Statistical Analysis - IMDB crawler

December 20, 2018

```
In [1]: import json
        from collections import Counter
```

```
In [2]: def read_data():
            data = "../files_for_tests/data.json"
            data_movies = []
            with open(data) as file:
                data_movies = json.load(file)
            return data_movies
```

```
In [3]: movies_data = read_data()
```

```
In [4]: movies_data[0]
```

```
Out[4]: {'_id': {'$oid': '5c19606ba21d900444de36b4'},
         'id': 1,
         'title': 'Aquaman',
         'runtime': 143,
         'summary': 'Arthur Curry learns that he is the heir to the underwater kingdom of Atlan
         'year': 2018,
         'rating': 7.9,
         'stars': ['Jason Momoa', 'Amber Heard', 'Willem Dafoe', 'Patrick Wilson'],
         'directors': ['James Wan'],
         'genre': ['Action', 'Adventure', 'Fantasy']}
```

0.0.1 Qual o tempo de duração médio dos filmes obtidos?

```
In [5]: def avg_runtime_movies(data):
            runtime_movies = 0
            runtime_movies_count = 0
            for movie_dict in data:
                for key,value in movie_dict.items():
                    if key == 'runtime':
                        runtime_movies += value
                        runtime_movies_count += 1

            return runtime_movies / runtime_movies_count
```

```
In [6]: avg_runtime_movies(movies_data)
```

```
Out[6]: 98.4
```

0.0.2 Quais são os diretores preferidos?

```
In [7]: def desired_directors(data):
            directors = []
            direcotrs_counter = []
            for movie_dict in data:
                for key,value in movie_dict.items():
                    if key == 'directors':
                        directors = directors + value

            direcotrs_counter = Counter(directors)
            direcotrs_counter = direcotrs_counter.most_common()

            return direcotrs_counter
```

```
In [8]: desired_directors(movies_data)[:10]
```

```
Out[8]: [('Jon Watts', 3),
         ('Bob Persichetti', 2),
         ('Peter Ramsey', 2),
         ('Rodney Rothman', 2),
         ('Anthony Russo', 2),
         ('Joe Russo', 2),
         ('David Leitch', 2),
         ('Peyton Reed', 2),
         ('Sam Levinson', 2),
         ('Brad Bird', 2)]
```

0.0.3 Qual a probabilidade de cada filme em seu gênero ter uma avaliação superior a 8?

```
In [48]: def probability_of_each_genre(data, threshold):

             #Getting genres
             genres = []
             for movie_dict in data:
                 for genre in movie_dict['genre']:
                     if genre not in genres:
                         genres.append(genre)


             #Making genres dict
             dict_genres_rating = {}
             for genre in genres:
                 dict_genres_rating[genre] = []

             #Getting the rating of each genre
             for movie in data:
                 for movie_genre in movie['genre']:
                     dict_genres_rating[movie_genre].append(movie['rating'])
```

```
            res = {}
            for gen in genres:
                total = len(dict_genres_rating[gen])
                total_of_best_rating = filter_by_rating(dict_genres_rating[gen], threshold)
                res[gen] = total_of_best_rating/total

            return res
```

```
In [49]: def filter_by_rating(ratings, threshold):
            ret = 0
            for v in ratings:
                if v > threshold: ret+=1
            return ret
```

```
In [50]: probability_of_each_genre(movies_data, 8)
```

```
Out[50]: {'Action': 0.1,
          'Adventure': 0.08163265306122448,
          'Fantasy': 0.1,
          'Animation': 0.23076923076923078,
          'Sci-Fi': 0.0,
          'Thriller': 0.14285714285714285,
          'Horror': 0.0,
          'Drama': 0.15789473684210525,
          'Comedy': 0.05555555555555555,
          'Crime': 0.0625,
          'Biography': 0.4,
          'Musical': 0.0,
          'Family': 0.0,
          'Romance': 0.0}
```

0.0.4 Qual a probabilidade de um filme ter avaliação superior a 8, considerando que ele não possui um diretor americano?

```
In [76]: def probability_movies_rating(data, threshold, year):
            dict_movies_limit_by_year = {}
            dict_movies_limit_by_year['before_'+str(year)] = []
            dict_movies_limit_by_year['after_'+str(year)] = []

            #povoar o dict
            for movie_dict in data:
                if movie_dict['year'] > year:
                    dict_movies_limit_by_year['before_'+str(year)].append(movie_dict['rating'
                else:
                    dict_movies_limit_by_year['after_'+str(year)].append(movie_dict['rating']

            res = {}
            res['before_'+str(year)] = filter_by_rating(dict_movies_limit_by_year['before_'+s
            res['after'+str(year)] = filter_by_rating(dict_movies_limit_by_year['after_'+str(

            return res
```

```
In [77]: probability_movies_rating(movies_data, 8, 2001)
```

```
Out[77]: {'before_2001': 0.07865168539325842, 'after2001': 0.18181818181818182}
```