Exploratory Data Analysis (EDA) on Anime Data from MyAnimeList – Project Report

TO: The Bridge Bootcamp faculty team
DATE: December 11, 2022
WRITTEN BY: Christian Díaz
CONTACT PERSON: Christian Díaz (christin.d.d.a@gmail.com)

Historical and version control

Version	Date	Authors	Comments
0.1	December 7, 2022	Christian Díaz	Creation, Introduction
0.2	December 8, 2022	Christian Díaz	API and Cleaning part
0.3	December 9, 2022	Christian Díaz	Conclusions and references
0.4	December 10, 2022	Christian Díaz	Grammar checking
1.0	December 11, 2022	Christian Díaz	Grammar checking and standardization document

The following personnel are required to approve this document The approval of the document implies that it meets the required quality, and is complete	
Project:	Exploratory Data Analysis (EDA) on Anime Data from MyAnimeList – Project Report
Delivered by:	Christian Díaz
Approved by:	The Bridge Bootcamp faculty team
Date:	December 11, 2022

Purpose and guidelines of this Document
Justification of comprehension of project scope

Índice

1 Topic Introduction

"Anime" is the term used by Western audiences to describe Japanese animated films and television programs (although it is used to describe any animation in Japan).

At first, animated pieces were known as senga eiga (線画映画, line-drawing film) or senga kigeki (線画喜劇, cartoon comedy film), which were often specified in katakana (cartoon comedy, カートン・コメディ, cartoon comedy).

Between 1907 and 1912 the first animated shorts were made, usually associated with political cartoonists, and in 1933 came the first piece of Japanese animation or spoken Japanese anime, Chikara to Onna Yo no Naka.

It was probably not until 1962 that the word anime began to be used, in a standardized way, in Japan to refer to animated productions. It seems that the film magazine Eiga Hyoron was the first.

During the 1970s some of the most popular anime in the country were born, such as Ashita no Joe (1970, Fuji TV), Arupusu no Shoujo Haiji (Heidi, 1974, Fuji TV), Lupin III (1971, YTV), Gatchaman (1972, Fuji TV), Mazinger Z (1972, Fuji TV), Uchuu Senkan Yamato (1974, YTV), Candy Candy (1976, TV Asahi) or Kidou Senshi Gundam (1979, NBN), but it was not until the 1980s when a real revolution would take place.

The first half of the 1990s saw the emergence of a new type of animation that was ready to blow the brains of viewers and burst the expectations of everything seen until then in cartoons. It was Japanese anime and there is no doubt that the two culprits of this irruption were Dragon Ball and Akira, two authentic bombshells that made us become aware of this new style of animation. These phenomena were accompanied by the arrival of private television stations, with many hours of programming to fill, which were busy with lots of Japanese anime series that were cheap to license. New publishers emerged, acquiring the rights to anime films destined to fill the shelves of video stores in response to this new demand.

Today's teenagers are getting hooked on Japanese anime again. It's happening to them as it happened in the 90s, for example, with Dragon Ball, The Knights of the Zodiac (Saint Seiya) or Chicho Earthquake (Chicho Terremoto - Dash Kappei). Not to mention the Dragon Ball phenomenon. It is true that now there are also many adults who maintain their taste for the series and movies created by the famous Japanese anime studios, and that has an influence.

Anime may be going through one of its best periods in history. The genre has audiences almost everywhere in the world. The stories are reaching diverse audiences and, technology permitting, a series of tools are available to improve manga adaptations or new proposals. A boom.

2 Topic Explanation

Usually when we think of Data Science, Machine Learning or Artificial Intelligence, we think of the models and the wonderful applications, but the first step is usually to explore and clean the data.

The purpose of this EDA is getting insights out of data while exploring it (after doing some Data cleaning/preparation/transformation) in order to answer some previos questions and prove some hypothesis.

The parts in the project are as follows:

- Introduction
- Getting the information
- Data Preparation, Cleaning and Descriptive Analysis
- Data and Visual Analysis
- Conclusions
- Reference

2.1 Structure of the Jupyter File

- Introduction
- Getting the information
- Data Preparation, Cleaning and Descriptive Analysis
    - Importing libraries and checking the raw information
    - Next esteps explanation
    - Cleaning the columns with empty lists
    - About released, Scored_by, Score and Rank
    - Dealing with Audience and Rating columns
    - Removing unncesary columns
    - Change null values to Unknown
    - Dealing with Zero values in N_Episodes column
    - Standardizing duration column
    - Checking results after the cleaning process

    - Saving cleaned df using Pickle
    - Opening cleaned df using Pickle
- Data and visual Analysis
    - One Dimensional-Analysis
        - Numerical Columns
            - N_Episodes
            - Duration
            - Score
            - Rank
            - Released
        - Categorical Columns
            - Type
            - Source
            - Rating
            - Genre
            - Theme
            - Producers
            - Studios
    - Bi-Dimensional-Analysis
        - Score and Type
        - Score and Source
        - Score and Number of Episodes
        - Score and Duration of the episodes
        - Score and Rating
        - Score and Genre
        - Score and Theme
        - Score and Producers
        - Score and Studios
- Conclusions

3 Data Collection

The data was collected from the website called MyAnimeList using a unofficial API called Jikan API.

MyAnimeList Website → https://myanimelist.net/

Jikan API → https://docs.api.jikan.moe/

In order to collect it, a script was created (Link).

Imported libraries used in the script:

- import requests, json, os, sys, time
- import pandas as pd
- from datetime import datetime

This script can be used from the jupyter file in case we want to collect data again.

Usually when we make a call to the REST API, we expect to get the entire requested data set. At first the data was collected going anime by anime using ID's. However, this process was taking way too much time. So, some investigation was done and the pagination process was found.

For pagination, if the result includes hundreds or thousands of records, we will likely retrieve the first batch with a limited number of records. This REST API uses the pagination method of splitting data sets into discrete pages – a set of paginated endpoints.

For this reason, in this script there are two loops.

One to go, page by page. (**Annex 1.1**)

The second for loop is to go anime by anime in the page, using a try/except in case there is an error. (**Annex 1.2**)

The main body to collect tha data inside the "Try" generates a dictionary and and append the information to a list. (**Annex 1.3**)

Some of the Keys in the dictionary uses a def due to some of the anime dows not have those fields. (**Annex 1.4**)

After all the information is calledted, we save it in a CSV file. (**Annex 1.5**)

In order to save the file in "src/data" an add the date and time in the name of the csv file, we create the next variables. (**Annex 1.6**)

4 Data Cleaning

The data cleaning process, also known as data scrubbing or data cleansing, can have a huge impact on the reliability and validity of the final data, as it ensures that we are only using the highest-quality data to perform our analysis. By rushing or eliminating the data cleaning step, we run the risk of including false, misleading, or duplicated records in our final dataset. Following a thorough data cleaning process will minimize errors made due to data that is formatted incorrectly.

This of data cleaning, is of utmost importance even though it is time consuming and the least enjoyable task of the data science process.

The process was done as follows:

4.1 Importing libraries and checking the raw information

First we import the necessary libraries and the path variables to folders (**Annex 2.1.1**).

Then we import the dataframe directly from the raw data (**Annex 2.1.2**).

Checking dtype of the columns, the number of null values and percenteage of null values. (**Annex 2.1.3**).

So far now we can see that we can't be completely confident in these results as we can see in the dataframe that there are many columns with empty lists. First, we will have to clean the data, compare it with the previous results to see the change and then we can proceed to analyze.

There are some columns that the information is storage inside a list and we would like to display them by removing brackets and commas from the list. in this way we can later analyze it if needed.

4.2 Next steps explanation

Now we can fully see the work we need to do.

We will remove the column Season due to the high number of missing values.

- In Audience, rating ,Genre, Theme, Studios and Producers we will proceed to change null values to Unknown, due to we do not have that information.
- About released, Scored_by, Score and Rank:
    - Released: We will find the missing values with interpolation
    - Scored_by: We will find the missing values with interpolation
    - Score: We will find the missing values with interpolation
    - Rank: once we have all the values in Scored_by and Score we will use linear regresion to predict the missing values.

- Cover, English_Title and Japanses_Title are not needed, we can also remove these columns.

- Audience column: In the case of Audience, it is basically the Japanese way to classify the people that would watch the anime. From My Anime List a different way of classification (more international style) was created to be more understandable for non japanese. For this reason, we are not going to use Audience and we are going to drop it.

Duration column. We are going to standardize is converting the information into a numeric value representing minutes.

4.3 Cleaning the columns with empty lists

Removing brackets and commas from the columns with lists. (**Annex 2.3.1**)

After doing that, we can see that in the columns "Audience","Genre","Theme","Studios" and "Producers" there are many empty spaces. For the purpose of checking missing data, we will replace them with NaN. However, later we will replace them with "Unknown". (**Annex 2.3.2**)

Let's check again the percenteage of null values. (**Annex 2.3.3**)

4.4 About released, Scored_by, Score and Rank

It is time to deal with the cleaning of Score, Scored_by and Rank values,. We check the number of null values in each of those columns. (**Annex 2.4.1**)

After that we replace teh NaN values to zeros and count how many of them we have. We have 2590 rows with zero values. (**Annex 2.4.2**)

Next we proceed with the interpolation to replace the missing null values in Released, Score and Scored_by. (**Annex 2.4.3**)

It is possible that during the interpolation in scored_by, we suddenly get negative number, so we are going to check the min of Scored_by and in case we have we will convert the Scored_by column to absolute value using abs from numpy. (**Annex 2.4.4**)

Now let's launch the prediction function to replace the zero values in the rank column. We also check how many zero and null values we have now, we can see that we do not have now. (**Annex 2.4.5**)

To finalize with Scored_by, Rank and Released columns, we will chenge the column type to integer. (**Annex 2.4.6**)

4.5 Dealing with Audience and Rating columns

As mentioned before, Audience column is basically the japanese way to classify the people that would watch the anime. From My Anime List a different way of classification (more international style) was created to be more understandable for non japanese.

We can print both columns to see how they are classified, however, the columna audience Will be remove. (**Annex 2.5.1**)

4.6 Removing unnecessary columns

Let's proceed dropping the unnecesary columns for this analysis. (**Annex 2.6.1**)

4.7 Change null values to Unknown

We should not forget to assign Unkowns to all the Nan in the CATEGORICAL VARIABLES columns. (**Annex 2.7.1**)

4.8 Dealing with Zero values in N_Episodes column

Sometimes we do not know the number of episodes or there will be series to be released that they still don't have the decided number of episodes to have.

So, first we check the number of animes with N_Episodes equal to zero

Since the number of animes with zero episodes is low, we can input them without having an impact in the data. (**Annex 2.8.1**)

We are going to Change zero values to the previous column value. (**Annex 2.8.2**)

We check again the number of animes with N_Episodes equal to zero. (**Annex 2.8.3**)

4.9 Standardizing duration column

To finalize, let's change the format in the duration column. Let's leave the columns just with the duration in minutes. (**Annex 2.9.1**)

Now are going to convert the Unknown values to nan in order to be able to interpolate the missing information. (**Annex 2.9.2**)

Replacing zero values to the previous column value. (**Annex 2.9.3**)

4.10 Checking results after the cleaning process

Finally, let's check the result of this cleaning process. As result, we see no null values in the columns and everything looks clean. (**Annex 2.10.1**)

We check the df after being cleaned. (**Annex 2.10.2**)

Let's have a look at the descriptive stats of the records. (**Annex 2.10.3**)

We find out that the max value in Released is 2024, so, let's get the count of values greater than 2022 in the column 'Released to see if this could affect to our analysis. (**Annex 2.10.4**)

4.11 Saving cleaned df using Pickle

We save the file in pickle style for a late review. Why saving it to pickle? Pickle is a serialized way of storing a Pandas dataframe. Basically, you are writing down the exact representation of the dataframe to disk. This means the types of the columns are and the indices are the same. If you simply save a file as csv, you are just storing it as a comma separated list. Depending on your data set, some information will be lost when you load it back up. (**Annex 2.11.1**)

4.12 Opening cleaned df using Pickle

Opening the information from pickle. (**Annex 2.12.1**)

Here is a review of the information got it so far:

Type: There are 7 different categories and the top on is TV.

Source: There are 17 different categories and the top one is Original.

N_Episodes: The average number of episodes is 15 episodes.

Duration: The usual duration of an episode is 24 minutes.

Rating: There are 7 different ratings, and the most common es PG-13 - Teens 13 or older.

Released: There are 206 animes after 2023 for future releases. Because this number is so low we will not allocate them to a different year.

Score: The mean rating of all the anime is around 6.5

For categorical variables, measures like mean, std, quartiles do not make sense, hence those positions are filled using NaN above and for continuous variables, measures like unique, top and frequency don’t make much sense hence those positions are filled using NaN values as well for those respective places.

5 Data and Visual Analysis

For the dataframe, a correlation matrix is made because it has many numerical variables that can be related to each other.
A couple of correlations are detected between score, N_Epsidoes and Duration. They will be analyzed later. (**Annex 3**)

5.1 One Dimensional-Analysis

5.1.1 Numerical Columns

5.1.1.1 N_Episodes

We do a describe and a boxplot to find information about it.
We find that the 75 % of the animes has 13 episodes. This is actually the standard number of episodes for an anime. (**Annex 3.1.1.1**)

5.1.1.2 Duration

First, let's try doing a histplot. (**Annex 3.1.1.2.1**)
At first glance is pretty hard to see using a regular histplot, for this reason we are going to use a logarithmic scale. (**Annex 3.1.1.2.2**)
However, using the logarithmic scale is also pretty hard to ger meaningful information. Let's try a different approach.
We do a .describe() in Duration to check the number of samples, the mean value, the standard deviation, the minimum, maximum, median and the values corresponding to the 25% and 75% percentiles. (**Annex 3.1.1.2.3**)
We have 24105 anime records, 5501 animes with more than 25 minutes and 18604 animes with more than 25 minutes.
For this reason, let's narrow down the graph with anime lower than 26 minutes. (**Annex 3.1.1.2.4**)
Now we can see that the distribution tells us that most of the animes has a duration between 5 and 25 minutes.
However, many of those will belong to OVAS, Specials or Movies, so let's check those to belong to Tv Series (**Annex 3.1.1.2.5**)
Basically, most of the Tv Series animes, 7345 out of 7509, has 25 or less episodes and just 164 with more than 25 minutes. And from those, 6060 have between 10 to 25 minutes.

5.1.1.3 Score

After doing a distplot and a boxplot we can see that Most of the animes are scored between 5 to 8. (**Annex 3.1.1.3.1**) and (**Annex 3.1.1.3.2**)
There are 3938 animes between 5 and 6 points.
There are 15301 animes between 6 and 7 points.
There are 3602 animes between 7 and 8 points.
There are 579 animes between 8 and 9 points. (**Annex 3.1.1.3.3**)

5.1.1.4 Rank

Here we can check the top 20 at the moment. We can see that Bleach: Sennen Kessen-hen is number 1 right now in Total Score and Kaguya-sama wa Kokurasetai is number 3. Both recently released. With time and more votes,

they might keep their position or change it, we wouldn't know for a while. (**Annex 3.1.1.4**)

5.1.1.5 Released

We can see a serious increase in the 80's. At that time, anime became mainstream in Japan, experiencing a boom in production with the rise in popularity of anime like Gundam, Macross, Dragon Ball, and genres such as real robot, space opera and cyberpunk. (**Annex 3.1.1.5.1**)
We can appreciate a decline in releases during 2019 and 2020, this could be duo to the COVID-19 period.
Another possible factor is that during the last years in the anime industry, there are so many copy-and-paste anime. The only difference may be the relationship between characters, slight changes in the plot, and different characters and/or setting. The anime industry has definitely declined in quality and uniqueness throughout the years. However, there are still releasing some great gems that keep the industry still strong, lately examples: My Hero Academia, Attack On Titan, Tokyo Revengers, Jujutsu Kaisen and Kimetsu no Yaiba.
Over here we can see the evolution of type of animes per year. So far TV series were the favorite type from the late 20's, however, since 2020 we can see that ONA is getting more popular. We will need to see what happen in the coming years. (**Annex 3.1.1.5.2**)

5.1.2 Categorical Columns

5.1.2.1 Type

First let's define some of this names:

- Original Net Animation (ONA is an anime that is directly released onto the Internet.
- Original Video Animation (OVA is an animated film or series made specially for release in home-video formats. OVA is created for selling (by Video or DVD). It's intended to the small number of viewer without advertisement. It means more otaku friendly theme.
- Movie are just regular movies of the anime, it could be part of the story or ir could be not.
- TV regular anime series broadcasted on TV.
- Special (aka TV Special) is not weekly. Usually yearly or one shot. It's have only one episode but it had longer length (ex 2 hours). It's still intended for broadcast.
- Music are Anime Music Video (anime music video, better known by the acronym AMV), is a music video generally made with an Anime theme. The main characteristic of AMVs is that they are composed of scenes from one or more anime series or movies accompanied by a song that seeks to synchronize with the rhythm of the latter.

We do a pie plot to see the percentage of each type. (**Annex 3.1.2.1**)
Then we do a value_counts() to see the number for each type.

5.1.2.2 Source

So, what is source material? It's the material that's the source of the story or content used in an anime. Sometimes anime has original stories, which I'll get to in a bit, but often they're based off pre-existing works such as manga, light novels, visual novels, etc.... (**Annex 3.1.2.2.1**)
We can see that most of the anime comes from an original idea (where anime companies write up the plot and design the characters themselves). Manga and game are also a popular source for animes. Of all the anime we have in the list, 24105, there are a total of 14681 that come from an original source, manga or game. This means that 60.90437666685334995 % of the anime belong to this group. (**Annex 3.1.2.2.2**)

5.1.2.3 Rating

These only represent target demographics and don't describe what content is in the anime. (**Annex 3.1.2.3.1**)

- Rated G: General audiences – All ages admitted.
- Rated PG: Parental guidance suggested – Some material may not be suitable for children.
- Rated PG-13: Parents strongly cautioned – Some material may be inappropriate for children under 13.
- Rated R: Restricted – Under 17 requires accompanying parent or adult guardian.
- Rated Rx: Hentai. No one under 18 admitted.
- Rated R+: Means that there is nudity. Restricted – Under 17 requires accompanying parent or adult guardian

PG-13 - Teens 13 or older is the most popular. (**Annex 3.1.2.3.2**)

5.1.2.4 Genre

The different types of anime are in the dozens. If you're an avid watcher of the Big 3 Anime, then you may have come to learn that every show is based on a specific anime genre. (**Annex 3.1.2.4.1**)
Today, anime is available in a wide range of genres such as drama, action, supernatural, and horror, to mention a few. They are made for a young girl or young boy as well as adults. They feature tough female characters and handsome male characters.
We can appreciate that Comedy Genre is the most typical one. (**Annex 3.1.2.4.2**)

5.1.2.5 Theme

Music, School and history theme are the ones with more animes. We remove the unknowns from the countplot to see it better. (**Annex 3.1.2.5**)

5.1.2.6 Producers

Top 10 producers with unknowns and Top 10 producers without unknowns.
NHK, Nippon Hoso Kyokai (Japan Broadcasting Corporation), is the producer with the most anime in the world. However, it is a public entity. So Aniplex would be the first private company that most anime produces.

Really easy to understand why Aniplex is leading the chart. It has tittles like Fullmetal Alchemist: Brotherhood,Sword Art Online, Naruto, Naruto: Shippuuden, Kimetsu no Yaiba, Ao no Exorcist, Nanatsu no Taizai, Bleach, Soul Eater, etc With tittles like this under them, pretty normal that they lead. (**Annex 3.1.2.6**)

5.1.2.7 Studios

Top 10 Studios without unknowns.
Toei Animation is the first private company with the most anime in the world. Not really surprising due to it is a studio nearly as old as anime itself, Toei Animation started as Japan Animated Films in 1948. Becoming a formidable powerhouse in the 1960s, the studio was responsible for classic, influential series such as Dragon Ball, Fist of the North Star, Slam Dunk, Mazinger Z, Galaxy Express 999, One Piece, and Sailor Moon. (**Annex 3.1.2.7**)

5.2 Bi-Dimensional-Analysis

In this part we basically do we same steps for each category.
The difference would be with Genre, Theme, Producers and Studios. Those categories could have different items in one cell (because they were lists of items). So, we do a first step of splitting the information, extract the data of a column with respect to another column (so if one cell had more than one producer, we will assign the respective score and scored_by to each of them) and then put them back in a df. Then we do the next 4 steps. (**Annex 3.2.1**)

- We can check graphically with a boxplot the Score to see the median.
- Calculate the mean,median, min and max of Score.
- Calculate the mean,median, min and max of Scored_by.
- Check the coefficient of variation of Score and Scored_by.

5.2.1 Score and Type

We can check graphically the Score of each Type to see the median.
This boxplot shows that the Types with better median are TV, Movie and Specials. It requires mores analytical study. (**Annex 3.2.1.1**)
We calculate the mean,median, min and max of Score for each Source.
It shows that the Sources with better mean are TV, Movie and Specials. (**Annex 3.2.1.2**)
No we calculate the mean,median, min and max of Scored_by for each Source.
It shows that the Source with better mean are TV, ONA and Movie. (**Annex 3.2.1.3**)
We also check the coefficient of variation of Score and Scored_by. I tis possible to see those were the scores are more regular TV, Special and Music.Those that their score is more irregular like Movie, OVA and ONA. (**Annex 3.2.1.4**)

5.2.2 Score and Source

We can check grafically the Score of each Source to see the median.
This boxplot shows that the Sources with better median are Light novel, Manga and Novel.
It requieres mores analytical study. (**Annex 3.2.2.1**)
We calculate the mean,median, min and max of Score for each Source.

It shows that the Sources with better mean are Light novel, Manga and Novel. (**Annex 3.2.2.2**)
Now we calculate the mean,median, min and max of Scored_by for each Source.
It shows that the Source with better mean are Light novel, Radio and Original. (**Annex 3.2.2.3**)
We also check the coefficient of variation of Score and Scored_by.
I tis possible to see those where the scores are more regular Web novel, Picture book and Book.
Those that their score is more iregular like Music, Manga and Web manga. (**Annex 3.2.2.4**)

5.2.3 Score and Rating

We can check grafically the Score of each Source to see the median. The boxplot shows that the median are pretty much the same.
We will not take into account Hentai, because animated porno is not the goal of the company.
It requieres mores analytical study. (**Annex 3.2.3.1**)
We calculate the mean,median, min and max of Score for each Rating.
It shows that the Rating with better mean are Production R - 17+ (violence & profanity), PG-13 - Teens 13 or older and PG - Children. (**Annex 3.2.3.2**)
Now we calculate the mean,median, min and max of Scored_by for each Rating.
It shows that the Ratings with better mean are PG - Children, R - 17+ (violence & profanity) and G - All Ages. (**Annex 3.2.3.3**)
We also check the coefficient of variation of Score and Scored_by.
I tis possible to see those where the scores are more regular PG - Children or G - All Ages and those that their score are more iregular like R+ - Mild Nudity and R - 17+ (violence & profanity). (**Annex 3.2.3.4**)

5.2.4 Score and Genre

We are going to create an auxiliar dataframe with three columns Genre, Score and Scored_by.
Since the Genre column could have different Genre for one raw, we need to separate and assign to each one of them the respective Score and Scored_by. (**Annex 3.2.4.1**)
We can check grafically the Score of each Genre to see the median.
This boxplot shows that the median is pretty much the same.
But slighly better in Romance, Sci-Fi and Drama.
We will not take into account Hentai, because animated porno is not the goal of the company.
It requieres mores analytical study (**Annex 3.2.4.2**)
Now we calculate the mean,median, min and max of Score for each Genre. From the top 10 Genres.
It confirmas the boxplot, that the Genres with better mean are Romance, Sci-Fi and Drama. (**Annex 3.2.4.3**)
Now we calculate the mean,median, min and max of Scored_by for each Genre. From the top 10 Genres.
It shows that the Genres with better mean are Supernatural, Fantasy and Romance. (**Annex 3.2.4.4**)
We also check the coefficient of variation of Score and Scored_by. I tis possible to see those where the scores are more regular like Slice of Life, Sci-Fi or Romance and

those that their score are more iregular like Adventure, Fantasy and Action. (**Annex 3.2.4.5**)

5.2.5 Score and Theme

We are going to create an auxiliar dataframe with three columns Theme, Score and Scored_by.
Since the Theme column could have different Theme for one raw, we need to separate and assign to each one of them the respective Score and Scored_by. (**Annex 3.2.5.1**)
We can check grafically the Score of each Theme to see the median.
This boxplot shows that the median is pretty much the same. But slighly better in Mecha, Military and Super Power. It requieres mores analytical study. (**Annex 3.2.5.2**)
Now that we have that auxiliar dataframe, we need to unify the Theme by its name and calculate the mean,median, min and max of Score for each Theme. From the top 10 Theme.
It confirms the boxplot, that the Themes with better mean are Mecha, Military and Super Power. (**Annex 3.2.5.3**)
Now we and calculate the mean,median, min and max of Scored_by for each Theme. From the top 10 Theme.
It shows that the Theme with better mean are Anthropomorphic, School and Super Power. (**Annex 3.2.5.4**)
We also check the coefficient of variation of Score and Scored_by. I tis possible to see those where the scores are more regular like Mecha or Mythology and those that their score are more iregular like Anthropomorphic or Super Power. (**Annex 3.2.5.5**)

5.2.6 Score and Producers

We are going to create an auxiliar dataframe with three columns Producers, Score and Scored_by.
Since the Producers column could have different Producers for one raw, we need to separate and assign to each one of them the respective Score and Scored_by. (**Annex 3.2.6.1**)
We can check grafically the Score of each Producer with the median.
This boxplot shows that the median is pretty much the same.
It requieres mores analytical study. (**Annex 3.2.6.2**)
Now that we have that auxiliar dataframe, we need to unify the Producers by its name and calculate the mean,median, min and max of Score for each Producer. From the top 10 Producers.
It shows that the Producers with better mean are Bandai Visual, Fuji TV and TV Tokyo. (**Annex 3.2.6.3**)
Now we calculate the mean,median, min and max of Scored_by for each Producers. From the top 10 Studios.
It shows that the Producers with better mean are Dentsu, Movic and Aniplex. (**Annex 3.2.6.4**)
We also check the coefficient of variation of Score and Scored_by. I tis possible to see those where the scores are more regular like Fuji TV or Bandai Visual and those that their score are more iregular like Lantis or Movic. (**Annex 3.2.6.5**)

5.2.7 Score and Studios

We are going to create an auxiliar dataframe with three columns Studios, Score and Scored_by.
Since the Studio column could have different Studios for one raw, we need to separate and assign to each one of them the respective Score and Scored_by. (**Annex 3.2.7.1**)
We can check grafically the Score of each studio with the median.
This boxplot shows that the median is pretty much the same but Shanghai Animation Film Studio seems a bit different than the rest.
It requieres mores analytical study. (**Annex 3.2.7.2**)
Now that we have that auxiliar dataframe, we need to unify the studios by its name and calculate the mean,median, min and max of Score for each Studio. From the top 10 Studios.
It shows that the Studios with better mean are Production I.G, Madhouse and Studio Deen. (**Annex 3.2.7.3**)

Now we calculate the mean,median, min and max of Scored_by for each Studio. From the top 10 Studios.
It shows that the Studios with better mean are Shanghai Animation Film Studio, Madhouse and Pierrot. (**Annex 3.2.7.4**)
We also check the coefficient of variation of Score and Scored_by. I tis possible to see those where the scores are more regular like Shanghai Animation Film Studio or OLM and those that their score are more iregular like Production I.G. or Madhouse. (**Annex 3.2.7.5**)

6 Hypothesis

6.1 Hypothesis 1

[higher number of episodes = higher score]

Let's do a scatterplot to see the relationship between different variables (Score and Duration of the episodes). (**Annex 4.1.1**)

We observe some outliers that do not let us see properly the scatterplot.

There are 144 animes with more than 200 episodes in total. (**Annex 4.1.2**)

Since we have 24 thousand records, we will avoid those 144 animes in the plot. So, we will be able to see it clearly. (**Annex 4.1.3**)

It seems that there is correlation between Score and Number of Episodes. However, let's do some test to prove it.

A Spearman test will be performed to investigate the relationship between Score and Number of Episodes.

I will measure the strength of that relationship. (**Annex 4.1.4**)

A Spearman test is considered as very weak (0 to 0.19)

A Pearson test was performed to investigate the relationship between Score and Number of Episodes Results of the test showed there was a positive, weak correlation between the two variables

Next, we are going to test whether Score and Number of Episodes have a dependency relationship. (**Annex 4.1.5**)

Assumptions:

- Observations in each sample are independent and identically distributed.
- Observations in each sample can be ranked.

Interpretation

- H0: there is a dependency between the samples.
- H1: the two samples are independent.

6.1.1 Conclusion of the Hypothesis 1

Correlation test: Results of the test showed there was a positive, very weak correlation between the two variables.

Dependency relationship test: P-Value is 0, it indicates that there is no association between the two variables.

Conclusion: We reject the null hypothesis but We cannot reject the alternative hypothesis the two samples are independent.

Observations: We can say that as the number of episodes increases, it does not mean that it will have a higher value. So, the number of episodes does not influence. Or it does not mean by having many chapters, it will have better valuation.

Seems that the higher concentration of higher scores are between 12 and 25 episodes

6.2 Hypothesis 1

[longer duration of episodes = higher score]

Let's do a scatterplot to see the relationship between different variables (Score and Duration of the episodes). (**Annex 4.1.1**)

There are 4 animes with a duration of more than 200 minutes. (**Annex 4.1.2**)

We observe 4 outliers that do not let us see properly the scatterplot.

Since there are only 4 outliers, we will discard them from the plotting. (**Annex 4.1.3**)

It seems that there is correlation between Score and Duration of the episodes. However let's do some test to prove it.

A Spearman test will be perform to investigate the relationship between Score and Duration of the episodes.

I will measures the strength of that relationship. (**Annex 4.1.4**)

The Spearman rank correlation is considered as weak (0.20 to 0.39)

Results of the correlation test showed there was a positive, weak correlation between the two variables

Next, we are going to test whether Score and Duration of the episodes have a dependency relationship. (**Annex 4.1.5**)

Assumptions:

- Observations in each sample are independent and identically distributed (iid).

- Observations in each sample can be ranked.

Interpretation

- H0: there is a dependency between the samples.
- H1: the two samples are independent.

6.2.1 Conclusion of the Hypothesis 2

Correlation test: Results of the test showed there was a positive, weak correlation between the two variables.

Dependency relationship test: P-Value is 0, it indicates that there is no association between the two variables.

Conclusion: We reject the null hypothesis but We cannot reject the alternative hypothesis the two samples are independent.

Observations: We can say that as the Duration of the episodes increases, it does not mean that it will have a higher value. So, the Duration of the episodes does not influence. Or it does not mean by having more duration, it will have better valuation.

7 Other conclusions

Here we are summarized other conclusion we got while doing the analysis.

7.1 From Unidimensional Analysis

7.2 From Numerical One Dimensional-Analysis

- 75 % of the animes has 13 episodes.

- We have 24105 anime records, 5501 animes with more than 25 minutes and 18604 animes with more than 25 minutes. Basically, most of the Tv Series animes, 7345 out of 7509, has 25 or less minutes and just 164 with more than 25 minutes. And from those, 6060 have between 10 to 25 minutes.

- Most of the animes get a score between 5 to 8

- Seems that the higher concentration of higher scores are between 12 and 25 episodes

- Number 1 anime Fullmetal Alchemist: Brotherhood

- 2016 was the year that the most anime was released. This was followed by 2017

7.3 From Categorical One Dimensional-Analysis

- Most of the anime are TV type (regular series)

- Most of the anime comes from an original idea (where anime companies write up the plot and design the characters themselves)

- PG-13 - Teens 13 or older is the most popular

- Comedy Genre is the most typical

- Music, School and history theme are the ones with more animes.

- NHK, Nippon Hoso Kyokai (Japan Broadcasting Corporation), is the producer with the most anime in the world

- Toei Animation is the first private company with the most anime in the world

7.4 From Bidimensional Analysis

7.4.1 Most valued Types by users

Taking into account these results, this are the possible ways to go for.

1. If the goal is to reach as many people as possible, even though we might have irregular score, the best three Types to go for are: TV, ONA and Movie.

2. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three Types to go for are: TV, Special and Music.

   - Having the best score but not regular, the best three Types to go for are: TV, Movie and Specials.

However, taking also into account the results of the unidimensional analysis of Types:

- Specials and Music types has less animes then the rest.

- TV, Movie and OVA has mores animes then the rest

- ONA is increasing rapidly thanks to the online distribution companies like Netflix.

We could conclude that, going for TV, Movie and ONA would be the best idea

7.4.2 Best source to create an anime

Taking into account these results, this are the possible ways to go for.

3. If the goal is to reach as many people as possible, even though we might have irregular score, the best three Sources to go for are: Light novel, Radio and Original.

4. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three Sources to go for are: Web novel, Picture book and Book.

- Having the best score but not regular, the best three Sources to go for are: Light novel, Manga and Novel.

However, taking also into account the results of the unidimensional analysis of Sources:

- Picture book, Book, Radio and Web novel has almost no animes. It would be risky to go for these ones.

The best idea would be to go for Light novel, Manga and Original Sources

7.4.3 Best Rating to focus

Considering the results, this are the possible ways to go for.

5. If the goal is to reach as many people as possible, even though we might have irregular score, the best three Themes to go for are: PG - Children, R - 17+ (violence & profanity) and G - All Ages.

6. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three Themes to go for are: PG - Children or G - All Ages.

   - Having the best score but not regular, the best three Themes to go for are: R - 17+ (violence & profanity), PG-13 - Teens 13 or older and PG - Children.

7.4.4 Best Genres to use in anime

Taking into account the results, this are the possible ways to go for.

1. If the goal is to reach as many people as possible, even though we might have irregular score, the best three Themes to go for are: Supernatural, Fantasy and Romance.

2. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three Themes to go for are: Slice of Life, Sci-Fi or Romance.

   - Having the best score but not regular, the best three Themes to go for are: Romance, Sci-Fi and Drama.

7.4.5 Best themes to use in anime

Considering the results, this are the possible ways to go for.

1. If the goal is to reach as many people as possible, even though we might have irregular score, the best three Themes to go for are: Anthropomorphic, School and Super Power.

2. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three Themes to go for are: Mecha, Mythology and Music.

   - Having the best score but not regular, the best three Themes to go for are: Mecha, Military and Super Power.

7.4.6 Best producers to hire

Taking into account the results, this are the possible ways to go for.

1. If the goal is to reach as many people as possible, even though we might have irregular score, the best three producers to go for are: Dentsu, Movic and Aniplex.

2. If the goal is about the score, the best three studios to go for are: Bandai Visual, Fuji TV and TV Tokyo. This three has the best score and also happen to have the more regular scores.

7.4.7 Best studios to hire

It will depend of the goal. Considering the results, this are the possible ways to go for.

1. If the goal is to reach as many people as possible, even though we might have irregular score, the best three studios to go for are: Shanghai Animation Film Studio, Madhouse and Pierrot.

2. If the goal is about the score there are two options:

   - Not Having the best score but the most regular one, the best three studios to go for are: Shanghai Animation Film Studio, OLM and Toei Animation.

   - Having the best score but not regular, the best three studios to go for are: Production I.G, Madhouse and Studio Deen.

8 References

MyAnimeList.net	https://myanimelist.net/
from Data to Viz	https://www.data-to-viz.com/#connectedscatter
Seaborn Documentation	https://seaborn.pydata.org/
Matplotlib Documentation	https://matplotlib.org/
Data Visualization con pandas y seaborn	https://medium.com/ironhack/data-visualization-con-pandas-y-seaborn-1044906af34f
The Python Graph Gallery	https://www.python-graph-gallery.com/
Stack Overflow	https://stackoverflow.com/
codificandobits	https://www.codificandobits.com/blog/analisis-exploratorio-de-datos/
Towards data science	https://towardsdatascience.com/beautifying-the-messy-plots-in-python-solving-common-issues-in-seaborn-7372e6479fb
pandas documentation	https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.explode.html

9 Annexes

9.1 Annex 1

9.1.1 Annex 1.1

```
for page in range (1,n_pages +1):
    r_page = requests.get(url + '?page=' + str(page)) # request to a web page (url)
    content = r_page.json()
    print (page)
    data = content["data"]
    time.sleep(1)
```

9.1.2 Annex 1.2

```
for char in data:

    try: # First try yo check if the page exist or not
        # Creation of the necessary dictionary o store the values in each loop # We specify which information to get in each Item

    # Ending of the first try specifying the error
    except:
        if r_page.status_code == 429: #If there is a 429 error we show it on screen and tell us the respuesta.reason
            print (f"El código de estado de la petición es: {r_page.status_code}. Estatus {r_page.reason}. No se puede recoger información de la página {id}\n")
        else:
            #If there is a any other error we show it on screen and tell us the respuesta.reason
            print (f"El código de estado de la petición es: {r_page.status_code}. Estatus {r_page.reason}. No se puede recoger información de la página {id}\n")
        continue
```

9.1.3 Annex 1.3

```
def try_it(i):
    try:
        return i["name"]
    except:
        return None
```

9.1.4 Annex 1.4

```
anime_dict = {"Cover" : char["images"]["jpg"]["large_image_url"] if char["images"]["jpg"]["large_image_url"]  else None,
              "English_Title" : char["title"] if char["title"]  else None,
              "Japanses_Title" : char["title_japanese"] if char["title_japanese"]  else None,
              "Type" : char["type"] if char["type"]  else None,
              "Source" : char["source"] if char["status"] else None,
              "Audience" : [try_it(i) for i in char["demographics"]], # List comprehension calling the Def try_it
              "N_Episodes" : (int(char["episodes"])) if char["episodes"] else 0,
              "Duration" : char["duration"] if char["duration"]  else None,
              "Rating" : char["rating"] if char["rating"] else None,
              "Score" : char["score"] if char["score"]  else None,
              "Scored_by" : char["scored_by"] if char["scored_by"]  else None,
              "Rank" : (int(char["rank"])) if char["rank"] else None,
              "Season" : char["season"] if char["season"] else None,
              "Genre" : [try_it(i) for i in char["genres"]],# List comprehension calling the Def try_it
              "Theme" : [try_it(i) for i in char["themes"]],# List comprehension calling the Def try_it
              "Released" : (int(char["aired"]["prop"]["from"]["year"])) if char["aired"]["prop"]["from"]["year"] else None, # If else in one line
              "Studios" : [try_it(i) for i in char["studios"]],# List comprehension calling the Def try_it
              "Producers" : [try_it(i) for i in char["producers"]]# List comprehension calling the Def try_it
              }

anime_list.append(anime_dict) # Append the loop info to anime_list
```

9.1.5 Annex 1.5

```
# We create df from anime_list and save it in a csv file adding actual date and time variables to the name
anime_df = pd.DataFrame(anime_list)
anime_csv = os.path.join(data_folder,"anime_" + actual_date+ "_" +current_time + ".csv")# Saving the image to the images folder
anime_df.to_csv(anime_csv, sep = ';', index = False)
print(f'anime_{actual_date}{current_time}.csv created\n\n')
```

9.1.6 Annex 1.6

```
'''
Preparing folder variables.
'''
os.chdir(os.path.dirname(sys.path[0])) # This command makes the notebook the main path and can work in cascade.
main_folder = sys.path[0]
data_folder = (main_folder + "\data")

'''
Creating time variables.
'''
current_time = time.strftime("%H_%M_%S",time.localtime())
date = datetime.now()
actual_date = date.strftime("%Y_%m_%d")
```

9.2 Annex 2

9.2.1 Annex 2.1

9.2.1.1 Annex 2.1.1

```
#Library imports
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
import sys
import seaborn as sns
from collections import Counter
from utils import utils
from scipy.stats import chi2_contingency
from sklearn.linear_model import LinearRegression
import warnings
import pickle
warnings.simplefilter(action='ignore', category=FutureWarning)
```

9.2.1.2 Annex 2.1.2

	Cover	English_Title	Japanses_Title	Type	Source	Audience	N_Episodes	Duration	Rating	Score	Scored_by	Rank	Season	Genre	Theme	Released	Studios	Producers
5732	https://cdn.myanimelist.net/images/anime/1240/...	Yamato Takeru: After War	ヤマトタケル ～After War～	OVA	Unknown	[]	2	23 min per ep	PG-13 - Teens 13 or older	6.27	215.0	7861.0	NaN	['Action', 'Adventure', 'Fantasy', 'Sci-Fi']	['Mecha', 'Space']	1995.0	['Nippon Animation']	[]
20342	https://cdn.myanimelist.net/images/anime/1432/...	Tunshu Zhen Lequ Sanzijing	豚鼠鎮乐趣三字经	TV	Original	['Kids']	100	14 min per ep	PG - Children	NaN	NaN	16897.0	NaN	['Fantasy']	['Anthropomorphic']	2017.0	[]	[]
1811	https://cdn.myanimelist.net/images/anime/7/688...	Shizuku	しずく	Movie	Original	[]	1	4 min	PG - Children	5.62	1461.0	10963.0	NaN	['Comedy']	[]	1965.0	['Mushi Production']	[]
22675	https://cdn.myanimelist.net/images/anime/1535/...	Sleepless: A Midsummer Night's Dream - The Ani...	SLEEPLESS ～A Midsummer Night's Dream～ The Anim...	OVA	Visual novel	[]	2	16 min per ep	Rx - Hentai	5.71	1052.0	NaN	NaN	['Hentai']	[]	2022.0	['BreakBottle']	['Showten']
4298	https://cdn.myanimelist.net/images/anime/13/12...	Koisuru Tenshi Angelique: Chibi Character Adve...	恋する天使アンジェリーク ちびキャラ Adventure	Special	Unknown	[]	1	4 min	G - All Ages	5.92	227.0	9685.0	NaN	['Adventure', 'Comedy']	[]	2006.0	[]	[]

9.2.1.3 Annex 2.1.3

```
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Cover           24105 non-null  object
 1   English_Title   24105 non-null  object
 2   Japanses_Title  23914 non-null  object
 3   Type            24029 non-null  object
 4   Source          24105 non-null  object
 5   Audience        24105 non-null  object
 6   N_Episodes      24105 non-null  int64
 7   Duration        24105 non-null  object
 8   Rating          23315 non-null  object
 9   Score           14874 non-null  float64
 10  Scored_by       14874 non-null  float64
 11  Rank            21515 non-null  float64
 12  Season          5348 non-null   object
 13  Genre           24105 non-null  object
 14  Theme           24105 non-null  object
 15  Released        22885 non-null  float64
 16  Studios         24105 non-null  object
 17  Producers       24105 non-null  object
dtypes: float64(4), int64(1), object(13)
memory usage: 3.3+ MB
```

```
Cover              0
English_Title      0
Japanses_Title   191
Type              76
Source             0
Audience           0
N_Episodes         0
Duration           0
Rating           790
Score           9231
Scored_by       9231
Rank            2590
Season         18757
Genre              0
Theme              0
Released        1220
Studios            0
Producers          0
dtype: int64
```

```
Season         77.813732
Score          38.294960
Scored_by      38.294960
Rank           10.744659
Released        5.061191
Rating          3.277328
Japanses_Title  0.792367
Type            0.315287
Studios         0.000000
Theme           0.000000
Genre           0.000000
Cover           0.000000
English_Title   0.000000
Duration        0.000000
N_Episodes      0.000000
Audience        0.000000
Source          0.000000
Producers       0.000000
dtype: float64
```

9.2.2 Annex 2.2

9.2.3 Annex 2.3

9.2.3.1 Annex 2.3.1

```
non_numeric = ["Audience","Genre","Theme","Studios","Producers"]
for column in non_numeric:
    df_copy[column] = df_copy[column].apply(eval).str.join(',') #remove brackets and commas

df_copy.sample(5)
```

9.2.3.2 Annex 2.3.2

```
# Replacing empty space with NaN
list_not_num = ["Audience","Genre","Theme","Studios","Producers"]
df_copy[list_not_num] = df_copy[list_not_num].apply(lambda x: x.str.strip() if isinstance(x, str) else x).replace('', np.nan)
```

9.2.3.3 Annex 2.3.3

```
Season          77.813732
Audience        60.912674
Producers       53.839452
Theme           46.081726
Studios         43.410081
Scored_by       38.294960
Score           38.294960
Genre           19.207633
Rank            10.744659
Released         5.061191
Rating           3.277328
Japanses_Title   0.792367
Type             0.315287
Duration         0.000000
English_Title    0.000000
N_Episodes       0.000000
Source           0.000000
Cover            0.000000
dtype: float64
```

9.2.4 Annex 2.4

9.2.4.1 Annex 2.4.1

```
list_num = ["Rank",'Score','Scored_by',"Released"]
for cat in list_num:
    null_num = df_copy[cat].isna().sum()
    number = df_copy[cat][df_copy[cat] == 0].count()
    print("In the column ndamed",cat,"there are:",number,"of zero values")
    print("In the column ndamed",cat,"there are:",null_num,"of null values")
```

```
In the column ndamed Rank there are: 0 of zero values
In the column ndamed Rank there are: 2590 of null values
In the column ndamed Score there are: 0 of zero values
In the column ndamed Score there are: 9231 of null values
In the column ndamed Scored_by there are: 0 of zero values
In the column ndamed Scored_by there are: 9231 of null values
In the column ndamed Released there are: 0 of zero values
In the column ndamed Released there are: 1220 of null values
```

9.2.4.2 Annex 2.4.2

```
#Replacing nan with ZEROS in numerical columns
list_num = ["Rank"]
for cat in list_num:
    df_copy[list_num] = df_copy[list_num].fillna(0)
```

9.2.4.3 Annex 2.4.3

```
df_copy['Released'] = df_copy['Released'].interpolate(method = "spline", order = 1,  limit_direction = "both", downcast = "infer")
df_copy['Score'] = df_copy['Score'].interpolate(method = "spline", order = 3,  limit_direction = "both", downcast = "infer")
df_copy['Scored_by'] = df_copy['Scored_by'].interpolate(method = "spline", order = 3,  limit_direction = "both", downcast = "infer")
```

9.2.4.4 Annex 2.4.4

```
df_copy['Scored_by'].agg(['min', 'max'])
```

```
-1.581879e+06
 4.128912e+06
Scored_by, dtype: float64
```

```
df_copy['Scored_by'] = np.abs(df_copy['Scored_by']) # convert the Scored_by column to absolute value using abs from numpy
```

9.2.4.5 Annex 2.4.5

```
utils.predict(df_copy)
```

```
list_num = ["Rank",'Score','Scored_by',"Released"]
for cat in list_num:
    null_num = df_copy[cat].isna().sum()
    number = df_copy[cat][df_copy[cat] == 0].count()
    print("In the column ndamed",cat,"there are:",number,"of zero values")
    print("In the column ndamed",cat,"there are:",null_num,"of null values")
```

```
In the column ndamed Rank there are: 0 of zero values
In the column ndamed Rank there are: 0 of null values
In the column ndamed Score there are: 0 of zero values
In the column ndamed Score there are: 0 of null values
In the column ndamed Scored_by there are: 0 of zero values
In the column ndamed Scored_by there are: 0 of null values
In the column ndamed Released there are: 0 of zero values
In the column ndamed Released there are: 0 of null values
```

```
'''
To predict the missing information in the rank column
'''
def predict(df):

    # df with the zero values in rank column
    with_zeros = df[['Score','Scored_by','Rank']].copy()

    # setup x and y for training
    # drop data with zero in the row
    clean_df = with_zeros[with_zeros.Rank  != 0]

    # separate variables into my x and y
    x = clean_df[['Score','Scored_by']].values
    y = clean_df['Rank'].values

    # fit my model (adjusting the parameters in the model to improve accuracy.)
    lm = LinearRegression()
    lm.fit(x, y)

    # get the rows I am trying to do my prediction on
    predict_x = with_zeros[with_zeros['Rank'] == 0][['Score','Scored_by']].values

    # perform my prediction
    lm.predict(predict_x)

    # Get index of missing data
    missing_index = with_zeros[with_zeros['Rank'] == 0].index

    # Replace
    df.loc[missing_index, 'Rank'] = lm.predict(predict_x)
```

9.2.4.6 Annex 2.4.6

```
df_copy[['Scored_by','Rank','Released']] = df_copy[['Scored_by','Rank','Released']].astype('int') # change the type of the column to integer
```

9.2.5 Annex 2.5

9.2.5.1 Annex 2.5.1

```
df_copy['Audience'].str.split(',').explode().value_counts()
```

```
Kids        5785
Shounen     2035
Seinen       915
Shoujo       700
Josei        105
Name: Audience, dtype: int64
```

```
df_copy['Rating'].str.split(',').explode().value_counts()
```

```
PG-13 - Teens 13 or older          8192
G - All Ages                       7230
PG - Children                      3989
Rx - Hentai                        1455
R - 17+ (violence & profanity)     1376
R+ - Mild Nudity                   1073
Name: Rating, dtype: int64
```

```
df_copy.drop(["Audience"] , axis = 1 , inplace = True)
```

9.2.6 Annex 2.6

9.2.6.1 Annex 2.6.1

```
# Dorpping unncesseray columns
df_copy.drop(["Cover" ,"Japanses_Title","Season",] , axis = 1 , inplace = True)
```

9.2.7 Annex 2.7

9.2.7.1 Annex 2.7.1

```
list_num = ["Type",'Rating','Genre',"Released",'Studios',"Producers","Theme"]
for cat in list_num:
    df_copy[cat] = df_copy[cat].fillna('Unknown')
```

9.2.8 Annex 2.8

9.2.8.1 Annex 2.8.1

```
# Get count of animes with N_Episodes equal to zero
count = df_copy["N_Episodes"][df_copy["N_Episodes"] == 0].count()
print("There are",count,"animes with N_Episodes equal to zero")
```

```
There are 867 animes with N_Episodes equal to zero
```

9.2.8.2 Annex 2.8.2

```
#We can see that N_Episodes has a minumin of 0 episodes. That cannot be possible, in this case we are going to Change zero values to the previous column value
N_Episodes = df_copy['N_Episodes']
N_Episodes.replace(to_replace = 0,  method='ffill', inplace=True)
```

9.2.8.3 Annex 2.8.3

```
# Get count of animes with N_Episodes equal to zero
count = df_copy["N_Episodes"][df_copy["N_Episodes"] == 0].count()
print("There are",count,"animes with N_Episodes equal to zero")
```

```
There are 0 animes with N_Episodes equal to zero
```

9.2.9 Annex 2.9

9.2.9.1 Annex 2.9.1

To finalize, let's change the format in the duration column. Let's leave the columns just with the duration in minutes

```
utils.to_minutes(df_copy)
```

```
'''
Convert the string time to just minutes
'''
def to_minutes(df):
    df['Duration'] = df['Duration'].apply(lambda positions : positions.split(' ')).apply(lambda positions : positions[0] if len(positions) <= 1 else
            (int(positions[0]) / 60 if positions[1] == 'sec' else # If position 1 is sec, then we devide position 1 by 60
            (int(positions[0]) if positions[1] == 'min' else # if potision 1 is min, then pring position 0
            (int(positions[0]) * 60 if positions[1] == 'hr' and len(positions) == 2 else # If position 1 is hr and lenght of the string equals 2 (3 w
            (int(positions[0]) * 60 + int(positions[2]) if positions[1] == 'hr' and positions[2] != 'per' and positions[2] != 'min'  else # If positi
            int(positions[0]) * 60  )))))
```

9.2.9.2 Annex 2.9.2

```
df_copy['Duration'] = df_copy['Duration'].replace('Unknown', np.nan)
df_copy['Duration'] = df_copy['Duration'].interpolate(method = "spline", order = 1,  limit_direction = "both", downcast = "infer")
```

```
df_copy[['Duration']] = df_copy[['Duration']].astype('int') # change the type of the column to integer
```

9.2.9.3 Annex 2.9.3

```
Duration = df_copy['Duration']
Duration.replace(to_replace = 0,  method='ffill', inplace=True)
```

9.2.10 Annex 2.10

9.2.10.1 Annex 2.10.1

```
1  print(((df_copy.isnull().sum() / len(df_copy))*100).sort_values(ascending = False))
2  print(f"Total number of records: {len(df_copy)}")
```

```
English_Title     0.0
Type              0.0
Source            0.0
N_Episodes        0.0
Duration          0.0
Rating            0.0
Score             0.0
Scored_by         0.0
Rank              0.0
Genre             0.0
Theme             0.0
Released          0.0
Studios           0.0
Producers         0.0
dtype: float64
Total number of records: 24105
```

9.2.10.2 Annex 2.10.2

```
1  df_copy.sample(5) # printing a sample
```

Python

	English_Title	Type	Source	N_Episodes	Duration	Rating	Score	Scored_by	Rank	Genre	Theme	Released	Studios	Producers
6753	Crayon Shin-chan Movie 20: Arashi wo Yobu! Ora...	Movie	Manga	1	110	G - All Ages	6.72	2716	5293	Comedy	Unknown	2012	Shin-Ei Animation	Unknown
9525	Persona 4 the Golden Animation: Thank you Mr. ...	Special	Game	1	16	PG-13 - Teens 13 or older	6.50	4696	6533	Adventure,Drama,Mystery,Supernatural	Unknown	2014	A-1 Pictures	Unknown
7541	Hiiro no Kakera: Totsugeki! Tonari no Ikemenzu	OVA	Visual novel	1	11	PG-13 - Teens 13 or older	6.71	2598	5378	Comedy,Fantasy	Unknown	2013	Studio Deen	Bandai Visual,Lantis
7794	Hello Kitty no Kurumi Wari Ningyou	OVA	Unknown	1	15	G - All Ages	5.85	147	9984	Fantasy	Unknown	2001	Unknown	Sanrio
3070	Shin Onimusha: Dawn of Dreams the Story	OVA	Game	1	120	PG-13 - Teens 13 or older	5.90	407	9784	Action	Historical,Martial Arts,Mythology,Samurai	2006	Unknown	Capcom

9.2.10.3 Annex 2.10.3

```
df_copy.describe(include = "all")
```

	English_Title	Type	Source	N_Episodes	Duration	Rating	Score	Scored_by	Rank	Genre	Theme	Released	Studios	Producers
count	24105	24105	24105	24105.000000	24105.000000	24105	24105.000000	2.410500e+04	24105.000000	24105	24105	24105.000000	24105	24105
unique	24050	7	17	NaN	NaN	7	NaN	NaN	NaN	1008	800	NaN	1437	4237
top	Genshin Impact	TV	Original	NaN	NaN	PG-13 - Teens 13 or older	NaN	NaN	NaN	Unknown	Unknown	NaN	Unknown	Unknown
freq	6	7509	8920	NaN	NaN	8192	NaN	NaN	NaN	4630	11108	NaN	10464	12978
mean	NaN	NaN	NaN	15.234972	22.656669	NaN	6.453147	5.810970e+04	10794.168720	NaN	NaN	2007.544368	NaN	NaN
std	NaN	NaN	NaN	49.978538	26.334477	NaN	0.718209	2.356684e+05	5918.284321	NaN	NaN	14.557824	NaN	NaN
min	NaN	NaN	NaN	1.000000	1.000000	NaN	1.850000	0.000000e+00	1.000000	NaN	NaN	1917.000000	NaN	NaN
25%	NaN	NaN	NaN	1.000000	5.000000	NaN	6.210000	6.740000e+02	6003.000000	NaN	NaN	2002.000000	NaN	NaN
50%	NaN	NaN	NaN	2.000000	18.000000	NaN	6.530696	3.897000e+03	10551.000000	NaN	NaN	2012.000000	NaN	NaN
75%	NaN	NaN	NaN	13.000000	25.000000	NaN	6.650000	2.706400e+04	15655.000000	NaN	NaN	2017.000000	NaN	NaN
max	NaN	NaN	NaN	3057.000000	1440.000000	NaN	9.110000	4.128912e+06	28237.000000	NaN	NaN	2024.000000	NaN	NaN

9.2.10.4 Annex 2.10.4

```
# Get count of values greater than 2022 in the column 'Released'
count = df_copy["Released"][df_copy["Released"] > 2022].count()
print("There are",count,"animes after 2023 for future releases")
```

```
There are 206 animes after 2023 for future releases
```

9.2.11 Annex 2.11

9.2.11.1 Annex 2.11.1

```
fichero = open(data_folder + "/" +"anime.pkl","wb")
pickle.dump(df_copy,fichero)
fichero.close()
```

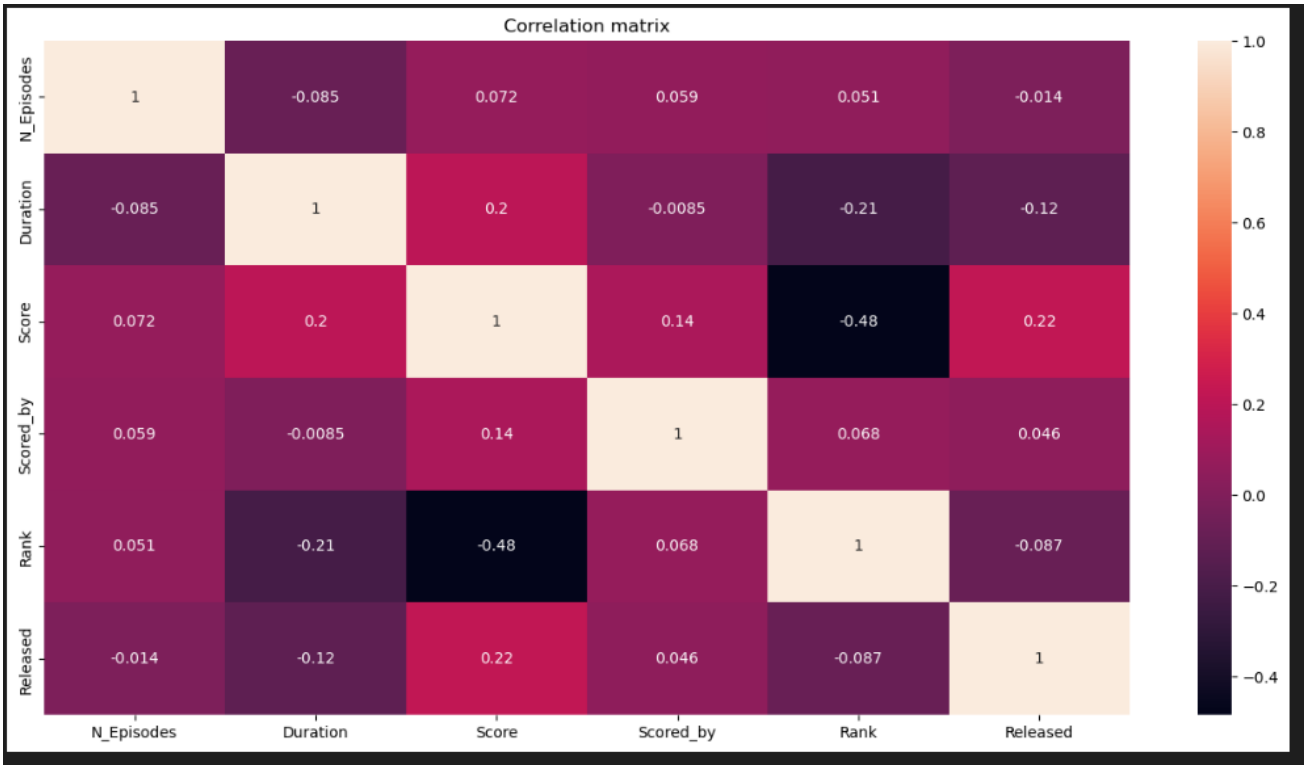
9.2.12 Annex 2.12

9.2.12.1 Annex 2.12.1

```
fichero = open(data_folder + "/" +"anime.pkl","rb")
df_copy = pickle.load(fichero)
fichero.close()
```

9.3 Annex 3

```
# Checking possible correlations for future studies.
plt.rc("figure", figsize=(16,8))

corr = df_copy.corr()
sns.heatmap(corr, annot=True)
plt.title('Correlation matrix')
plt.savefig(os.path.join(img_folder, 'Correlation matrix.png'),dpi=600)# Saving the image to the images folder
plt.show()
plt.close()# Close the plot
```



9.3.1 Annex 3.1

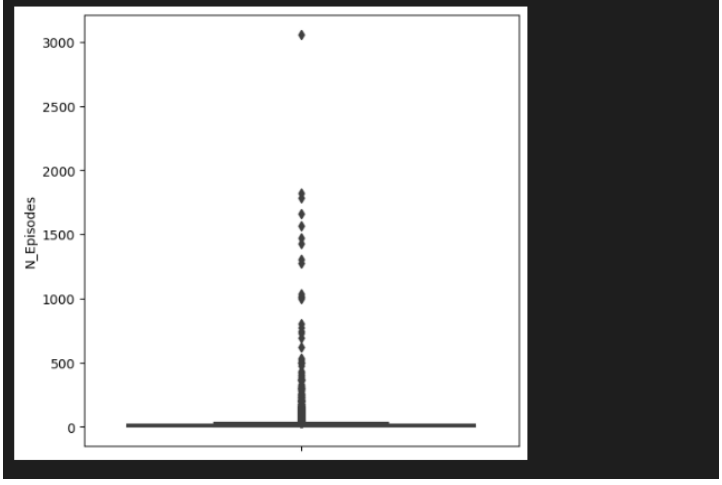
9.3.1.1 Annex 3.1.1

9.3.1.1.1 Annex 3.1.1.1

```
1 df_copy["N_Episodes"].describe()
```

```
count    24105.000000
mean        15.234972
std         49.978538
min          1.000000
25%          1.000000
50%          2.000000
75%         13.000000
max       3057.000000
Name: N_Episodes, dtype: float64
```

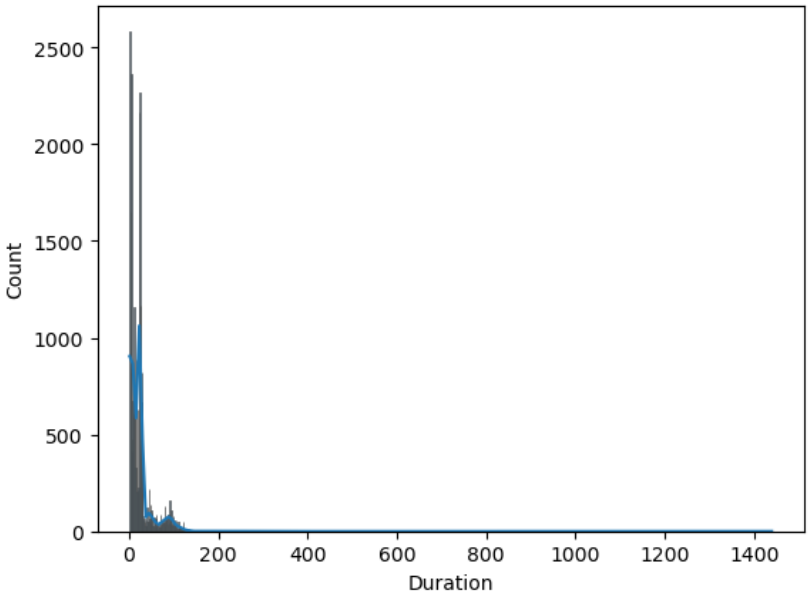
```
1 utils.box(df_copy,'N_Episodes') # Hacer una escala logaricmica  numpy loc de n_pisodios+1
```



9.3.1.1.2 Annex 3.1.1.2

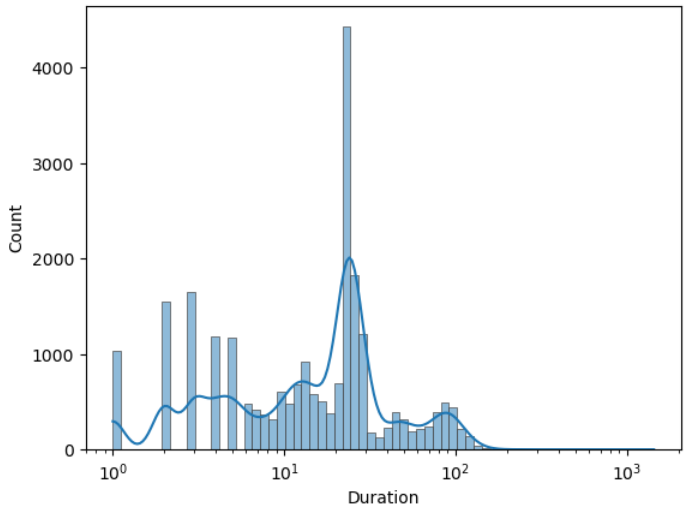
9.3.1.1.2.1 Annex 3.1.1.2.1

```
sns.histplot(data = df_copy , x = 'Duration', kde = True, palette="light:m_r", log_scale=False, edgecolor=".3", linewidth=.5)
plt.savefig(os.path.join(img_folder + "/" + 'histplot_' + "Duration" + '.png'),dpi=600)# Saving the image to the images folder
```



9.3.1.1.2.2 Annex 3.1.1.2.2

```
sns.histplot(data = df_copy , x = 'Duration', kde = True, palette="light:m_r", log_scale=True, edgecolor=".3", linewidth=.5)
plt.savefig(os.path.join(img_folder + "/" + 'histplot_log_scale_' + "Duration" + '.png'),dpi=600)# Saving the image to the images folder
```



9.3.1.1.2.3 Annex 3.1.1.2.3

```
1  df_copy["Duration"].describe()
✓ 0.3s

count    24105.000000
mean        22.656669
std         26.334477
min          1.000000
25%          5.000000
50%         18.000000
75%         25.000000
max       1440.000000
Name: Duration, dtype: float64
```

```
# Get count of animes with more than 25 minutes
count = df_copy["Duration"][df_copy["Duration"] >25].count()
print("There are",count,"animes with more than 25 minutes")
```
✓ 0.5s

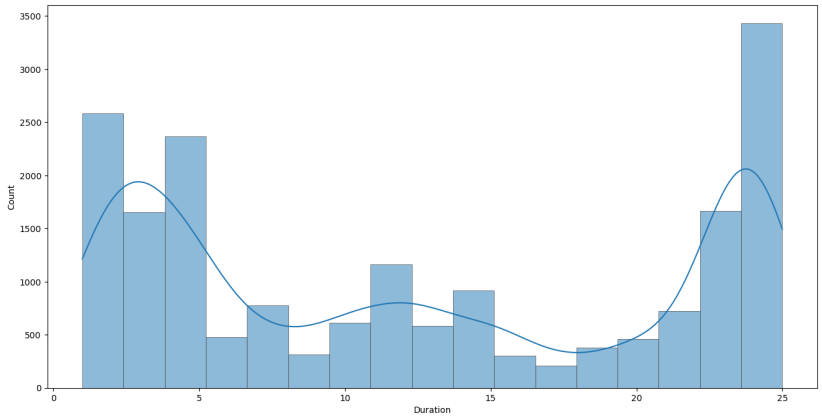
```
There are 5501 animes with more than 25 minutes
```

```
# Get count of animes with more than 25 minutes
count = df_copy["Duration"][df_copy["Duration"] <= 25].count()
print("There are",count,"animes with lee than 26 minutes")
```
✓ 0.4s

```
There are 18604 animes with lee than 26 minutes
```

9.3.1.1.2.4 Annex 3.1.1.2.4

```
unti26 = df_copy[df_copy["Duration"] < 26]
sns.histplot(data = unti26 , x = 'Duration', kde = True, palette="light:m_r", log_scale=False, edgecolor=".3", linewidth=.5)
plt.savefig(os.path.join(img_folder + "/" + 'histplot_less_26_' + "Duration" + '.png'),dpi=600)# Saving the image to the images folder
```



9.3.1.1.2.5 Annex 3.1.1.2.5

```
# Get count of animes with more than 26 minutes and are tv series
count = df_copy["Duration"][(df_copy["Duration"] > 25) & (df_copy["Type"] == "TV")].count()
print("There are",count,"animes with more then 25 minutes and are Tv Series")
```
✓ 0.4s

```
There are 164 animes with more then 25 minutes and are Tv Series
```

```
# Get count of animes with less than 26 minutes and are tv series
count = df_copy["Duration"][(df_copy["Duration"] <= 25) & (df_copy["Type"] == "TV")].count()
print("There are",count,"animes with equal or same as 25 minutes and are Tv Series")
```
✓ 0.4s

```
There are 7345 animes with equal or same as 25 minutes and are Tv Series
```

```
# Get count of animes with more than 10 minutes and are tv series
count = df_copy["Duration"][(df_copy["Duration"] >= 10) & (df_copy["Type"] == "TV")].count()
print("There are",count,"animes with equal or more than 10 minutes and are Tv Series")
```
✓ 0.4s

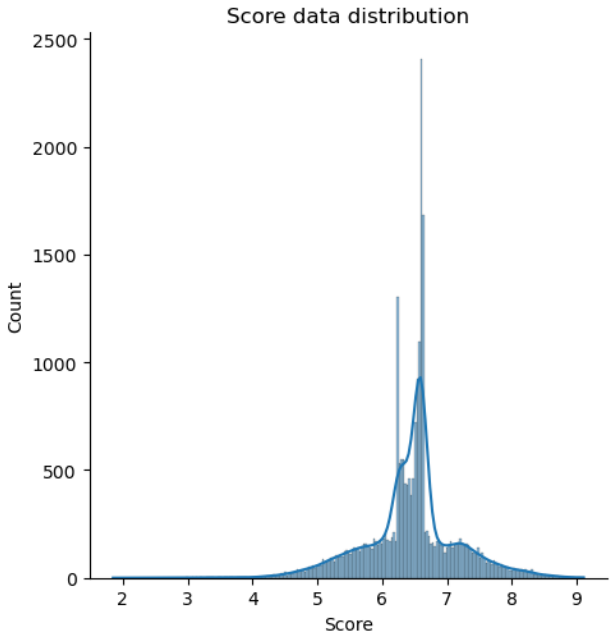
```
There are 6060 animes with equal or more than 10 minutes and are Tv Series
```

```
# Get count of animes that belongs to TV series
tv = df_copy["Type"][df_copy["Type"] == "TV"].count()
print("There are",tv,"animes that belongs to TV series")
```
✓ 0.3s

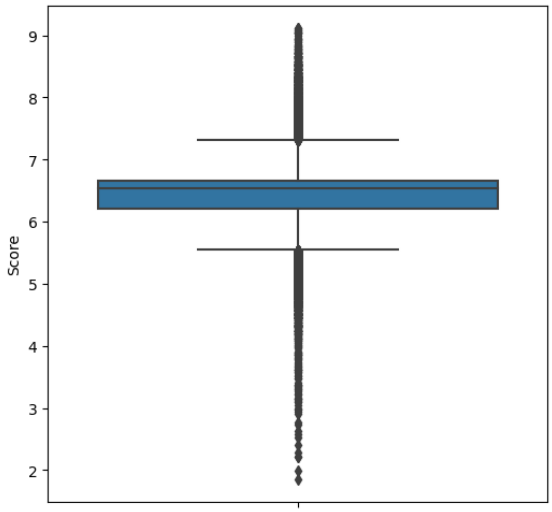
```
There are 7509 animes that belongs to TV series
```

9.3.1.1.3 Annex 3.1.1.3

9.3.1.1.3.1 Annex 3.1.1.3.1



9.3.1.1.3.2 Annex 3.1.1.3.2



9.3.1.1.3.3 Annex 3.1.1.3.3

```
print("There are",len(df_copy[(df_copy['Score'] >= 5) & (df_copy['Score'] <= 6)]),"animes between 5 and 6 points")
print("There are",len(df_copy[df_copy['Score'].between(6,7)]),"animes between 6 and 7 points")
print("There are",len(df_copy[df_copy['Score'].between(7,8)]),"animes between 7 and 8 points")
print("There are",len(df_copy[(df_copy['Score'] >= 8) & (df_copy['Score'] <= 9)]),"animes between 8 and 9 points")
```

9.3.1.1.4 Annex 3.1.1.4

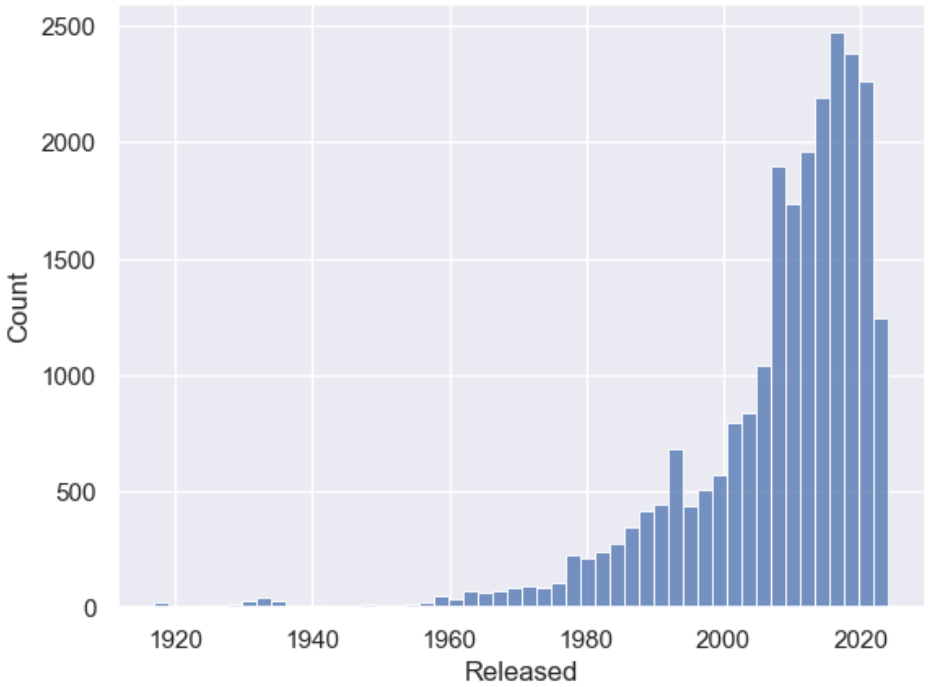
```
df_rank_top = df_copy[df_copy["Rank"] < 21].sort_values(by=["Rank"], ascending=True)
df_rank_top
```

English_Title	Type	Source	N_Episodes	Duration	Rating	Score	Scored_by	Rank	Genre	Theme	Released	Studios	Producers
Bleach: Sennen Kessen-hen	TV	Manga	13	24	R - 17+ (violence & profanity)	9.11	80321	1	Action,Adventure,Fantasy	Unknown	2022	Pierrot	TV Tokyo,Aniplex,Dentsu,Shueisha
Fullmetal Alchemist: Brotherhood	TV	Manga	64	24	R - 17+ (violence & profanity)	9.11	1933742	2	Action,Adventure,Drama,Fantasy	Military	2009	Bones	Aniplex,Square Enix,Mainichi Broadcasting Syst...
Kaguya-sama wa Kokurasetai: Ultra Romantic	TV	Manga	13	23	PG-13 - Teens 13 or older	9.09	373142	3	Comedy,Romance	Psychological,School	2022	A-1 Pictures	Aniplex,Mainichi Broadcasting System,Magic Cap...
Steins;Gate	TV	Visual novel	24	24	PG-13 - Teens 13 or older	9.08	1286088	4	Drama,Sci-Fi,Suspense	Psychological,Time Travel	2011	White Fox	Frontier Works,Media Factory,Movic,AT-X,Kadoka...
Gintama°	TV	Manga	51	24	PG-13 - Teens 13 or older	9.07	227495	5	Action,Comedy,Sci-Fi	Gag Humor,Historical,Parody,Samurai	2015	Bandai Namco Pictures	TV Tokyo,Aniplex,Dentsu
Shingeki no Kyojin Season 3 Part 2	TV	Manga	10	23	R - 17+ (violence & profanity)	9.06	1386387	6	Action,Drama	Gore,Military,Survival	2019	Wit Studio	Production I.G,Dentsu,Mainichi Broadcasting Sy...
Gintama'	TV	Manga	51	24	PG-13 - Teens 13 or older	9.05	217364	7	Action,Comedy,Sci-Fi	Gag Humor,Historical,Parody,Samurai	2011	Sunrise	TV Tokyo,Aniplex,Dentsu,Trinity Sound,Miracle ...
Gintama: The Final	Movie	Manga	1	104	PG-13 - Teens 13 or older	9.05	56776	8	Action,Comedy,Drama,Sci-Fi	Gag Humor,Historical,Parody,Samurai	2021	Bandai Namco Pictures	TV Tokyo,Warner Bros. Japan
Gintama': Enchousen	TV	Manga	13	24	PG-13 - Teens 13 or older	9.04	151944	9	Action,Comedy,Sci-Fi	Gag Humor,Historical,Parody,Samurai	2012	Sunrise	TV Tokyo,Aniplex,Dentsu,Shueisha,Miracle Bus
Hunter x Hunter (2011)	TV	Manga	148	23	PG-13 - Teens 13 or older	9.04	1569486	10	Action,Adventure,Fantasy	Unknown	2011	Madhouse	VAP,Nippon Television Network,Shueisha

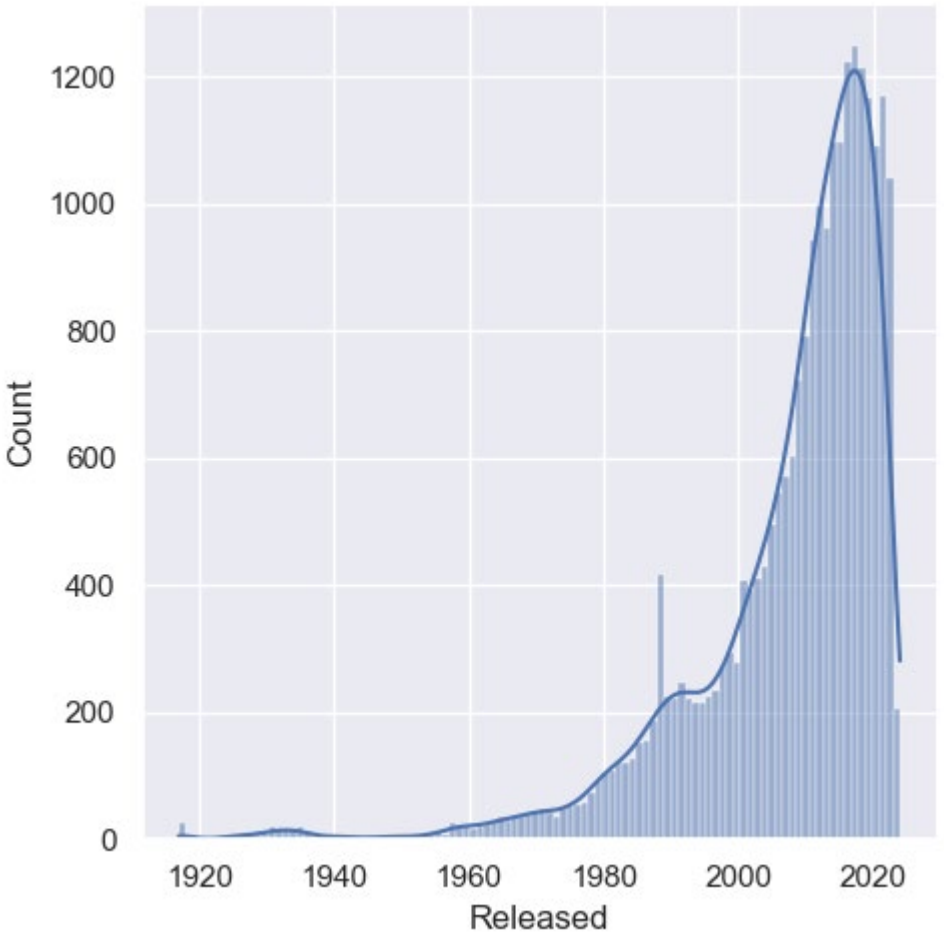
9.3.1.1.5 Annex 3.1.1.5

9.3.1.1.5.1 Annex 3.1.1.5.1

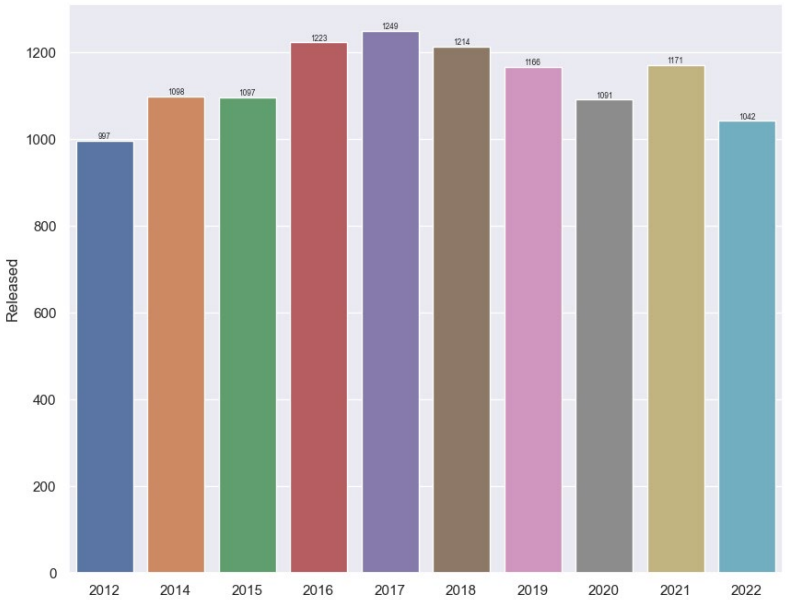
```
# set a grey background (use sns.set_theme() if seaborn version 0.11.0 or above)
sns.set(style="darkgrid")

sns.histplot(data=df_copy, x="Released", bins=50)

plt.savefig(os.path.join(img_folder + "/" + 'histplot_Released.png'),dpi=600)# Saving the image to the images folder
plt.show()
```



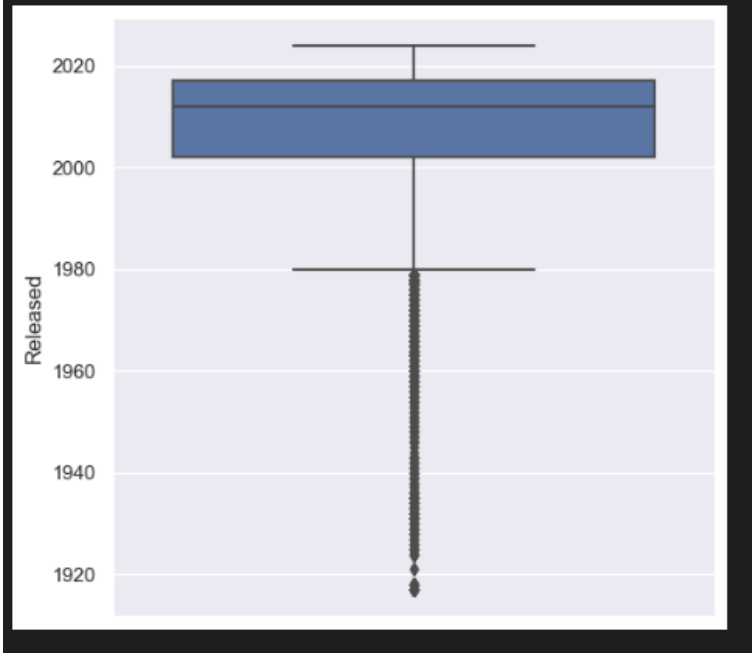

```
sns.displot(data = df_copy , x = 'Released' , kde = True) #anderson darling
plt.savefig(os.path.join(img_folder + "/" + 'displot_Released.png'),dpi=600)# Saving the image to the images folder
```




```
utils.barplot_top10(df_copy['Released'],'Released')
```

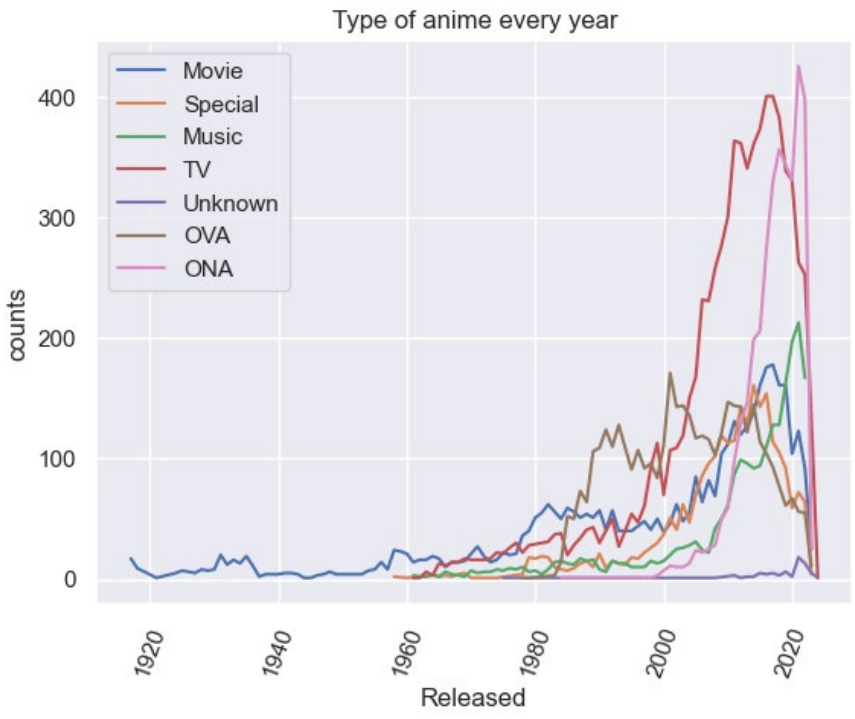


```
1  utils.box(df_copy,'Released')
```



9.3.1.1.5.2 Annex 3.1.1.5.2

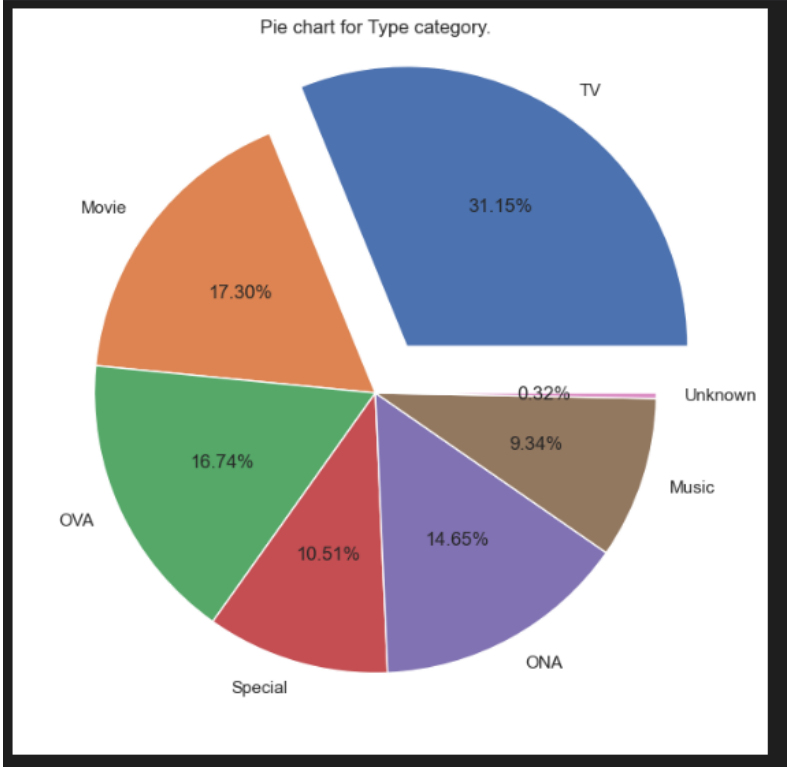
```
df_type_year =(df_copy.assign(genre=df_copy['Type'])
    .groupby(['Released','Type']).size()
    .reset_index(name='counts')
)

sns.lineplot(data=df_type_year, x='Released', y='counts', hue='Type')
plt.tick_params(axis='x', labelrotation = 70)
plt.title('Type of anime every year')
plt.legend(loc='upper left')
plt.savefig(os.path.join(img_folder, 'Lineplot_Type_by_year.png'),dpi=600)# Saving the image to the images folder
plt.show() # Show graphic
```



9.3.1.2 Annex 3.1.1.2

9.3.1.2.1 Annex 3.1.2.1

```
1  utils.pieplot(df_copy["Type"],"Type")
```



9.3.1.2.2 Annex 3.1.2.2

9.3.1.2.2.1 Annex 3.1.2.2.1

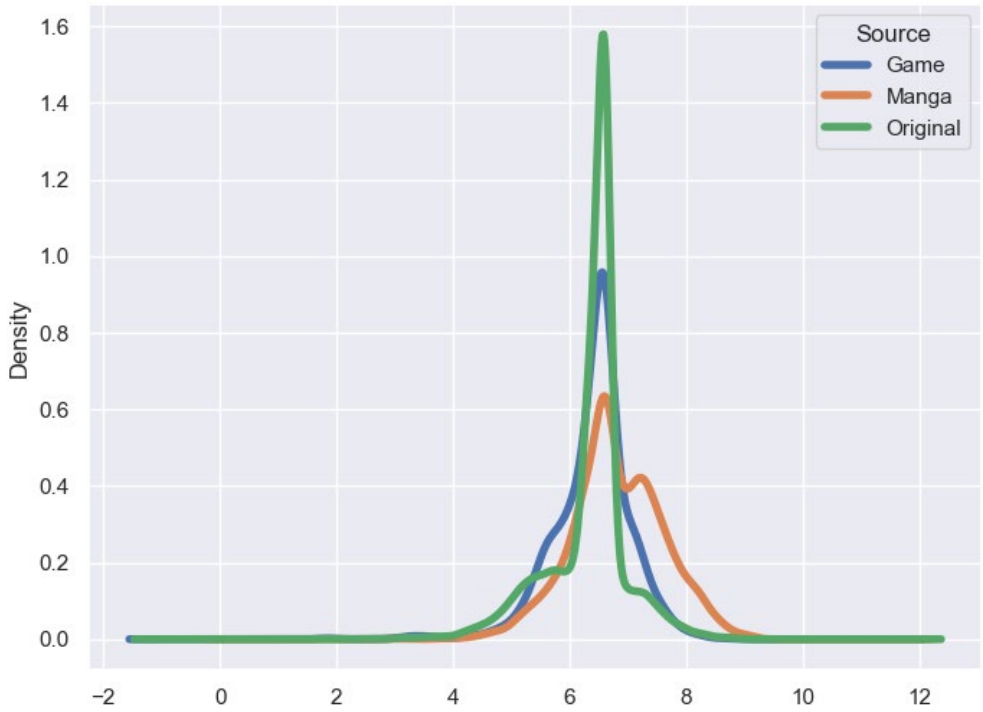
```
1  nsource_top10 = df_copy['Source'].value_counts()
```

```
1  nsource_top10
```

```
Original        8920
Manga           4570
Unknown         3899
Game            1191
Visual novel    1102
Other            964
Light novel      951
Novel            735
Web manga        383
Music            378
4-koma manga     312
Picture book     198
Book             177
Mixed media      125
Web novel        118
Card game         69
Radio             13
Name: Source, dtype: int64
```

9.3.1.2.2.2 Annex 3.1.2.2.2

```
main_sources = df_copy[(df_copy['Source'].isin(['Original', 'Manga',"Game"]))]


# Converting to wide dataframe
data_wide = main_sources.pivot(columns = 'Source',
                               values = "Score")

# plotting multiple density plot
data_wide.plot.kde(figsize = (8, 6),
                   linewidth = 4)
plt.savefig(os.path.join(img_folder, 'density_Source_Score.png'),dpi=600)# Saving the image to the images folder
```



9.3.1.2.2.3 Annex 3.1.2.2.3

```
print("Of all the anime we have in the list,",len(df_copy),",there are a total of",len(main_sources),"that come from an original source, manga or game. This means that",
((len(main_sources) * 100) / len(df_copy)),"% of the anime belong to this group.")
```

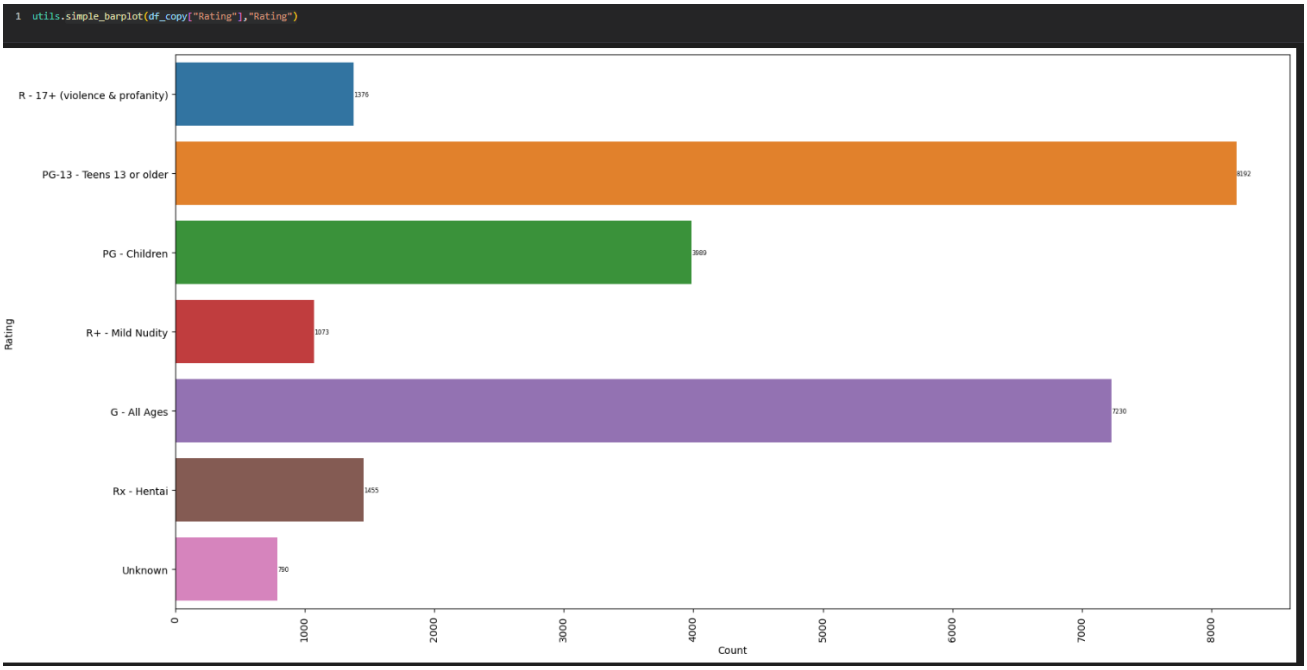
9.3.1.2.3 Annex 3.1.2.3

9.3.1.2.3.1 Annex 3.1.2.3.1

```
1  df_copy["Rating"].value_counts()
```

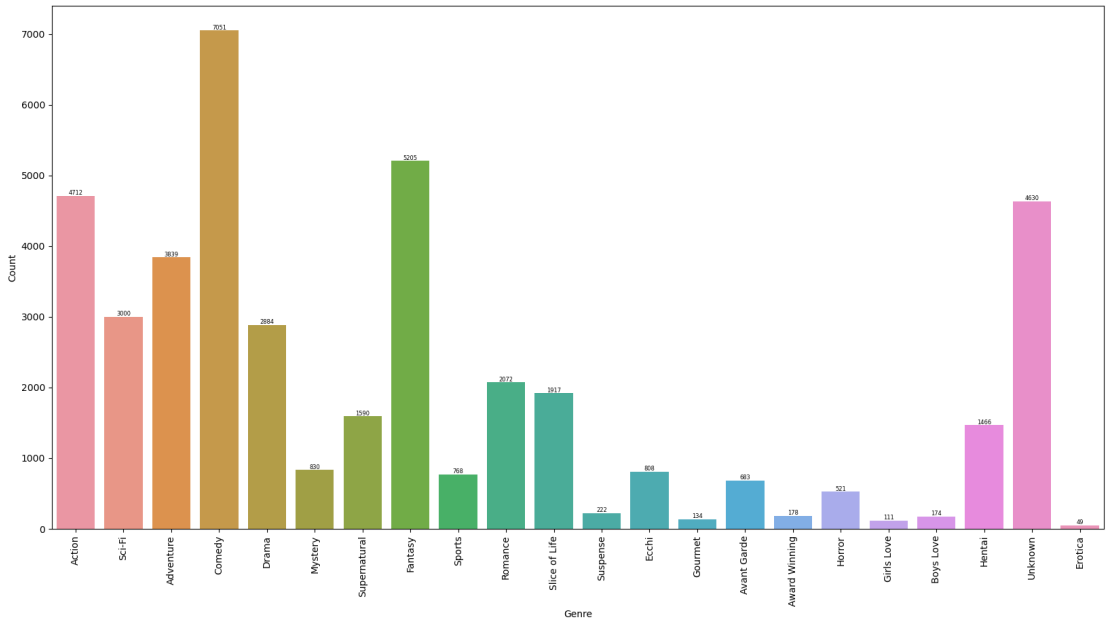
```
PG-13 - Teens 13 or older     8192
G - All Ages                  7230
PG - Children                 3989
Rx - Hentai                   1455
R - 17+ (violence & profanity)1376
R+ - Mild Nudity              1073
Unknown                        790
Name: Rating, dtype: int64
```

9.3.1.2.3.2 Annex 3.1.2.3.2

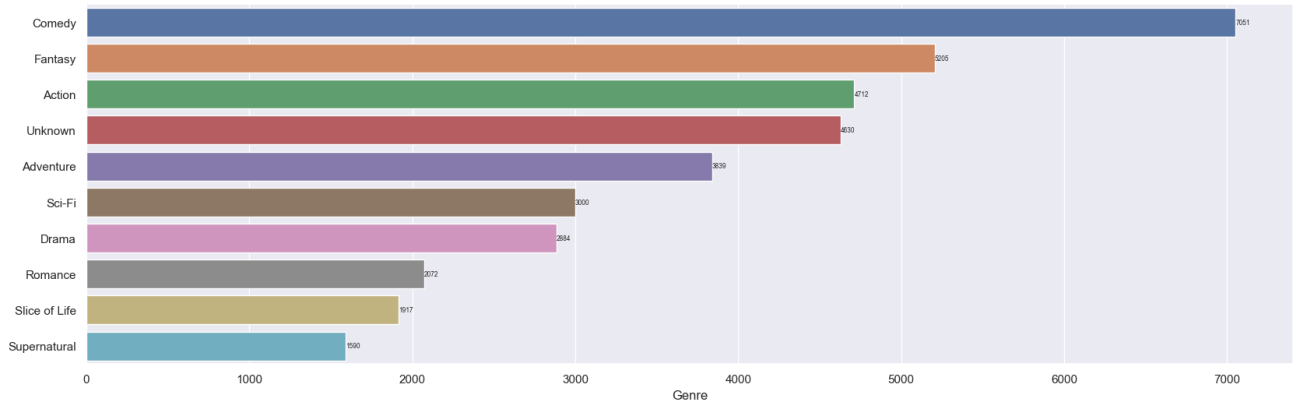


9.3.1.2.4 Annex 3.1.2.4

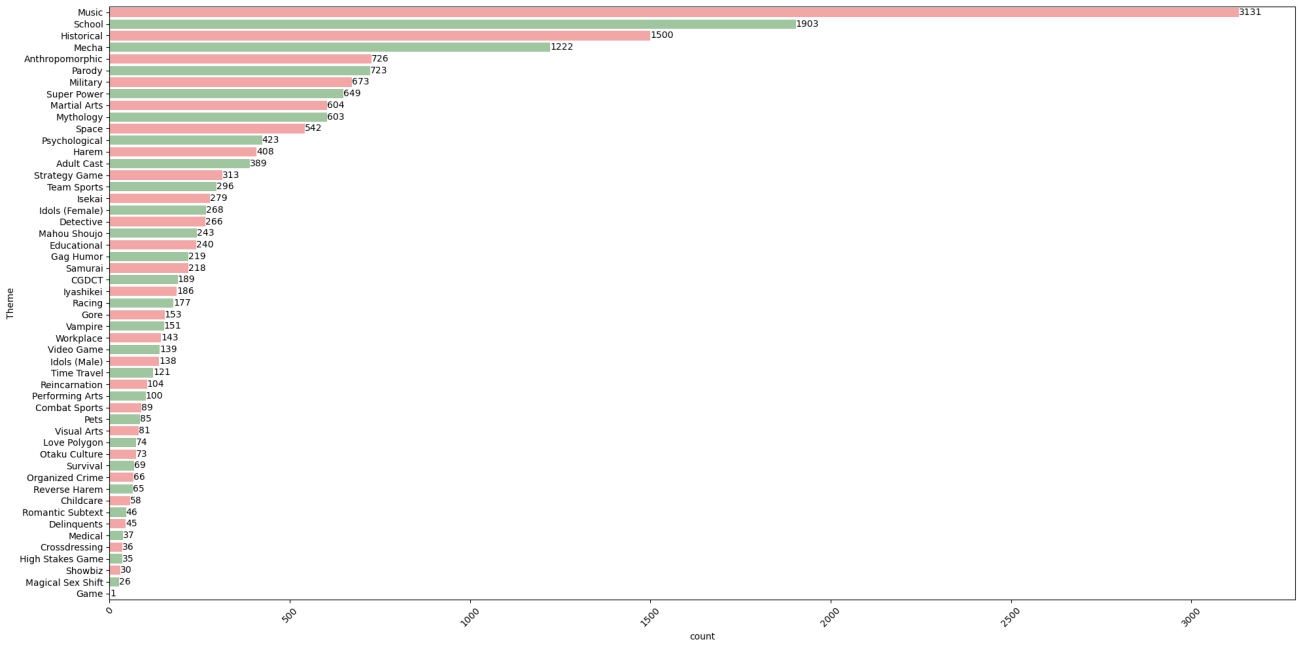
9.3.1.2.4.1 Annex 3.1.2.4.1



9.3.1.2.4.2 Annex 3.1.2.4.2

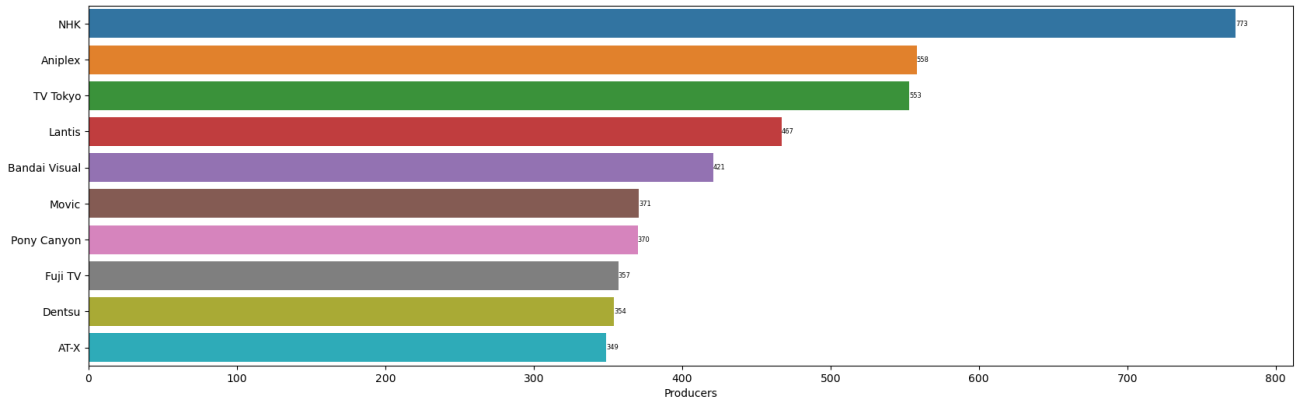


9.3.1.2.5 Annex 3.1.2.5

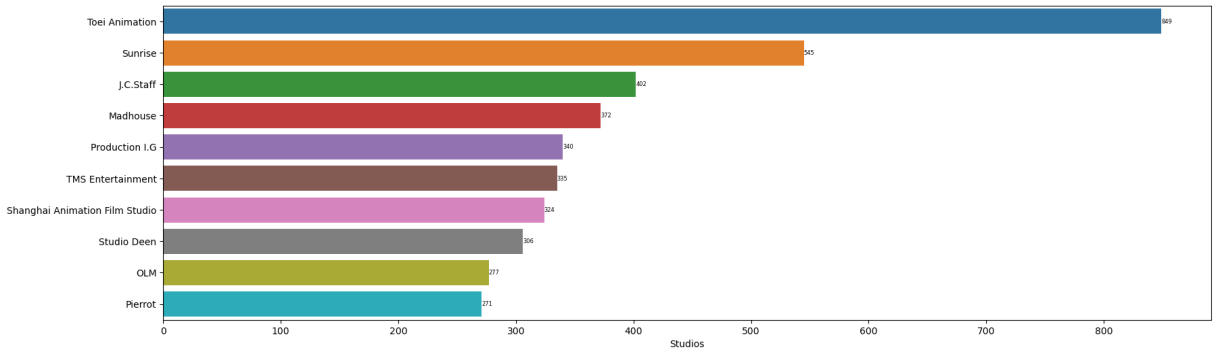


9.3.1.2.6 Annex 3.1.2.6

```
df_copy2 = df_copy.copy()
no_unknown = df_copy2[df_copy2["Producers"] != "Unknown"]
utils.complex_barplot_top10(no_unknown["Producers"],"Producers")
```



9.3.1.2.7 Annex 3.1.2.7



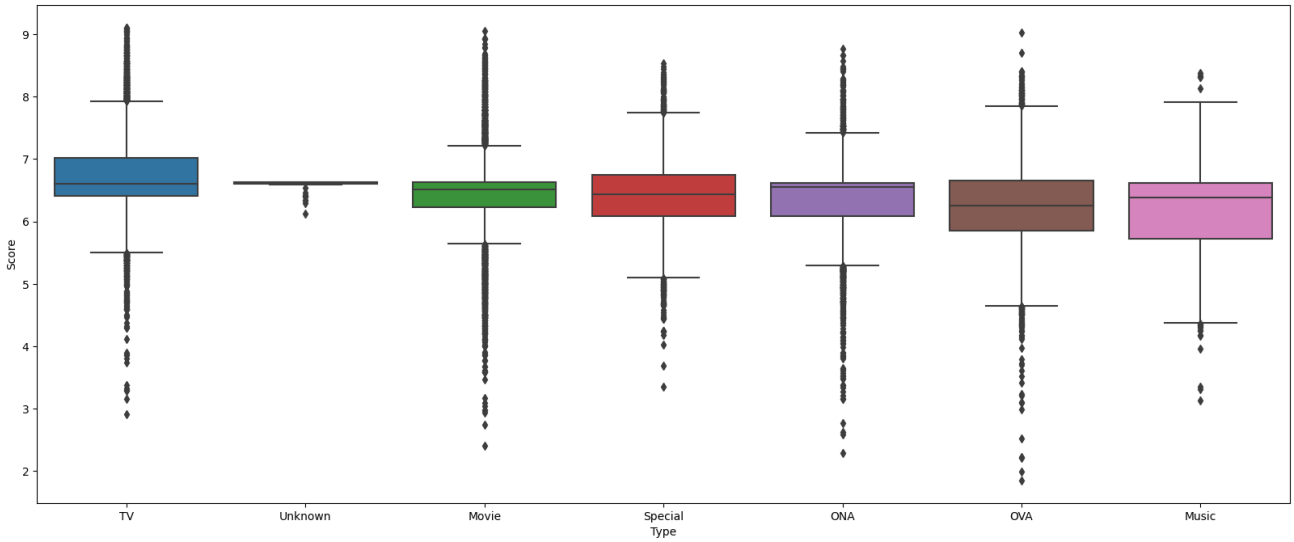
9.3.2 Annex 3.2

```python
# Create a copy of df_copy
df_anime = df_copy.copy()

#Since the infomation are in lists we separate the information to have a better analysis of them.
df_anime['Genre_Split'] = df_anime['Genre'].apply(lambda x : x.split(','))

#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Genre columns with their score
df_copy_aux['Genre'] = pd.Series([x for _list in df_anime['Genre_Split'] for x in _list]) # split the information by comma.
df_copy_aux['Score'] = utils.series_extract(df_anime,'Genre', 'Score') #Funciton to extract the data of a column with respect to another column.
df_copy_aux['Scored_by'] = utils.series_extract(df_anime,'Genre_Split', 'Scored_by') #Funciton to extract the data of a column with respect to another column.
top10 = df_copy_aux[df_copy_aux['Genre'].isin(df_anime['Genre'].str.split(',').explode().value_counts()[0:11].index)] # find the top 10.
```

```python
'''
Function to extract the data of a columns with respect to another column.

'''
def series_extract(df,index_name , target_name): #Col name is the column to which we want to extract the data.

  datos = [] #Store the data
  cont = 0
  index = df.columns.get_loc(target_name) #To know the index of the desired column

  for list_items in df[index_name]:

    for gen in list_items:

        value = df.iloc[cont , index ] #Find the value of the desired column
        datos.append(value) #Store values from the value to the data list

    cont += 1

  return pd.Series(datos)
```

9.3.2.1 Annex 3.2.1

9.3.2.1.1 Annex 3.2.1.1



9.3.2.1.2 Annex 3.2.1.2

```
# finding the mean, median, min and max of Score
type_score = utils.various(df_copy,'Type','Score')
type_score
```

	Score			
	mean	median	min	max
Type				
TV	6.715764	6.598315	2.91	9.11000
Unknown	6.584459	6.618241	6.12	6.62804
Movie	6.429894	6.520000	2.40	9.05000
Special	6.420673	6.434526	3.35	8.54000
ONA	6.328962	6.550301	2.29	8.77000
OVA	6.265786	6.260000	1.85	9.03000
Music	6.182972	6.387958	3.13	8.38000

9.3.2.1.3 Annex 3.2.1.3

```
# finding the mean, median, min and max of Scored_by
type_Scored_by = utils.various(df_copy,'Type','Scored_by')
type_Scored_by
```

	Scored_by			
	mean	median	min	max
Type				
TV	99997.016913	15327.0	7	4125905
ONA	67098.823280	2105.0	0	4128912
Movie	52538.337251	3819.0	0	2113452
OVA	26750.072385	2394.0	1	1650790
Unknown	19307.855263	3274.5	34	279164
Special	19191.662194	2488.0	0	1400265
Music	15936.709147	610.0	8	783563

9.3.2.1.4 Annex 3.2.1.4

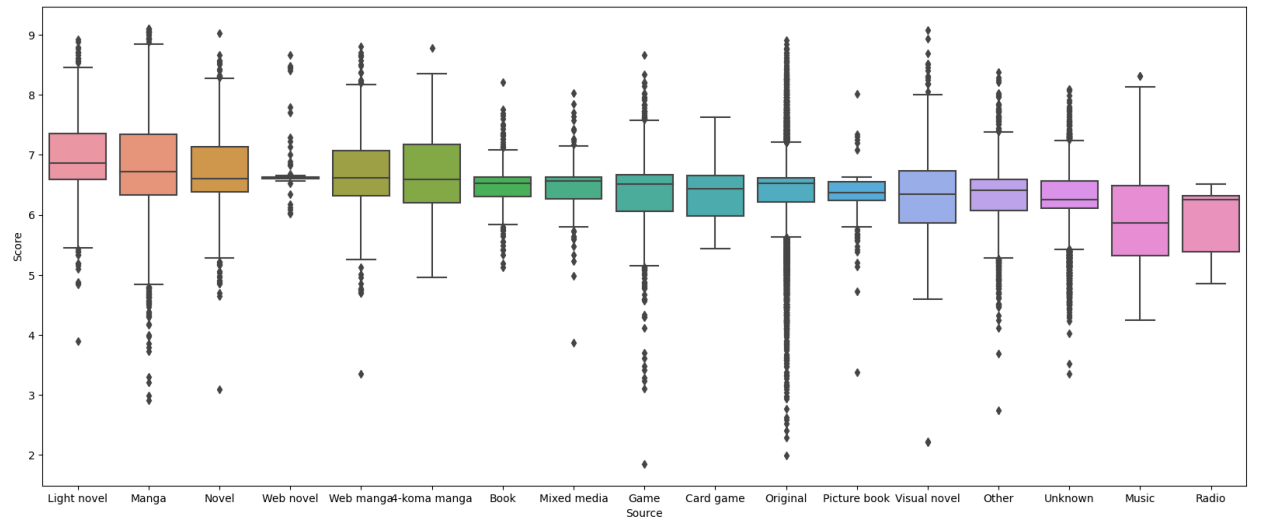
```python
#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Rating columns with their score
df_copy_aux['Type'] =df_copy['Type']
df_copy_aux['Score'] =df_copy['Score']
df_copy_aux['Scored_by'] =df_copy['Scored_by']

# Group a dataframe and calculate coefficient of variation
df_copy_aux.groupby('Type').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

Type	Score	Scored_by
Movie	0.126655	5.913974
OVA	0.117100	9.425140
ONA	0.104425	7.796345
Music	0.104079	5.655682
Special	0.102092	6.817547
TV	0.094912	3.766870
Unknown	0.014019	64.668327

9.3.2.2 Annex 3.2.2

9.3.2.2.1 Annex 3.2.2.1

```python
mean = df_copy.groupby(['Source'] , as_index=False).agg({'Score' : 'mean'}).sort_values('Score' , ascending = False)
utils.box_bidi(df_copy,mean,'Source','Score')
```



9.3.2.2.2 Annex 3.2.2.2

```
# finding the mean, median, min and max of Score
source_score = utils.various(df_copy,'Source','Score')
source_score
```

	Score			
	mean	median	min	max
Source				
Light novel	6.936777	6.870000	3.90	8.92
Manga	6.808442	6.720000	2.91	9.11
Novel	6.724102	6.608612	3.09	9.03
Web novel	6.712726	6.616334	6.02	8.66
Web manga	6.674496	6.612601	3.35	8.81
4-koma manga	6.659434	6.588360	4.96	8.78
Book	6.508041	6.524908	5.13	8.21
Mixed media	6.461490	6.570860	3.87	8.03
Game	6.399492	6.512886	1.85	8.66
Card game	6.349354	6.430000	5.44	7.63

9.3.2.2.3 Annex 3.2.2.3

```
# finding the mean, median, min and max of Scored_by
source_Scored_by = utils.various(df_copy,'Source','Scored_by')
source_Scored_by
```

	Scored_by			
	mean	median	min	max
Source				
Light novel	98362.619348	25598.0	7	2011482
Original	75785.548094	4151.0	3	4128912
Radio	60704.615385	1895.0	105	710474
Unknown	55386.275712	2160.0	1	4125905
Manga	52862.519037	6206.5	0	3371923
Other	52506.056017	3144.0	2	4027113
Web manga	52055.438642	6357.0	26	2047625
Book	50475.067797	5145.0	110	1494226
4-koma manga	32562.711538	7046.0	64	537536
Picture book	30411.641414	4478.5	2	657301

9.3.2.2.4 Annex 3.2.2.4

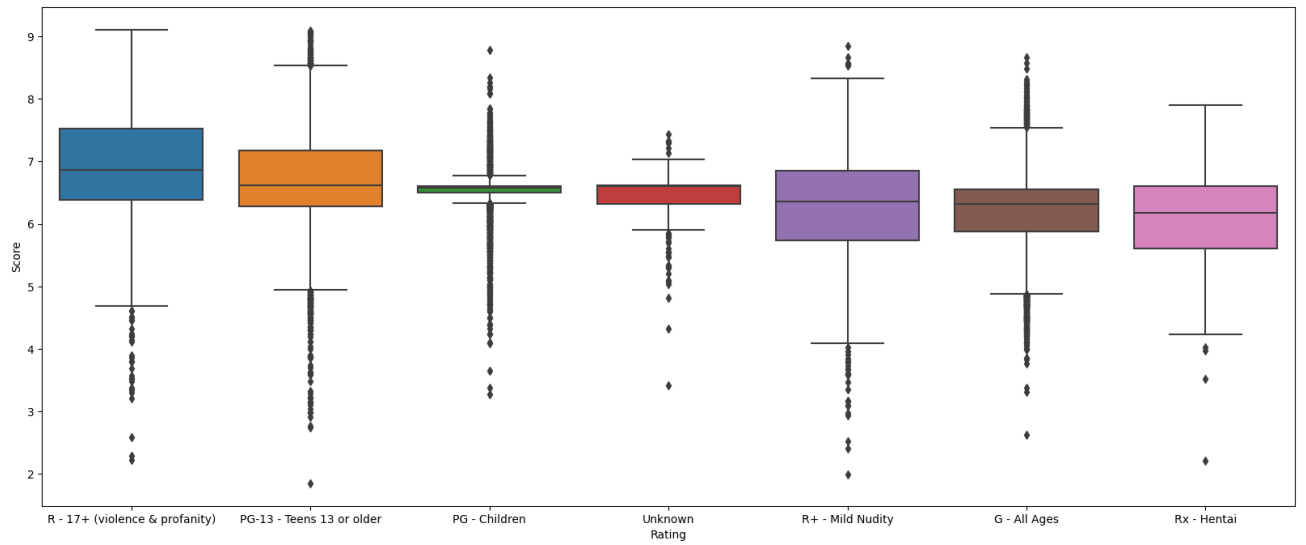
```python
#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Rating columns with their score
df_copy_aux['Source'] =df_copy['Source']
df_copy_aux['Score'] =df_copy['Score']
df_copy_aux['Scored_by'] =df_copy['Scored_by']

# Group a dataframe and calculate coefficient of variation
df_copy_aux.groupby('Source').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

Source	Score	Scored_by
Music	0.129471	5.202485
Manga	0.117149	3.390833
Web manga	0.110569	3.576082
Visual novel	0.108484	3.893348
Original	0.107949	6.513916
4-koma manga	0.105820	2.567235
Novel	0.105060	3.640975
Game	0.101873	6.771364
Other	0.099552	6.629459
Radio	0.098240	3.382923
Light novel	0.097413	1.874981
Mixed media	0.083956	2.505151
Unknown	0.082839	9.466641
Card game	0.079343	3.708277
Book	0.072188	3.914024
Picture book	0.066284	5.743005
Web novel	0.064653	17.249253

9.3.2.3 Annex 3.2.3

9.3.2.3.1 Annex 3.2.3.1

```python
mean = df_copy.groupby(['Rating'] , as_index=False).agg({'Score' : 'mean'}).sort_values('Score' , ascending = False)
utils.box_bidi(df_copy,mean,'Rating','Score')
```



9.3.2.3.2 Annex 3.2.3.2

```
# finding the mean, median, min and max of Score
top_10_Score = utils.various(df_copy,'Rating','Score')
top_10_Score
```

	Score			
	mean	median	min	max
Rating				
R - 17+ (violence & profanity)	6.882970	6.865000	2.22	9.11
PG-13 - Teens 13 or older	6.683179	6.623080	1.85	9.09
PG - Children	6.495474	6.577830	3.27	8.78
Unknown	6.463277	6.599874	3.41	7.44
R+ - Mild Nudity	6.265436	6.360000	1.99	8.85
G - All Ages	6.185382	6.316877	2.63	8.66
Rx - Hentai	6.098950	6.170000	2.21	7.90

9.3.2.3.3 Annex 3.2.3.3

```
# finding the mean, median, min and max of Scored_by
top_10_Scored_by = utils.various(df_copy,'Rating','Scored_by')
top_10_Scored_by
```

	Scored_by			
	mean	median	min	max
Rating				
PG - Children	137004.455001	14269.0	4	4127786
R - 17+ (violence & profanity)	93470.055233	15635.0	3	2554804
G - All Ages	42597.671369	1493.0	1	4128912
R+ - Mild Nudity	42325.785648	5638.0	6	1581878
Unknown	41535.712658	4252.5	2	1556639
PG-13 - Teens 13 or older	40835.374268	4435.0	0	3688044
Rx - Hentai	3350.725086	1820.0	113	191517

9.3.2.3.4 Annex 3.2.3.4

```
#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Rating columns with their score
df_copy_aux['Rating'] =df_copy['Rating']
df_copy_aux['Score'] =df_copy['Score']
df_copy_aux['Scored_by'] =df_copy['Scored_by']

# Group a dataframe and calculate coefficient of variation
df_copy_aux.groupby('Rating').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

	Score	Scored_by
Rating		
R+ - Mild Nudity	0.148224	4.343502
R - 17+ (violence & profanity)	0.136874	2.577308
PG-13 - Teens 13 or older	0.113721	3.584067
Rx - Hentai	0.113070	2.700367
G - All Ages	0.100157	8.815201
PG - Children	0.059216	5.073127
Unknown	0.045300	28.582256

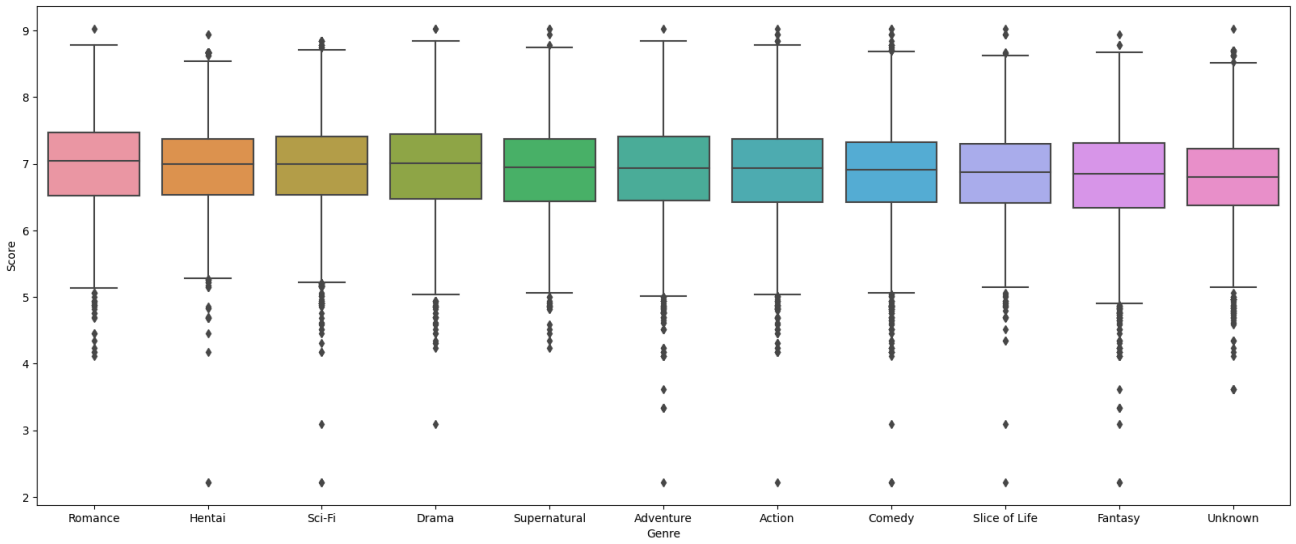
9.3.2.4 Annex 3.2.4

9.3.2.4.1 Annex 3.2.4.1

```python
# Create a copy of df_copy
df_anime = df_copy.copy()

#Since the infomation are in lists we separate the information to have a better analysis of them.
df_anime['Genre_Split'] = df_anime['Genre'].apply(lambda x : x.split(','))

#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Genre columns with their score
df_copy_aux['Genre'] = pd.Series([x for _list in df_anime['Genre_Split'] for x in _list]) # split the information by comma.
df_copy_aux['Score'] = utils.series_extract(df_anime,'Genre', 'Score') #Funciton to extract the data of a column with respect to another column.
df_copy_aux['Scored_by'] = utils.series_extract(df_anime,'Genre_Split', 'Scored_by') #Funciton to extract the data of a column with respect to another column.
top10 = df_copy_aux[df_copy_aux['Genre'].isin(df_anime['Genre'].str.split(',').explode().value_counts()[0:11].index)] # find the top 10
```

9.3.2.4.2 Annex 3.2.4.2

```python
mean = top10.groupby(['Genre'] , as_index=False).agg({'Score' : 'mean'}).sort_values('Score' , ascending = False)
utils.box_bidi(df_copy_aux,mean,'Genre','Score')
```



9.3.2.4.3 Annex 3.2.4.3

```python
# finding the mean, median, min and max of Score
top_10_Score = utils.various(top10,'Genre','Score')
top_10_Score
```

	Score			
	mean	median	min	max
Genre				
Romance	6.981447	7.04	4.11	9.03
Hentai	6.959182	7.00	2.22	8.94
Sci-Fi	6.954737	7.00	2.22	8.85
Drama	6.953298	7.01	3.09	9.03
Supernatural	6.900145	6.95	4.24	9.03
Adventure	6.896871	6.94	2.22	9.03
Action	6.894336	6.94	2.22	9.03
Comedy	6.868456	6.91	2.22	9.03
Slice of Life	6.840565	6.88	2.22	9.03
Fantasy	6.792027	6.85	2.22	8.94

9.3.2.4.4 Annex 3.2.4.4

```python
# finding the mean, median, min and max of Scored_by
top_10_Scored_by = utils.various(top10,'Genre','Scored_by')
top_10_Scored_by
```

	Scored_by			
	mean	median	min	max
Genre				
Unknown	87739.613823	3960.5	1	4127786
Fantasy	72304.437848	6514.0	0	4125905
Supernatural	62715.076101	7945.5	0	3688044
Romance	58279.396236	9289.0	6	2011482
Drama	56606.485784	6223.0	6	2554804
Adventure	55636.747591	4573.0	2	3298982
Action	54443.315365	4823.0	0	2554804
Comedy	49121.644022	4838.0	0	4128912
Sci-Fi	38875.956000	4085.0	17	1866134
Slice of Life	32962.775170	4449.0	2	1311695

9.3.2.4.5 Annex 3.2.4.5

```python
# Group a dataframe and calculate coefficient of variation
top10.groupby('Genre').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

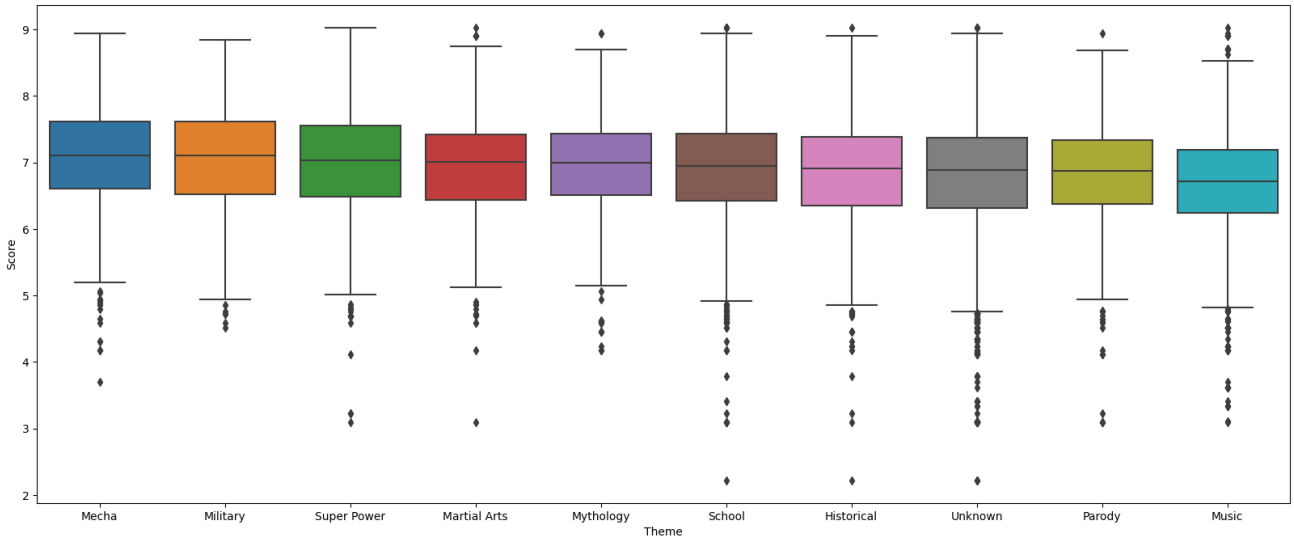
	Score	Scored_by
Genre		
Fantasy	0.116002	4.381558
Adventure	0.110825	4.316449
Action	0.109082	3.473258
Drama	0.108373	3.927941
Supernatural	0.108079	3.155532
Comedy	0.107416	5.747615
Romance	0.106876	2.381233
Sci-Fi	0.106729	3.936526
Hentai	0.106257	2.777146
Unknown	0.106112	6.404213
Slice of Life	0.105109	4.928605

9.3.2.5 Annex 3.2.5

9.3.2.5.1 Annex 3.2.5.1

```python
# Create a copy of df_copy
df_anime = df_copy.copy()

#Since the infomation are in lists we separate the information to have a better analysis of them.
df_anime['Theme_Split'] = df_anime['Theme'].apply(lambda x : x.split(','))

#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Theme columns with their score
df_copy_aux['Theme'] = pd.Series([x for _list in df_anime['Theme_Split'] for x in _list]) # split the information by comma.
df_copy_aux['Score'] = utils.series_extract(df_anime,'Theme', 'Score') #Funciton to extract the data of a column with respect to another column.
df_copy_aux['Scored_by'] = utils.series_extract(df_anime,'Theme_Split', 'Scored_by') #Funciton to extract the data of a column with respect to another column.
top10 = df_copy_aux[df_copy_aux['Theme'].isin(df_anime['Theme'].str.split(',').explode().value_counts()[0:11].index)] # find the top 10
```

9.3.2.5.2 Annex 3.2.5.2



9.3.2.5.3 Annex 3.2.5.3

```
1  top_10_Score = utils.various(top10,'Theme','Score') #
2  top_10_Score
```

	Score			
	mean	median	min	max
Theme				
Mecha	7.074089	7.110	3.70	8.94
Military	7.037845	7.100	4.51	8.85
Super Power	6.989333	7.030	3.09	9.03
Martial Arts	6.930396	7.005	3.09	9.03
Mythology	6.928585	7.000	4.17	8.94
School	6.904887	6.950	2.22	9.03
Historical	6.860463	6.910	2.22	9.03
Unknown	6.828389	6.890	2.22	9.03
Parody	6.820069	6.870	3.09	8.94
Music	6.687040	6.720	3.09	9.03

9.3.2.5.4 Annex 3.2.5.4

```
1  top_10_Scored_by = utils.various(top10,'Theme','Scored_by')
2  top_10_Scored_by
```

	Scored_by			
	mean	median	min	max
Theme				
Anthropomorphic	205861.563361	31196.0	44	4118873
School	71842.141356	13145.0	0	1898783
Super Power	67173.983051	10018.0	41	2047625
Unknown	63542.958408	3819.5	0	4128912
Mythology	60195.835821	4384.0	33	2113452
Military	49539.936107	5736.0	101	2554804
Historical	45073.244667	3576.0	33	1827864
Parody	38766.771784	3470.0	12	2047625
Mecha	37236.290507	2965.5	17	2111993
Martial Arts	35203.192053	2609.5	77	1811391

9.3.2.5.5 Annex 3.2.5.5

```
# Group a dataframe and calculate coefficient of variation
top10.groupby('Theme').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

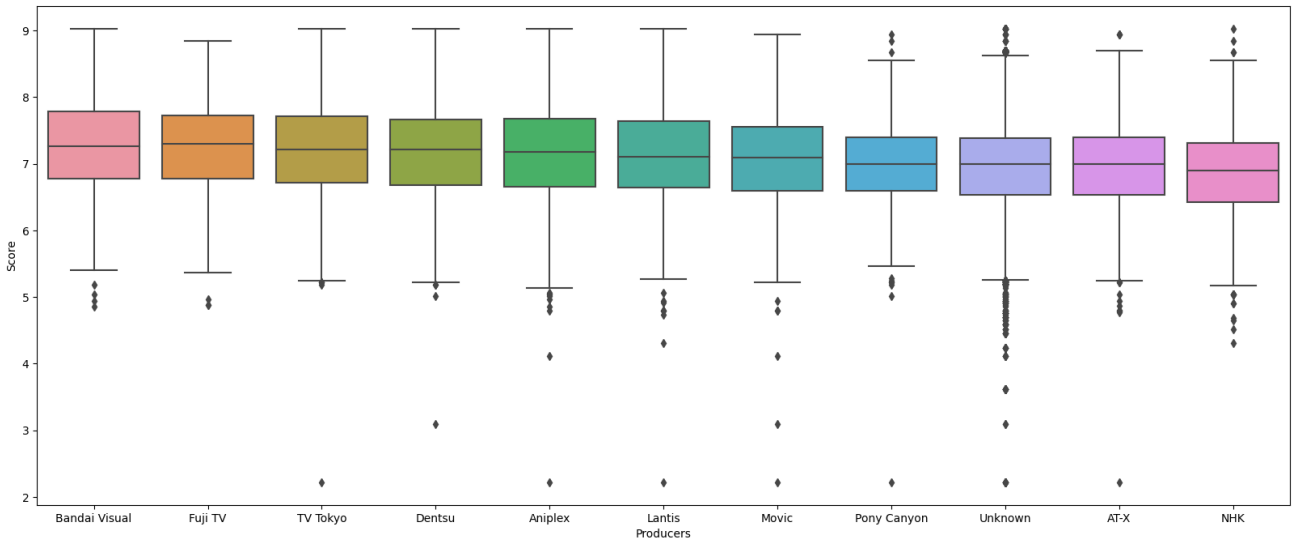
Theme	Score	Scored_by
Anthropomorphic	0.133040	3.772931
Parody	0.122309	4.069218
Super Power	0.121807	3.179654
Unknown	0.120079	7.261295
Historical	0.119683	4.496769
School	0.118293	2.408797
Martial Arts	0.118215	5.791825
Military	0.116598	4.001174
Music	0.116153	5.351533
Mythology	0.114823	3.371834
Mecha	0.112494	4.477915

9.3.2.6 Annex 3.2.6

9.3.2.6.1 Annex 3.2.6.1

```
# Create a copy of df_copy
df_anime = df_copy.copy()

#Since the infomation are in lists we separate the information to have a better analysis of them.
df_anime['Producers_Split'] = df_anime['Producers'].apply(lambda x : x.split(','))

#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the producers columns with their score
df_copy_aux['Producers'] = pd.Series([x for _list in df_anime['Producers_Split'] for x in _list]) # split the information by comma.
df_copy_aux['Score'] = utils.series_extract(df_anime,'Producers', 'Score') #Funciton to extract the data of a column with respect to another column.
df_copy_aux['Scored_by'] = utils.series_extract(df_anime,'Producers_Split', 'Scored_by') #Funciton to extract the data of a column with respect to another column.
top10 = df_copy_aux[df_copy_aux['Producers'].isin(df_anime['Producers'].str.split(',').explode().value_counts()[0:11].index)] # find the top 10

```

9.3.2.6.2 Annex 3.2.6.2



9.3.2.6.3 Annex 3.2.6.3

```
# finding the mean, median, min and max of Score
top_10_Score = utils.various(top10,'Producers','Score')
top_10_Score
```

	Score			
	mean	median	min	max
Producers				
Bandai Visual	7.284394	7.260	4.85	9.03
Fuji TV	7.251485	7.300	4.88	8.85
TV Tokyo	7.201465	7.220	2.22	9.03
Dentsu	7.178023	7.220	3.09	9.03
Aniplex	7.151595	7.175	2.22	9.03
Lantis	7.120236	7.110	2.22	9.03
Movic	7.081294	7.090	2.22	8.94
Pony Canyon	6.974838	6.990	2.22	8.94
Unknown	6.947467	7.000	2.22	9.03
AT-X	6.941461	6.990	2.22	8.94

9.3.2.6.4 Annex 3.2.6.4

```
# finding the mean, median, min and max of Scored_by
top_10_Scored_by = utils.various(top10,'Producers','Scored_by')
top_10_Scored_by
```

	Scored_by			
	mean	median	min	max
Producers				
Dentsu	192169.759887	46539.5	114	2554804
Movic	160091.566038	49017.0	96	1898783
Aniplex	153482.274194	49176.0	117	2011482
AT-X	132934.008596	68613.0	96	1442254
Pony Canyon	99686.697297	23238.5	104	2554804
Fuji TV	87661.128852	15245.0	131	1250813
Lantis	82159.199143	24281.0	127	2047625
TV Tokyo	72853.336347	12820.0	121	2047625
Unknown	67894.512406	3723.0	0	4128912
Bandai Visual	43384.330166	9616.0	120	2047625

9.3.2.6.5 Annex 3.2.6.5

```
# Group a dataframe and calculate coefficient of variation
top10.groupby('Producers').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

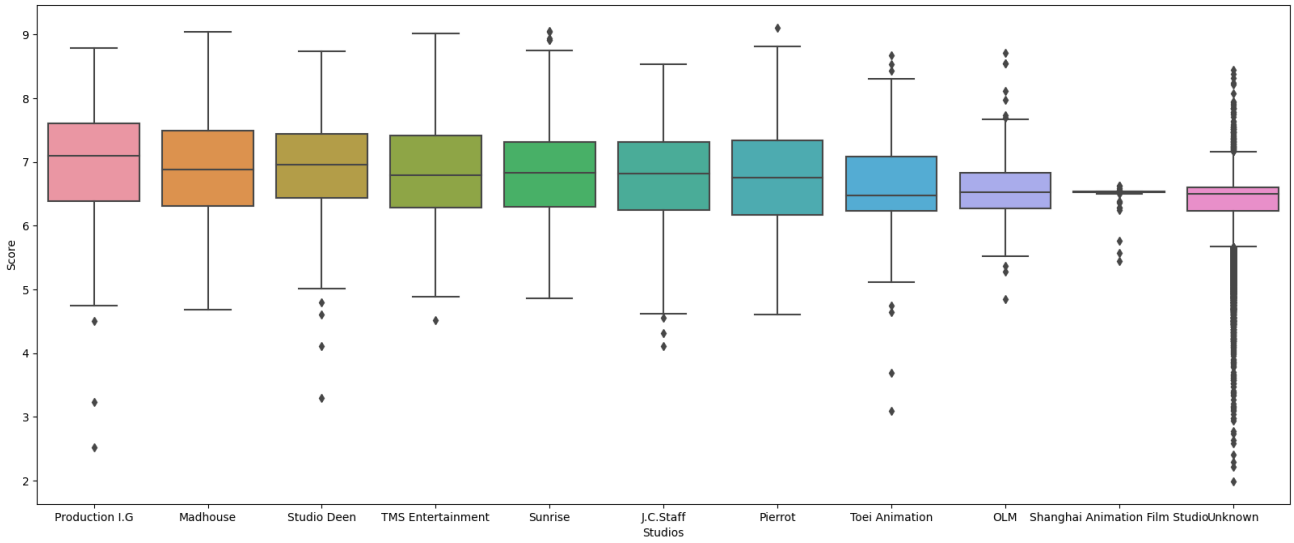
Producers	Score	Scored_by
Aniplex	0.114587	1.780259
Lantis	0.113323	2.135781
Movic	0.112961	1.713928
AT-X	0.111794	1.448191
Unknown	0.109032	7.139484
Dentsu	0.108199	1.804679
NHK	0.106444	6.234538
TV Tokyo	0.106250	2.722853
Pony Canyon	0.101995	2.675487
Bandai Visual	0.099583	3.504355
Fuji TV	0.095369	2.409855

9.3.2.7 Annex 3.2.7

9.3.2.7.1 Annex 3.2.7.1

```
# Create a copy of df_copy
df_anime = df_copy.copy()

#Since the infomation are in lists we separate the information to have a better analysis of them.
df_anime['Studios_Split'] = df_anime['Studios'].apply(lambda x : x.split(','))

#create a new auxiliar dataframe
df_copy_aux = pd.DataFrame()

#Set the Studios columns with their score
df_copy_aux['Studios'] = pd.Series([x for _list in df_anime['Studios_Split'] for x in _list]) # split the information by comma.
df_copy_aux['Score'] = utils.series_extract(df_anime,'Studios_Split', 'Score') #Funciton to extract the data of a column with respect to another column.
df_copy_aux['Scored_by'] = utils.series_extract(df_anime,'Studios_Split', 'Scored_by') #Funciton to extract the data of a column with respect to another column.
top10 = df_copy_aux[df_copy_aux['Studios'].isin(df_anime['Studios'].str.split(',').explode().value_counts()[0:11].index)] # find the top 10
```

9.3.2.7.2 Annex 3.2.7.2



9.3.2.7.3 Annex 3.2.7.3

```
# finding the mean, median, min and max of Score
top_10_Score = utils.various(top10,'Studios','Score')
top_10_Score
```

	Score			
	mean	median	min	max
Studios				
Production I.G	6.980485	7.100000	2.52	8.79000
Madhouse	6.917399	6.880000	4.68	9.04000
Studio Deen	6.903755	6.955000	3.30	8.74000
TMS Entertainment	6.869647	6.790000	4.52	9.02000
Sunrise	6.832438	6.830000	4.86	9.05000
J.C.Staff	6.756772	6.820000	4.11	8.54000
Pierrot	6.752000	6.760000	4.60	9.11000
Toei Animation	6.587783	6.480000	3.09	8.67000
OLM	6.578704	6.520000	4.85	8.71000
Shanghai Animation Film Studio	6.514768	6.526434	5.45	6.62701

9.3.2.7.4 Annex 3.2.7.4

```
# finding the mean, median, min and max of Scored_by
top_10_Scored_by = utils.various(top10,'Studios','Scored_by')
top_10_Scored_by
```

	Scored_by			
	mean	median	min	max
Studios				
Shanghai Animation Film Studio	155512.469136	35262.5	122	861144
Unknown	77902.834289	3326.0	0	4128912
Madhouse	73506.147849	6864.5	122	2520521
Pierrot	68742.638376	4885.0	112	1811391
J.C.Staff	62649.457711	10147.5	38	1288419
Production I.G	49655.214706	10398.5	113	1116553
Studio Deen	43860.411765	8708.0	112	1162643
TMS Entertainment	33958.364179	4605.0	86	928782
Sunrise	24485.658716	2825.0	115	1298956
Toei Animation	23860.241461	2334.0	30	1392685

9.3.2.7.5 Annex 3.2.7.5

```
# Group a dataframe and calculate coefficient of variation
top10.groupby('Studios').apply(lambda x: np.std(x, ddof=1) / np.mean(x)).sort_values('Score' , ascending = False)
```

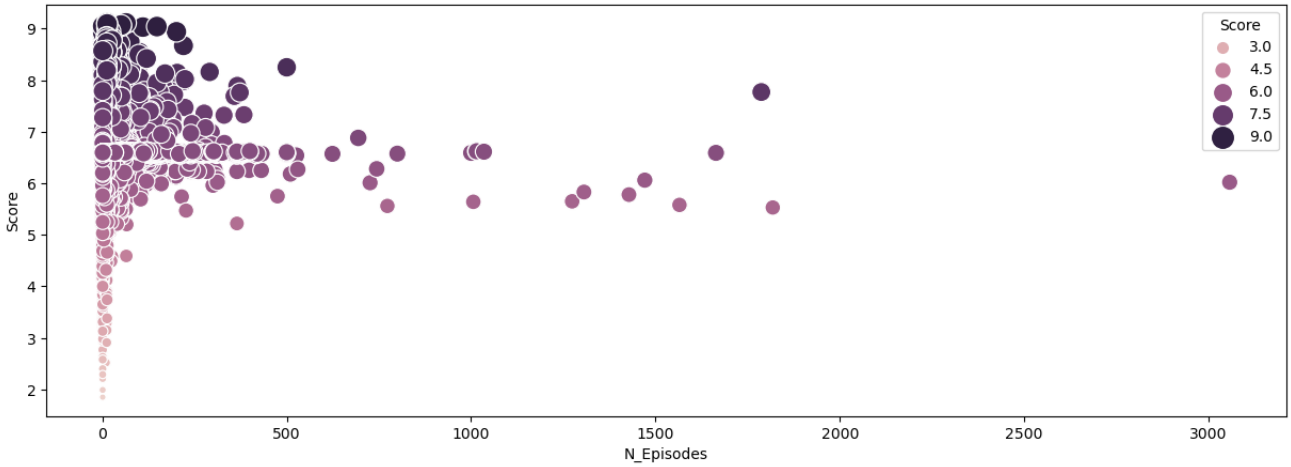
	Score	Scored_by
Studios		
Production I.G	0.124596	2.583855
Madhouse	0.121096	3.180562
J.C.Staff	0.118023	2.320688
Pierrot	0.117720	3.198851
Studio Deen	0.107735	2.574662
TMS Entertainment	0.107275	2.665198
Sunrise	0.105539	3.919097
Toei Animation	0.101398	3.927580
Unknown	0.094762	4.125380
OLM	0.079682	2.278482
Shanghai Animation Film Studio	0.014618	1.425880

9.4 Annex 4

9.4.1 Annex 4.1

9.4.1.1 Annex 4.1.1

```
utils.scat(df_copy,"N_Episodes","Score","N_Episodes_VS_Score")
```

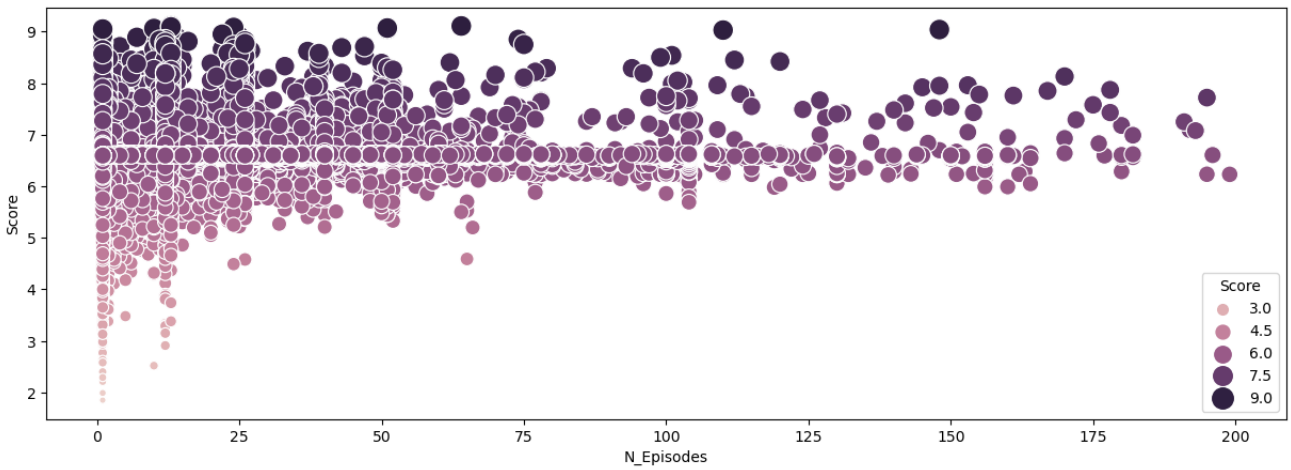


9.4.1.2 Annex 4.1.2

```
# Get count of values greater than 2022 in the column 'Released'
count = df_copy["N_Episodes"][df_copy["N_Episodes"] > 200].count()
print("There are",count,"animes with more than 200 episodes")
```

9.4.1.3 Annex 4.1.3

```
until2000 = df_copy[df_copy["N_Episodes"] < 200] #there are a couple of outliers, it is better to do not take them into account
utils.scat(until2000,"N_Episodes","Score","N_Episodes_VS_Score_Less200")
```



9.4.1.4 Annex 4.1.4

```
# Spearman's Rank Correlation Test
# Revisamos la correlacion
df_copy['N_Episodes'].corr(df_copy['Score'] , method = 'spearman')
```

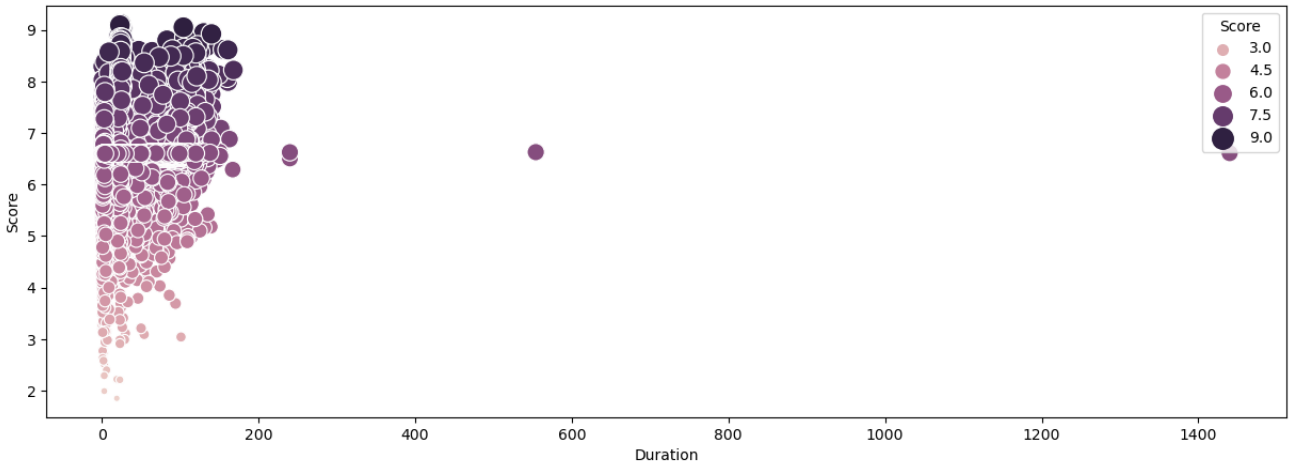
9.4.1.5 Annex 4.1.5

```
# Pearson's dependency test
from scipy.stats import pearsonr
data1 = df_copy['N_Episodes']
data2 = df_copy['Score']
stat, p = pearsonr(data1, data2)
print('stat=%.3f, p=%.3f' % (stat, p))
```

9.4.2 Annex 4.2

9.4.2.1 Annex 4.1.1

```
utils.scat(df_copy,"Duration","Score","Duration_VS_Score")
```

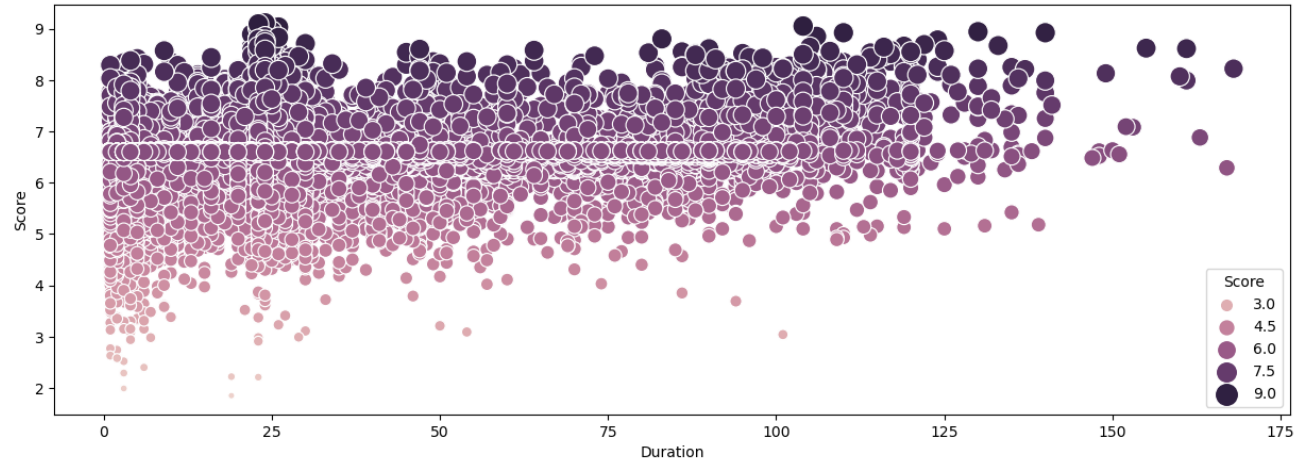


9.4.2.2 Annex 4.1.2

```
# Get count of values greater than 2022 in the column 'Released'
count = df_copy["Duration"][df_copy["Duration"] > 200].count()
print("There are",count,"animes with a duration of more than 200 minutes")
```

9.4.2.3 Annex 4.1.3

```
until400 = df_copy[df_copy["Duration"] < 200] #there are a couple of outliers, it is better to do not taking them into account
utils.scat(until400,"Duration","Score","Duration_VS_Score_less 200")
```



9.4.2.4 Annex 4.1.4

```
# Spearman's Rank Correlation Test
# Revisamos la correlacion
df_copy['Duration'].corr(df_copy['Score'], method = 'spearman')
```

9.4.2.5 Annex 4.1.5

```
# Spearman' dependency test
from scipy.stats import spearmanr
data1 = df_copy['Score']
data2 = df_copy['Duration']
stat, p = spearmanr(data1, data2)
print('stat=%.33f, p=%.3f' % (stat, p))
```

Thank you for your time and attention!

Please do not hesitate to contact us if you have any questions about information highlighted in this document.

END OF DOCUMENT