count of each genre, per year

data1 = [
    go.Bar(
        x=movies_yearly_genre_counts.index.values,
        y=movies_yearly_genre_counts[genre],
        name=genre
    ) for genre in unique_genres
]

data2 = [
    go.Scatter(
        x=movies_per_year['Year'],
        y=movies_per_year['movieId'],
        name='Movies per Year',
        mode='lines',
        line=dict(
            color = 'yellow',
            width = 2
        )
    )
]

data = data1 + data2

layout = go.Layout(
    barmode='stack',
    title='Movie Genre Hits per Year',
    yaxis={'title': 'Movie Genre Hits per Year', 'range': [0,2500]},
    xaxis={'title': 'Year', 'range': [1912,2015]},
    bargap=0.0,
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='genre-hits-per-year')
```



Year

This works, but the escalation of the number of movies made over time makes it difficult to see trends in yearly genre composition.

Let's normalize the plot to show all genres as a percentage out of 100 of each individual year:

In [9]:

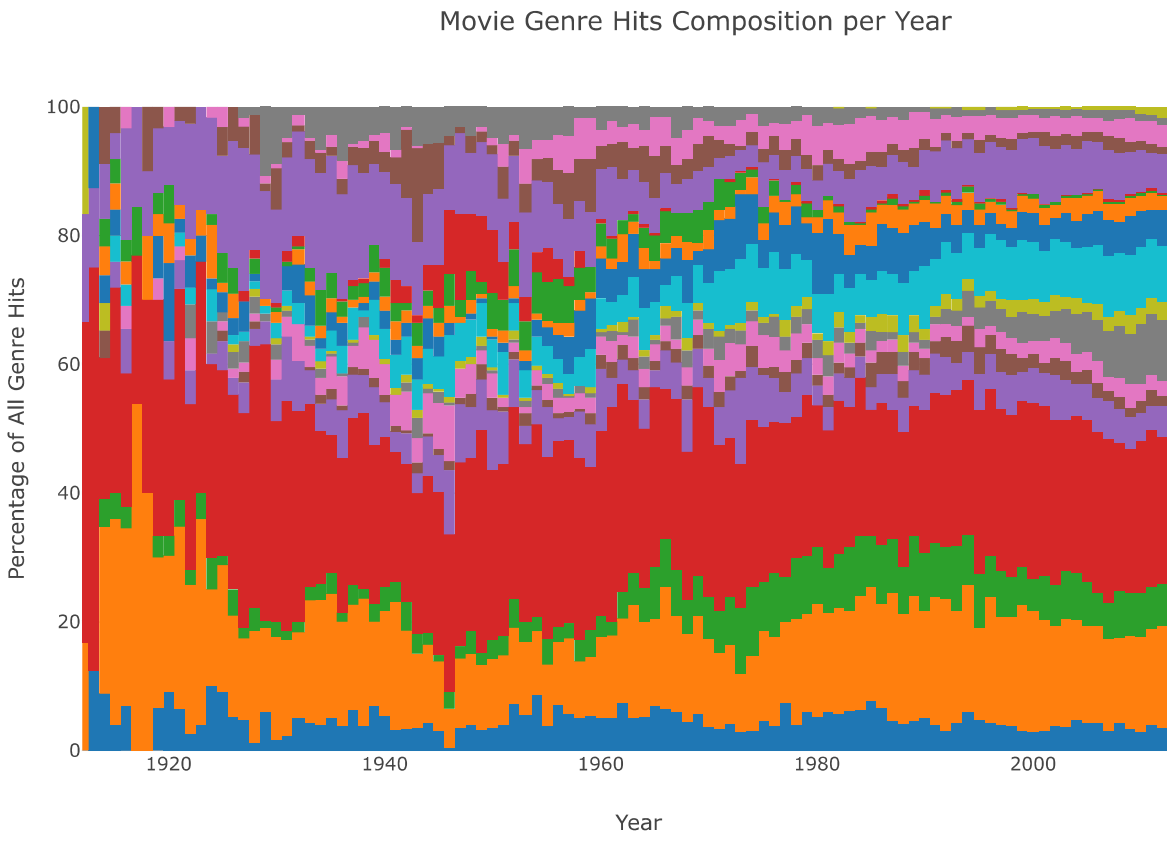
```
# Normalzie values as a percentage out of 100

movies_yearly_genre_counts_normalized = movies_yearly_genre_counts.div(movies_yearly_genre_counts['TOTAL'], axis=0)
movies_yearly_genre_counts_normalized = movies_yearly_genre_counts_normalized.fillna(0)
movies_yearly_genre_counts_normalized = movies_yearly_genre_counts_normalized.drop(['TOTAL'], axis=1)
movies_yearly_genre_counts_normalized = movies_yearly_genre_counts_normalized * 100
movies_yearly_genre_counts_normalized.tail()
```

Out[9]:

| Year | Adventure | Comedy | Action | Drama | Crime | Children | Mystery | Documentary | Animation | Thriller | Horror |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **2011** | 4.014411 | 14.770973 | 6.793618 | 24.240865 | 3.808543 | 2.367473 | 2.418940 | 9.366958 | 2.521873 | 9.315492 | 4.426145 |
| **2012** | 3.485952 | 15.764828 | 6.607700 | 22.840791 | 4.786681 | 1.612903 | 2.185224 | 9.729448 | 2.705515 | 8.688866 | 5.723205 |
| **2013** | 4.137560 | 13.970983 | 6.555615 | 23.481999 | 4.406233 | 1.773240 | 2.418055 | 10.370768 | 2.632993 | 9.564750 | 4.836110 |
| **2014** | 4.282655 | 15.774447 | 7.922912 | 22.483940 | 5.781585 | 2.355460 | 2.284083 | 7.280514 | 2.355460 | 8.779443 | 4.782298 |
| **2015** | 7.207207 | 16.666667 | 9.459459 | 20.720721 | 4.504505 | 3.603604 | 2.702703 | 6.756757 | 2.702703 | 10.360360 | 3.603604 |

In [73]:

```
# Now, plot a stacked bar chart depicting the PERCENTAGE of each genre of all counts per year

data = [
    go.Bar(
        x=movies_yearly_genre_counts_normalized.index.values,
        y=movies_yearly_genre_counts_normalized[genre],
        name=genre
    ) for genre in unique_genres
]

layout = go.Layout(
    barmode='stack',
    title='Movie Genre Hits Composition per Year',
    yaxis={'title': 'Percentage of All Genre Hits', 'range': [0,100]},
    xaxis={'title': 'Year', 'range': [1912,2015]},
    bargap=0.0,
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='genre-hits-composition-per-year')
```



In [11]:

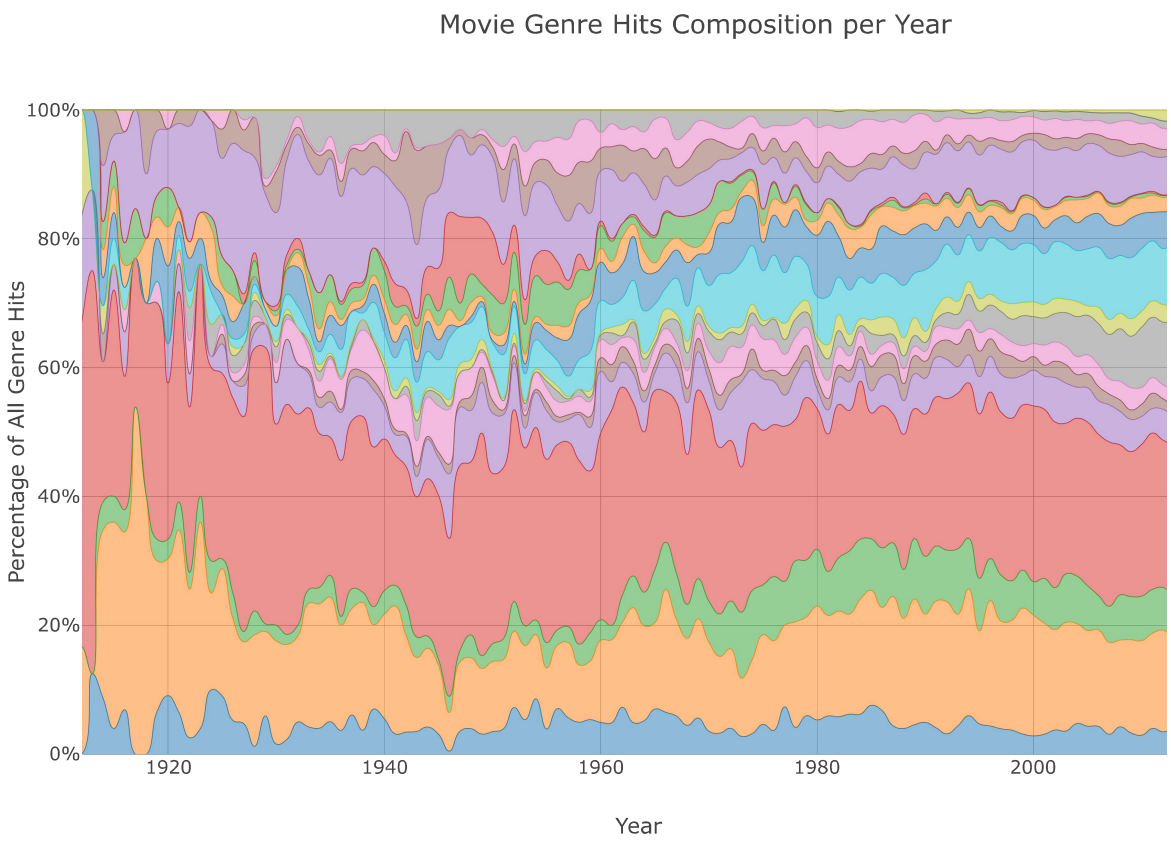
```
# Reformat values for creation of stacked area chart

movies_yearly_genre_counts_cumulative = movies_yearly_genre_counts_normalized.copy(deep=True)
movies_yearly_genre_counts_cumulative = movies_yearly_genre_counts_cumulative.cumsum(axis=1)
movies_yearly_genre_counts_cumulative.head()
```

Out[11]:

| Year | Adventure | Comedy | Action | Drama | Crime | Children | Mystery | Documentary | Animation | Thriller | Horror | Fantasy | Wes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1891** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **1893** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **1894** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | |
| **1895** | 0.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 1 |
| **1896** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | |

In [12]:

```
# Replot as a stacked area chart

data = [
    go.Scatter(
        x=movies_yearly_genre_counts_cumulative.index.values,
        y=movies_yearly_genre_counts_cumulative[genre],
        text=round(movies_yearly_genre_counts_normalized[genre],1),
        hoverinfo='x+text+name',
        name=genre,
        mode='lines',
        line=dict(
            width=0.5,
            shape='spline',
            smoothing=1.3
        ),
        fill='tonexty',
    ) for genre in unique_genres
]

layout = go.Layout(
    title='Movie Genre Hits Composition per Year',
    yaxis={'title': 'Percentage of All Genre Hits', 'range': [0,100], 'ticksuffix':'%', 'hoverformat': '%'},
    xaxis={'title': 'Year', 'range': [1912,2015]},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='genre-hits-composition-per-year-area-stack')
```



# How do ratings compare with these genre frequencies?

Let's bring in data from the ratings CSV file and try to determine how genres over the years are rated.

In [13]:

```
# Import CSV source data files

ratings_file = r'C:\Users\Mark Pothier\Documents\Code\data_science_tutorials\edX - UC San Diego\Python for Data Scienc
ratings = pd.read_csv(ratings_file)
```

In [14]:

```
ratings_simple = ratings[['movieId','rating']].copy(deep=True)
ratings_simple.head()
```

Out[14]:

| | movieId | rating |
|---|---|---|
| **0** | 2 | 3.5 |
| **1** | 29 | 3.5 |
| **2** | 32 | 3.5 |
| **3** | 47 | 3.5 |
| **4** | 50 | 3.5 |

In [15]:

```
ratings_with_genres = ratings_simple.merge(movies_expanded, on='movieId', how='inner')
ratings_with_genres.tail()
```

Out[15]:

| | movieId | rating | title | genres | Year | Adventure | Comedy | Action | Drama | Crime | ... | Thriller | Ho |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **19999850** | 121017 | 3.5 | The Gentleman from Epsom (1962) | Comedy\|Crime | 1962 | False | True | False | False | True | ... | False | F |
| **19999851** | 121019 | 4.5 | The Great Spy Chase (1964) | Action\|Comedy\|Thriller | 1964 | False | True | True | False | False | ... | True | F |
| **19999852** | 121021 | 4.5 | Taxi for Tobruk (1961) | Drama\|War | 1961 | False | False | False | True | False | ... | False | F |
| **19999853** | 110167 | 4.5 | Judge and the Assassin, The (Juge et l'assassi... | Crime\|Drama | 1976 | False | False | False | True | True | ... | False | F |
| **19999854** | 110510 | 4.5 | Série noire (1979) | Film-Noir | 1979 | False | False | False | False | False | ... | False | F |

5 rows × 24 columns

In [16]:

```
for column in ratings_with_genres:
    if column in unique_genres:
        ratings_with_genres[column] = ratings_with_genres[column] * ratings_with_genres['rating']

ratings_with_genres = ratings_with_genres.replace(0, np.NaN)
ratings_with_genres.tail()
```

Out[16]:

| | movieId | rating | title | genres | Year | Adventure | Comedy | Action | Drama | Crime | ... | Thriller | Hc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **19999850** | 121017 | 3.5 | The Gentleman from Epsom (1962) | Comedy\|Crime | 1962 | NaN | 3.5 | NaN | NaN | 3.5 | ... | NaN | |
| **19999851** | 121019 | 4.5 | The Great Spy Chase (1964) | Action\|Comedy\|Thriller | 1964 | NaN | 4.5 | 4.5 | NaN | NaN | ... | 4.5 | |
| **19999852** | 121021 | 4.5 | Taxi for Tobruk (1961) | Drama\|War | 1961 | NaN | NaN | NaN | 4.5 | NaN | ... | NaN | |
| **19999853** | 110167 | 4.5 | Judge and the Assassin, The (Juge et l'assassi... | Crime\|Drama | 1976 | NaN | NaN | NaN | 4.5 | 4.5 | ... | NaN | |
| **19999854** | 110510 | 4.5 | Série noire (1979) | Film-Noir | 1979 | NaN | NaN | NaN | NaN | NaN | ... | NaN | |

5 rows × 24 columns

In [17]:

```
total_yearly_genre_rating = ratings_with_genres.groupby(['Year'], as_index=True)[unique_genres].count()
total_yearly_genre_rating['TOTAL'] = total_yearly_genre_rating.sum(axis=1)
total_yearly_genre_rating.head()
```

Out[17]:

| | Adventure | Comedy | Action | Drama | Crime | Children | Mystery | Documentary | Animation | Thriller | Horror | Fantasy | Wes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Year** | | | | | | | | | | | | | |
| **1891** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **1893** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| **1894** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | |
| **1895** | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **1896** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | 0 | |

In [18]:

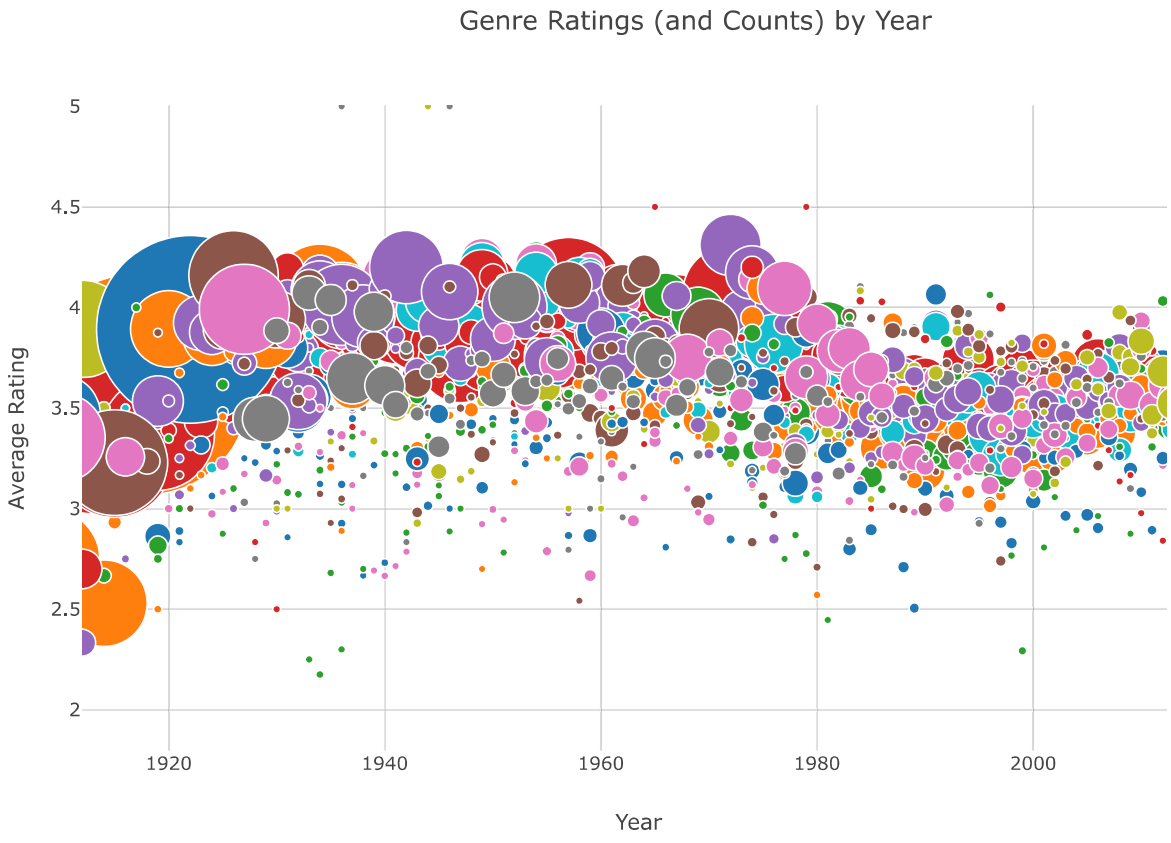
```
avg_yearly_genre_rating = ratings_with_genres.groupby(['Year'], as_index=True)[unique_genres].mean()
avg_yearly_genre_rating.head()
```

Out[18]:

| Year | Adventure | Comedy | Action | Drama | Crime | Children | Mystery | Documentary | Animation | Thriller | Horror | Fantasy | Wes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1891** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1893** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1894** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2.714286 | NaN | NaN | NaN | NaN | |
| **1895** | NaN | 2.25 | NaN | NaN | NaN | NaN | NaN | 4.000000 | NaN | NaN | NaN | NaN | |
| **1896** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 3.437500 | NaN | NaN | NaN | NaN | |

In [19]:

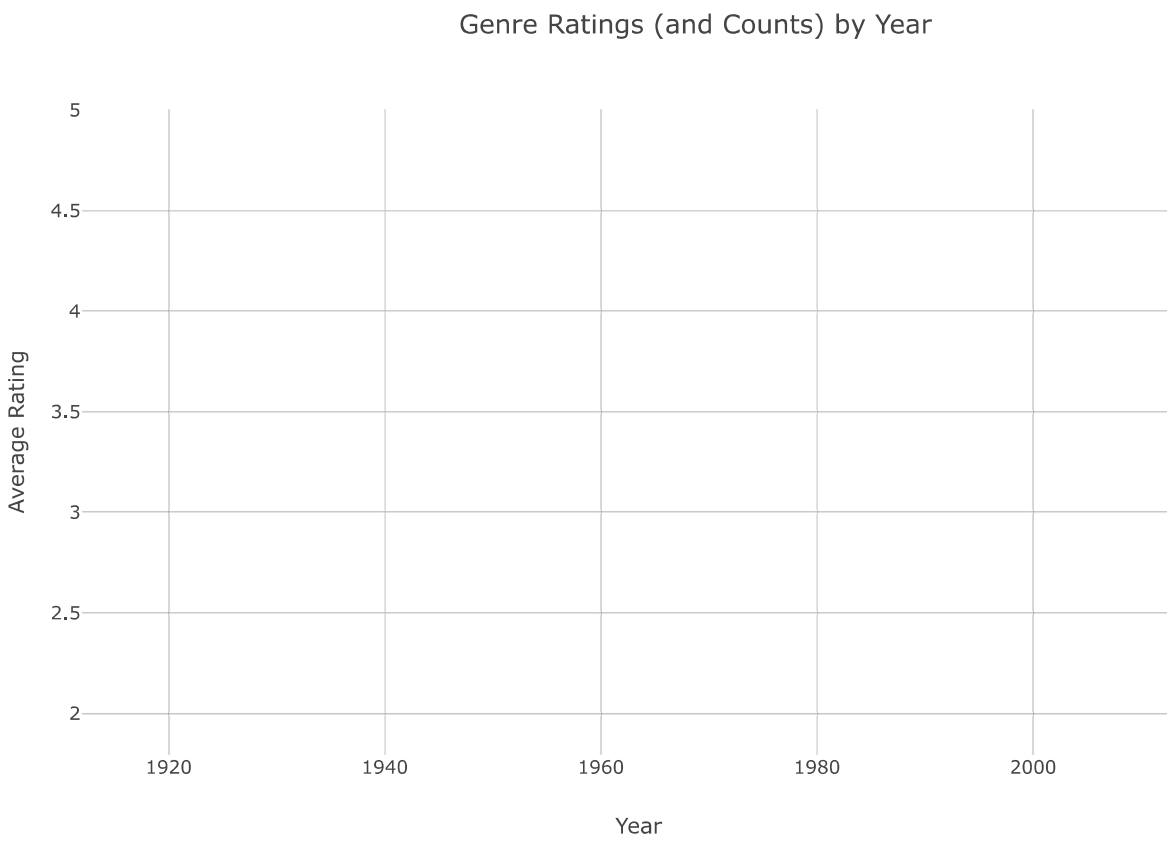
```
# Create a line plot, where:
# each line represents a genres,
# y-axis is the rating from 0-5,
# marker size reflects total number (count) of ratings

data = [
    go.Scatter(
        x=avg_yearly_genre_rating.index.values,
        y=avg_yearly_genre_rating[genre],
        name=genre,
        hoverinfo='x+y+name',
        mode='markers',
        line=dict(
            width = 1
        ),
        marker = dict(
            size = total_yearly_genre_rating[genre]/total_yearly_genre_rating['TOTAL']*150 + 5,
            opacity = 1
        ),
    ) for genre in unique_genres
]

layout = go.Layout(
    barmode='stack',
    title='Genre Ratings (and Counts) by Year',
    yaxis={'title': 'Average Rating', 'range': [1.8,5]},
    xaxis={'title': 'Year', 'range': [1912,2015]},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='genre-ratings-by-year')
```



That's too densely packed to be readily legible, so let's replot with the default trace visibility set to "legendsonly", then we can turn on individual or multiple traces as we please

In [20]:

```
data = [
    go.Scatter(
        x=avg_yearly_genre_rating.index.values,
        y=avg_yearly_genre_rating[genre],
        name=genre,
        hoverinfo='x+y+name',
        mode='markers',
        line=dict(
            width = 1
        ),
        marker = dict(
            size = total_yearly_genre_rating[genre]/total_yearly_genre_rating['TOTAL']*150 + 5,
            opacity = 1
        ),
        visible='legendonly'
    ) for genre in unique_genres
]

layout = go.Layout(
    barmode='stack',
    title='Genre Ratings (and Counts) by Year',
    yaxis={'title': 'Average Rating', 'range': [1.8,5]},
    xaxis={'title': 'Year', 'range': [1912,2015]},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='genre-ratings-by-year')
```



Finally, for the heck of it, let's look at the overall average rating of each genre:

In [31]:

```
avg_genre_rating = ratings_with_genres.copy(deep=True)
avg_genre_rating = avg_genre_rating[unique_genres]
avg_genre_rating = avg_genre_rating.mean(axis=0).sort_values(ascending=False)
avg_genre_rating
```

Out[31]:

```
Film-Noir      3.965381
War            3.809531
Documentary    3.739719
Crime          3.674528
Drama          3.674296
Mystery        3.663509
IMAX           3.655946
Animation      3.617494
Western        3.570498
Musical        3.558090
Romance        3.541803
Thriller       3.507112
Fantasy        3.505946
Adventure      3.501893
Action         3.443864
Sci-Fi         3.436765
Comedy         3.425990
Children       3.408114
Horror         3.277224
dtype: float64
```

In [70]:

```
data = [
    go.Bar(
        x=avg_genre_rating.index.values,
        y=avg_genre_rating
    )
]

layout = go.Layout(
    title='Average Rating by Genre',
    yaxis={'title': 'Average Rating', 'range': [3,4]},
    xaxis={'title': 'Genres', 'tickangle': -90},
    autosize=False,
    width=960,
    height=600,
    hovermode='closest'
)

fig = go.Figure(data=data, layout=layout)
plot(fig, filename='avg-genre-ratings')
```