Data-Analysis1

October 7, 2024

1 Data Analysis Example

2 Superhero Movies

- Explore the data with `info()`, `describe()`, `head()`
- How many DC? Marvel? `value_counts()`
- Highest Rated imdb movie? Lowest?
- dropping NaN values

The info() method shows information about the DataFrame. Specifically the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

[2]:
```python
import pandas as pd

sh = pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/master/
↪superhero/superhero-movie-dataset-1978-2012-header.csv")
sh.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 10 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Year                        49 non-null     int64
 1   Title                       49 non-null     object
 2   Comic                       49 non-null     object
 3   IMDB Score                  49 non-null     float64
 4   RT Score                    49 non-null     int64
 5   Composite Score             49 non-null     float64
 6   Opening Weekend  Box Office 46 non-null     float64
 7   Avg Ticket Price            49 non-null     float64
 8   Opening Weekend Attendance  46 non-null     float64
 9   US Population That Year     49 non-null     int64
dtypes: float64(5), int64(3), object(2)
memory usage: 4.0+ KB
```

The .describe() provides summary statistics for numerical columns in our DataFrame.

```
[3]: sh.describe()
```

```
[3]:               Year  IMDB Score   RT Score  Composite Score  \
     count    49.000000   49.000000  49.000000        49.000000
     mean   2001.326531    6.212245  53.204082        57.663265
     std       9.764706    1.530201  29.643001        21.815368
     min    1978.000000    2.700000   8.000000        19.500000
     25%    1997.000000    5.300000  26.000000        39.500000
     50%    2004.000000    6.400000  59.000000        62.500000
     75%    2008.000000    7.400000  79.000000        75.000000
     max    2012.000000    9.100000  95.000000        91.500000

            Opening Weekend  Box Office  Avg Ticket Price  \
     count                 4.600000e+01         49.000000
     mean                  5.620126e+07          5.963061
     std                   4.760047e+07          1.667506
     min                   8.700680e+05          2.340000
     25%                   1.622806e+07          4.590000
     50%                   5.265976e+07          6.210000
     75%                   6.555713e+07          7.180000
     max                   2.074387e+08          7.930000

            Opening Weekend Attendance  US Population That Year
     count                4.600000e+01             4.900000e+01
     mean                 8.654924e+06             2.844763e+08
     std                  6.345187e+06             2.784988e+07
     min                  1.895573e+05             2.225845e+08
     25%                  3.302858e+06             2.677836e+08
     50%                  7.773355e+06             2.930457e+08
     75%                  1.125807e+07             3.043748e+08
     max                  2.619176e+07             3.140560e+08
```

Let us look at some of the data. You can use .head() to get the first 5 rows or .tail() to get the last 5 rows. To obtain random rows use .sample() method

```
[4]: #look at some of the data
     sh.head()
```

```
[4]:    Year         Title Comic  IMDB Score  RT Score  Composite Score  \
     0  1978      Superman    DC         7.3        95             84.0
     1  1980   Superman II    DC         6.7        88             77.5
     2  1982   Swamp Thing    DC         5.3        60             56.5
     3  1983  Superman III    DC         4.9        24             36.5
     4  1984     Supergirl    DC         4.2         8             25.0

        Opening Weekend  Box Office  Avg Ticket Price  Opening Weekend Attendance  \
     0                    7465343.0              2.34                 3190317.521
```

```
1                14100523.0              2.69                 5241830.112
2                       NaN              2.94                         NaN
3                13352357.0              3.15                 4238843.492
4                 5738249.0              3.36                 1707812.202

   US Population That Year
0                222584545
1                227224681
2                231664458
3                233791994
4                235824902
```

[6]:
```
# are they all DC comics? Try a random same of 10
sh.sample(n=10)
```

[6]:
```
    Year                 Title   Comic  IMDB Score  RT Score  \
43  2011                  Thor  Marvel         7.0        77
25  2005         Batman Begins      DC         8.3        85
11  1995        Batman Forever      DC         5.4        42
39  2010            Iron Man 2  Marvel         7.1        74
45  2012  Marvel's The Avengers  Marvel         8.7        92
35  2008              Iron Man  Marvel         7.9        94
29  2006  X-Men: The Last Stand  Marvel         6.8        57
4   1984             Supergirl      DC         4.2         8
27  2005        Fantastic Four  Marvel         5.7        27
37  2009              Watchmen      DC         7.7        64

    Composite Score  Opening Weekend Box Office  Avg Ticket Price  \
43             73.5                  65723338.0              7.93
25             84.0                  48745440.0              6.41
11             48.0                  52784433.0              4.35
39             72.5                 128122480.0              7.89
45             89.5                 207438708.0              7.92
35             86.5                  98618668.0              7.18
29             62.5                 102750665.0              6.55
4              25.0                   5738249.0              3.36
27             42.0                  56061504.0              6.41
37             70.5                  55214334.0              7.50

    Opening Weekend Attendance  US Population That Year
43                8.287937e+06                311591917
25                7.604593e+06                295753151
11                1.213435e+07                262803276
39                1.623859e+07                308745538
45                2.619176e+07                314055984
35                1.373519e+07                304374846
29                1.568712e+07                298593212
```

```
4                 1.707812e+06              235824902
27                8.745944e+06              295753151
37                7.361911e+06              307006550
```

The column "comic" contains nominal or categorical data (e.g., names of comics), the .value_counts() method will return the count of unique values in that column. This output number of occurrences of each comic in descending order.

[10]:
```
## Who has more movies in the dataset? DC or Marvel?
sh['comic'].value_counts()
```

[10]:
```
Marvel    29
DC        19
Name: comic, dtype: int64
```

If set normalize to True we return relative frequency (proportion) of each unique value in the 'comic' column of the sh DataFrame. Instead of just showing the counts, it will show the proportion of total entries that each unique value represents. For example, if a comic appears 3 times in a column with a total of 10 entries, the result for that comic would be 0.3 (i.e., 30%).

[11]:
```
## let's see that as a percentage of the total
sh['comic'].value_counts(normalize=True)
```

[11]:
```
Marvel    0.604167
DC        0.395833
Name: comic, dtype: float64
```

[12]:
```
## what are the ratios in the last 10 years of data ?
sh[ sh['year'] >2002]['comic'].value_counts(normalize=True)
```

[12]:
```
Marvel    0.741935
DC        0.258065
Name: comic, dtype: float64
```

[13]:
```
# what about the first 10 years of data? 1978 - 1987?
sh[ sh['year'] < 1988]['comic'].value_counts(normalize=True)
```

[13]:
```
DC        0.833333
Marvel    0.166667
Name: comic, dtype: float64
```

[14]:
```
sh.head()
```

[14]:
```
   year          title  comic  imdb  rt  composite  opening_weeked_bo  \
0  1980    Superman II     DC   6.7  88       77.5         14100523.0
1  1982    Swamp Thing     DC   5.3  60       56.5                NaN
2  1983   Superman III     DC   4.9  24       36.5         13352357.0
```

```
3  1984         Supergirl      DC    4.2   8        25.0           5738249.0
4  1986  Howard the Duck  Marvel    4.3  16        29.5           5070136.0

   avg_ticket_price  opening_weekend_attend  us_pop_that_year
0              2.69             5241830.112         227224681
1              2.94                     NaN         231664458
2              3.15             4238843.492         233791994
3              3.36             1707812.202         235824902
4              3.71             1366613.477         240132887
```

Let us create a new DataFrame sh2 that is a copy of sh, but with all rows containing any NaN (missing) values removed. In other words, it filters out all incomplete rows from the DataFrame.

[8]:
```
## skip nulls in analysis
sh2 = sh.dropna()
sh2.head()
```

[8]:
```
   Year            Title   Comic  IMDB Score  RT Score  Composite Score  \
0  1978         Superman      DC         7.3        95             84.0
1  1980      Superman II      DC         6.7        88             77.5
3  1983     Superman III      DC         4.9        24             36.5
4  1984        Supergirl      DC         4.2         8             25.0
5  1986  Howard the Duck  Marvel         4.3        16             29.5

   Opening Weekend Box Office  Avg Ticket Price  Opening Weekend Attendance  \
0                   7465343.0              2.34                 3190317.521
1                  14100523.0              2.69                 5241830.112
3                  13352357.0              3.15                 4238843.492
4                   5738249.0              3.36                 1707812.202
5                   5070136.0              3.71                 1366613.477

   US Population That Year
0                222584545
1                227224681
3                233791994
4                235824902
5                240132887
```

Let us finds the highest value in the 'IMDB Score' column of the sh2 DataFrame and stores it in the variable.

[13]:
```
# Movie with the best IMDB score?
```

[12]:
```
best_imdb = sh2['IMDB Score'].max()
best_imdb
```

[12]:
```
9.1
```

Let us filter the sh2 DataFrame to return all rows where the 'IMDB Score' is equal to the maximum score stored in the variable best_imdb. This will give you row(s) corresponding to the movie(s) with the highest IMDB score in the DataFrame. It's a way of identifying the specific entries with the maximum score. We see only one movie has the score of 9.1.

```
[14]: sh2[ sh2['IMDB Score'] == best_imdb ]
```

```
[14]:     Year                  Title Comic  IMDB Score  RT Score  Composite Score  \
      46  2012  The Dark Knight Rises    DC         9.1        86             88.5

          Opening Weekend  Box Office  Avg Ticket Price  Opening Weekend Attendance  \
      46               160887295.0               7.92                   20314052.4

          US Population That Year
      46                314055984
```

```
[ ]:
```