```
# the imports
# seaborn
import seaborn as sns
# matplotlib
import matplotlib.pyplot as plt
# pandas
import pandas as pd
# numpy
import numpy as np
# regular expression for the score stuff
import re


# the data
chapterData = pd.read_csv("chapterData_2.csv")
mangaData = pd.read_csv("mangaData_2.csv")

# get the number of rows
print(f"mangaData: {mangaData.shape[0]} || chapterData: {chapterData.shape[0]}")
```

```
mangaData: 1784 || chapterData: 310
```

unlogical difference in rows

script has been stopped in the middle of execution so it might be due to that

```
# basic plots to explore the data
# histograme to see the distribution of the views

# sns.distplot(chapterData["views"], kde=False)
```

cant plot since views are strings in form of 1.2M and 1.3K

convert the views to numbers using numpy

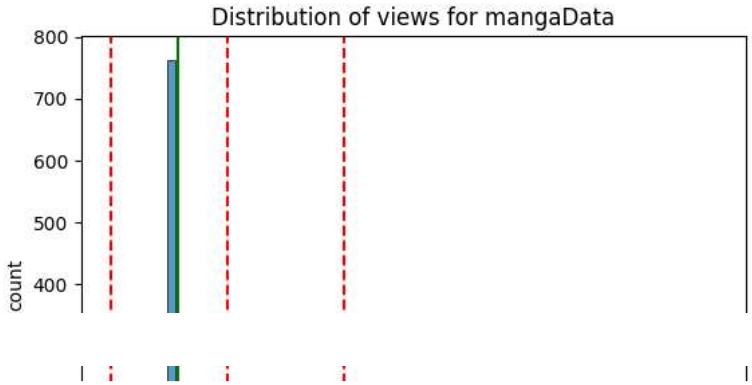
```
chapterData["views"] = chapterData["views"].apply(lambda x: float(x[:-1])*1000 if x[-1] == "K" else float(x[:-1])*1000000)
mangaData["views"] = mangaData["views"].apply(lambda x: float(x[:-1])*1000 if x[-1] == "K" else float(x[:-1])*1000000)

# see the type of the views
print(chapterData["views"].dtype)
```

```
float64
```

great now its in float64 so we can go ahead and plot it

```
sns.histplot(mangaData["views"], kde=False)
plt.axvline(mangaData["views"].mean(), color='r', linestyle='--')
plt.axvline(mangaData["views"].median(), color='g', linestyle='-')
# add labels for the lines
plt.text(mangaData["views"].mean()+1000000, 100, "mean", rotation=90)
plt.text(mangaData["views"].median()+1000000, 100, "median", rotation=90)
plt.xlabel("views (in millions)")
plt.ylabel("count")
plt.title("Distribution of views for mangaData")
# draw the standerd deviation
plt.axvline(mangaData["views"].mean()+mangaData["views"].std(), color='r', linestyle='--')
plt.axvline(mangaData["views"].mean()-mangaData["views"].std(), color='r', linestyle='--')
# add labels for the lines
plt.text(mangaData["views"].mean()+mangaData["views"].std()+1000000, 100, "std", rotation=90)
```

```
Text(17719799.709582537, 100, 'std')
```



Makes sense

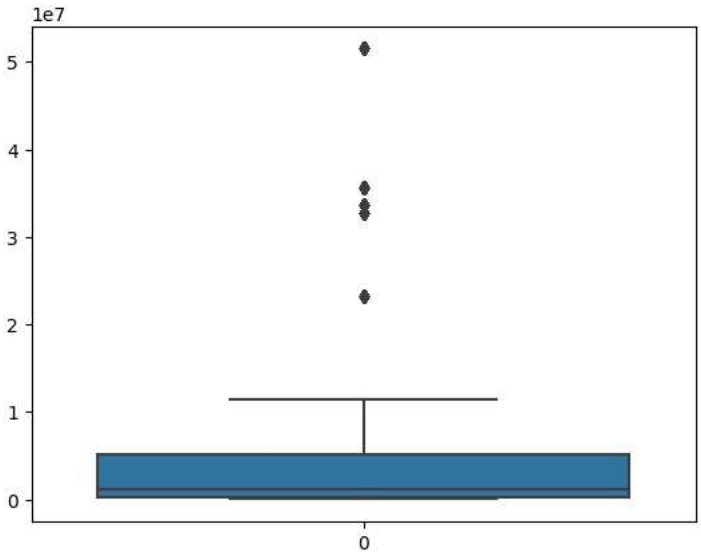
there are more mangas with few views

the distribution is right skewed

lets see the box plot

```
sns.boxplot(mangaData["views"])
```

```
<Axes: >
```



We must have a lot of outliers or its just

lets see how those outliers are and if it makes sense for them to have that many views

```
# find the outliers and print their lines
# get the quantiles
q1 = mangaData["views"].quantile(0.25)
q3 = mangaData["views"].quantile(0.75)
# get the interquartile range
iqr = q3 - q1
# get the lower and upper bounds
lower_bound = q1 - (1.5 * iqr)
upper_bound = q3 + (1.5 * iqr)
# get the outliers
outliers = mangaData[(mangaData["views"] < lower_bound) | (mangaData["views"] > upper_bound)]
# print the outliers
print(f"name: {outliers['name']}\nviews: {outliers['views']}\n")
```

```
name: 11                              Library of Heaven's Path
33      I was Stuck on the Same Day for One Hundred Th...
34      I was Stuck on the Same Day for One Hundred Th...
37                               Infinite Leveling: Murim
45                                            Existence
                            ...
```

```
1747                            Library of Heaven's Path
1769    I was Stuck on the Same Day for One Hundred Th...
1770    I was Stuck on the Same Day for One Hundred Th...
1773                             Infinite Leveling: Murim
1781                                           Existence
Name: name, Length: 191, dtype: object
views: 11      33700000.0
33      35600000.0
34      35600000.0
37      51500000.0
45      23300000.0
           ...
1747    33700000.0
1769    35600000.0
1770    35600000.0
1773    51500000.0
1781    23300000.0
Name: views, Length: 191, dtype: float64
```

```
len(outliers)
print(f"lower bound: {lower_bound} || upper bound: {upper_bound}")
```

```
lower bound: -7043875.0 || upper bound: 12386325.0
```

We got 191 outliers

out of 1784 which is 0.1 %

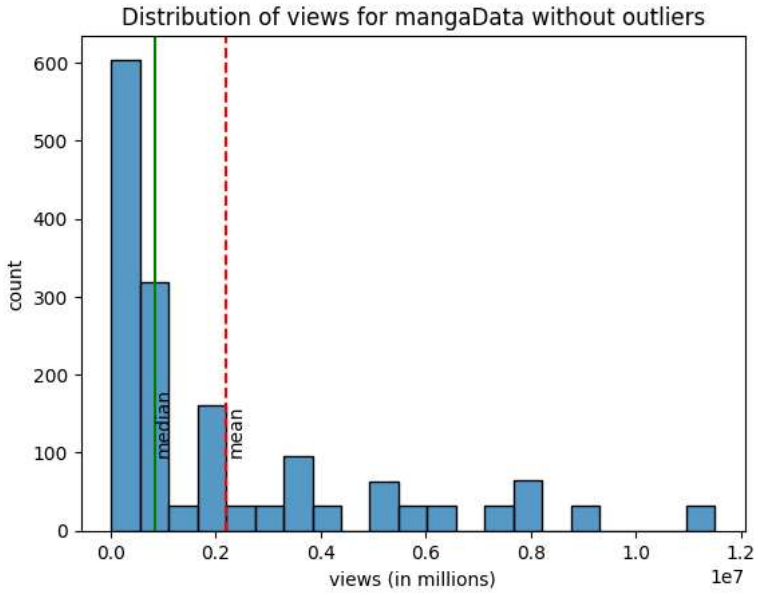
its less than 5% so we can just delete them

but we are intersted in the most popular ones so we might be able to delete the ones under the lower bound

```
# distribution without the outliers
no_outliers = mangaData[(mangaData["views"] > lower_bound) & (mangaData["views"] < upper_bound)]
```

```
sns.histplot(no_outliers["views"], kde=False)
# plto with it the mean and median as vertical lines
plt.axvline(no_outliers["views"].mean(), color='r', linestyle='--')
plt.axvline(no_outliers["views"].median(), color='g', linestyle='-')
# add labels for the lines
plt.text(no_outliers["views"].mean(), 100, "mean", rotation=90)
plt.text(no_outliers["views"].median(), 100, "median", rotation=90)
# labels for views with the unit
plt.xlabel("views (in millions)")
plt.ylabel("count")
plt.title("Distribution of views for mangaData without outliers")
```

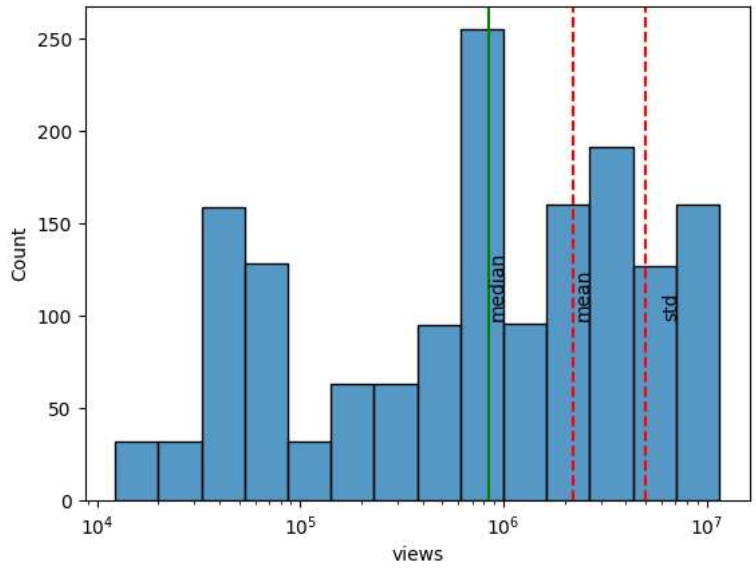
```
Text(0.5, 1.0, 'Distribution of views for mangaData without outliers')
```



right skewed lets see if we are going to get a normal dist if we transform it

```
sns.histplot(no_outliers["views"], kde=False, log_scale=True)
plt.axvline(no_outliers["views"].mean(), color='r', linestyle='--')
plt.axvline(no_outliers["views"].median(), color='g', linestyle='-')
# add labels for the lines
plt.text(no_outliers["views"].mean(), 100, "mean", rotation=90)
plt.text(no_outliers["views"].median(), 100, "median", rotation=90)
# draw the standerd deviation
plt.axvline(no_outliers["views"].mean()+no_outliers["views"].std(), color='r', linestyle='--')
plt.axvline(no_outliers["views"].mean()-no_outliers["views"].std(), color='r', linestyle='--')
# add labels for the lines
plt.text(no_outliers["views"].mean()+no_outliers["views"].std()+1000000, 100, "std", rotation=90)
```

```
Text(6001123.781321166, 100, 'std')
```

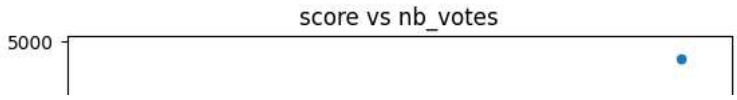


Lets see their ratings

```
# making the score and nb_votes numbers
mangaData["score"] = mangaData["score"].apply(lambda x: float(
    re.search(r"\d+\.\d+", x).group()) if type(x) == str else x)
mangaData["nb_votes"] = mangaData["nb_votes"].apply(
    lambda x: int(x.replace(",", "")) if type(x) == str else x)
```

```
# plot the score and nb_votes
sns.scatterplot(x="score", y="nb_votes", data=mangaData)
# add labels
plt.xlabel("score")
plt.ylabel("nb_votes")
plt.title("score vs nb_votes")
```

```
Text(0.5, 1.0, 'score vs nb_votes')
```



No correlation

```
# info
mangaData.info()
# describe
mangaData.describe()
# head
mangaData.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1784 entries, 0 to 1783
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  1784 non-null   int64
 1   name        1784 non-null   object
 2   genre       1784 non-null   object
 3   author      1784 non-null   object
 4   status      1784 non-null   object
 5   alt_title   1784 non-null   object
 6   views       1784 non-null   float64
 7   score       1784 non-null   float64
 8   nb_votes    1784 non-null   int64
dtypes: float64(2), int64(2), object(5)
memory usage: 125.6+ KB
```

	Unnamed: 0	name	genre	author	status	alt_title	views	score	nb_votes
0	0	Doraemon	Adventure-Comedy-Fantasy-Schoollife-Scifi-Slic...	\nFujiko F. Fujio	Completed	ドラえもん; 哆啦A夢; 多啦A梦; 小叮噹; 小叮当; 机器猫; Đôrêmon; 萬能小...	4000000.0	4.79	255
1	1	The Honor Student's Secret Job	Comedy-Schoollife-Shounen	\nAzuma Yuki	Ongoing	Yuutousei to Himitsu no Oshigoto; 優等生と秘密のお仕事	3800000.0	4.01	476
2	2	Kore kara Dandan Shiawase ni Natte Iku Kowai O...	Comedy-Drama-Romance-Sliceoflife	\nYano Toshinori	Ongoing	これからだんだん幸せになっていく怖い 女上司 ; رئيسة مرعبة تزداد سعا...	195200.0	4.49	137
3	3	I Will Surrender the Position as Empress	Drama-Fantasy-Historical-Romance-Shoujo	\n마이구미 - 한보연	Ongoing	황후 자리를 버리겠습니다 ; I Will Give Up The Position as...	547400.0	4.73	259

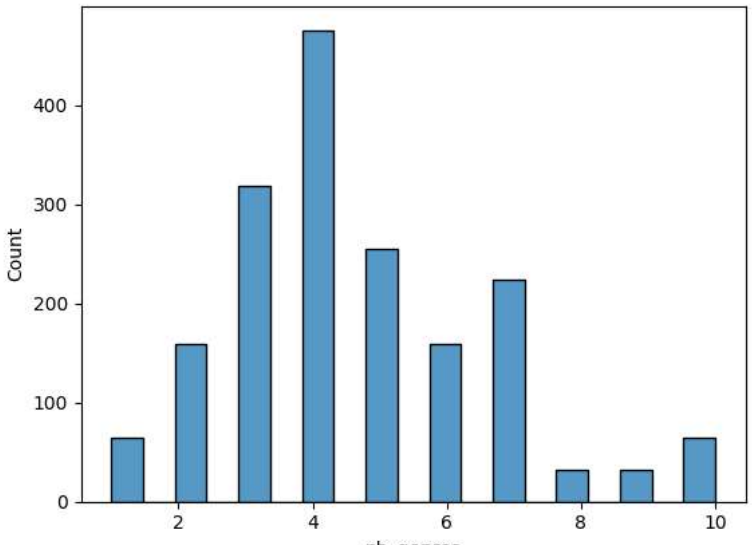
```
# genre is a string of genres seperated by a "-"
# split the genre into a list of genres
mangaData["genre"] = mangaData["genre"].apply(lambda x: x.split("-"))
# get the number of genres
mangaData["nb_genres"] = mangaData["genre"].apply(lambda x: len(x))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-57-fa8911cf7f00> in <cell line: 3>()
      1 # genre is a string of genres seperated by a "-"
      2 # split the genre into a list of genres
----> 3 mangaData["genre"] = mangaData["genre"].apply(lambda x: x.split("-"))
      4 # get the number of genres
      5 mangaData["nb_genres"] = mangaData["genre"].apply(lambda x: len(x))

4 frames
<ipython-input-57-fa8911cf7f00> in <lambda>(x)
      1 # genre is a string of genres seperated by a "-"
      2 # split the genre into a list of genres
----> 3 mangaData["genre"] = mangaData["genre"].apply(lambda x: x.split("-"))
      4 # get the number of genres
      5 mangaData["nb_genres"] = mangaData["genre"].apply(lambda x: len(x))

AttributeError: 'list' object has no attribute 'split'
```

RECHERCHER DANS STACK OVERFLOW

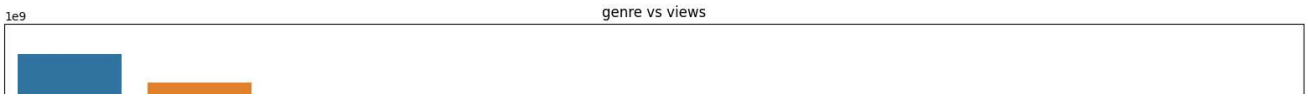
```
# plot the number of genres
sns.histplot(mangaData["nb_genres"])
```

```
<Axes: xlabel='nb_genres', ylabel='Count'>
```



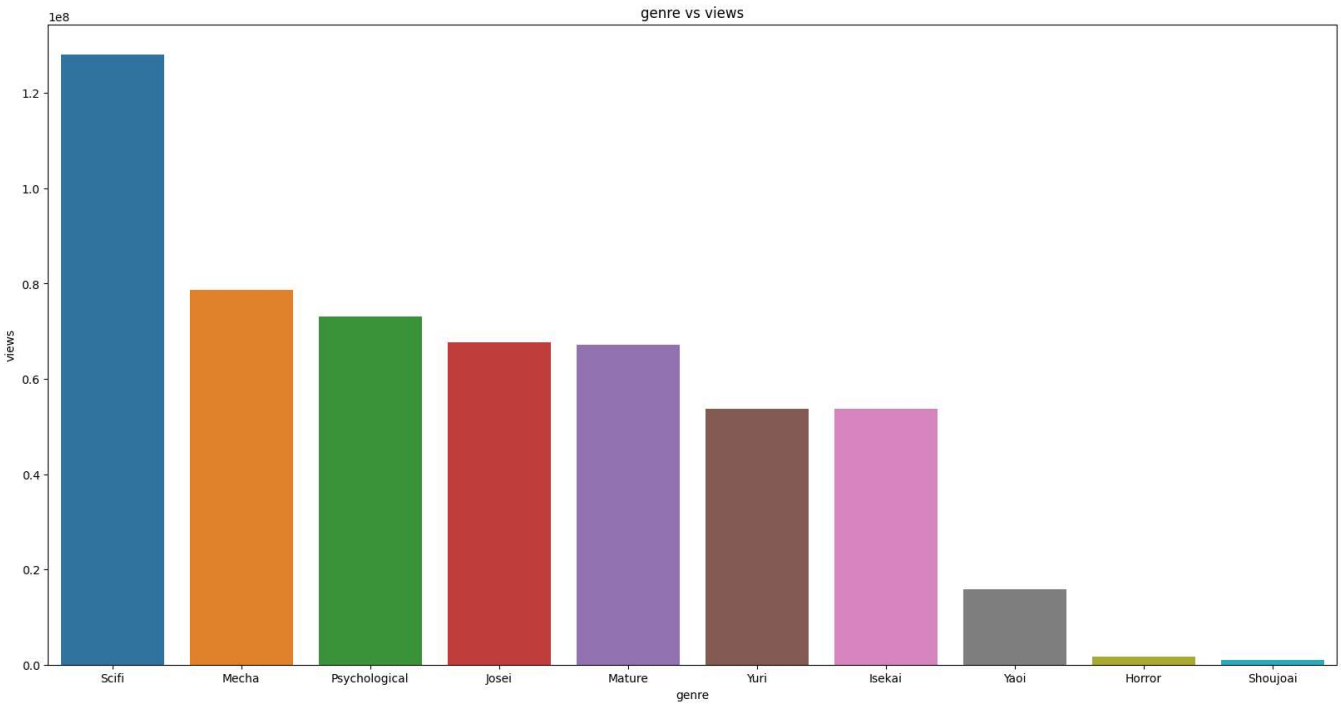
Let's see which genres preform the best

```
# genres with the most views
# get the genres
genres = mangaData["genre"].explode()
# get the views
views = mangaData["views"].explode()
# create a dataframe
genres_views = pd.DataFrame({"genre": genres, "views": views})
# group by genre and sum the views
genres_views = genres_views.groupby("genre").sum()
# sort the values
genres_views = genres_views.sort_values("views", ascending=False)
# plot the genres with the most views
sns.barplot(x=genres_views.index[:10], y="views", data=genres_views[:10])
# add labels
plt.xlabel("genre")
plt.ylabel("views")
plt.title("genre vs views")
plt.gcf().set_size_inches(20, 10)
```



Let's see the ones with the worst views

```
# plot the worst genres
sns.barplot(x=genres_views.index[-10:], y="views", data=genres_views[-10:])
# add labels
plt.xlabel("genre")
plt.ylabel("views")
plt.title("genre vs views")
# make it wider so the genres wont overlap
plt.gcf().set_size_inches(20, 10)
```

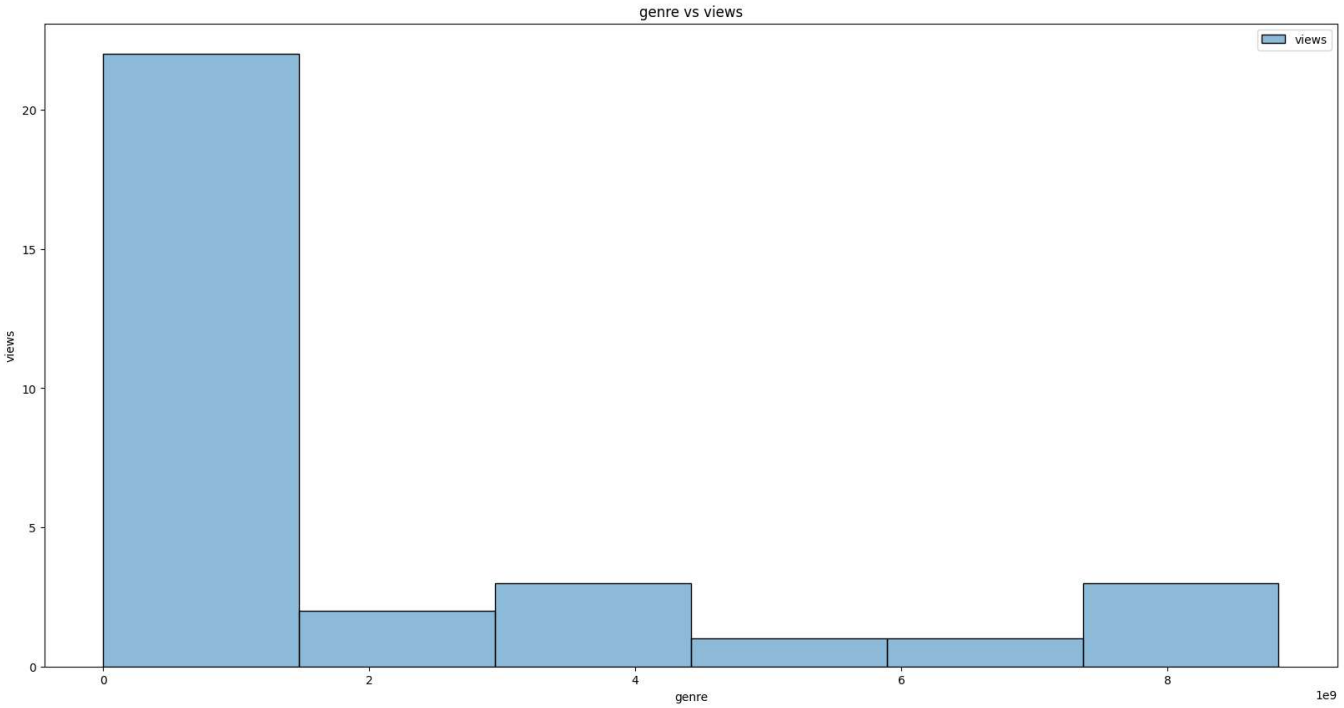


shoujou ai is lower than shonen ai

but yaoi is lower than yuri

weird

```
# plot the worst genres
sns.histplot(genres_views)
# add labels
plt.xlabel("genre")
plt.ylabel("views")
plt.title("genre vs views")
# make it wider so the genres wont overlap
plt.gcf().set_size_inches(20, 10)
```



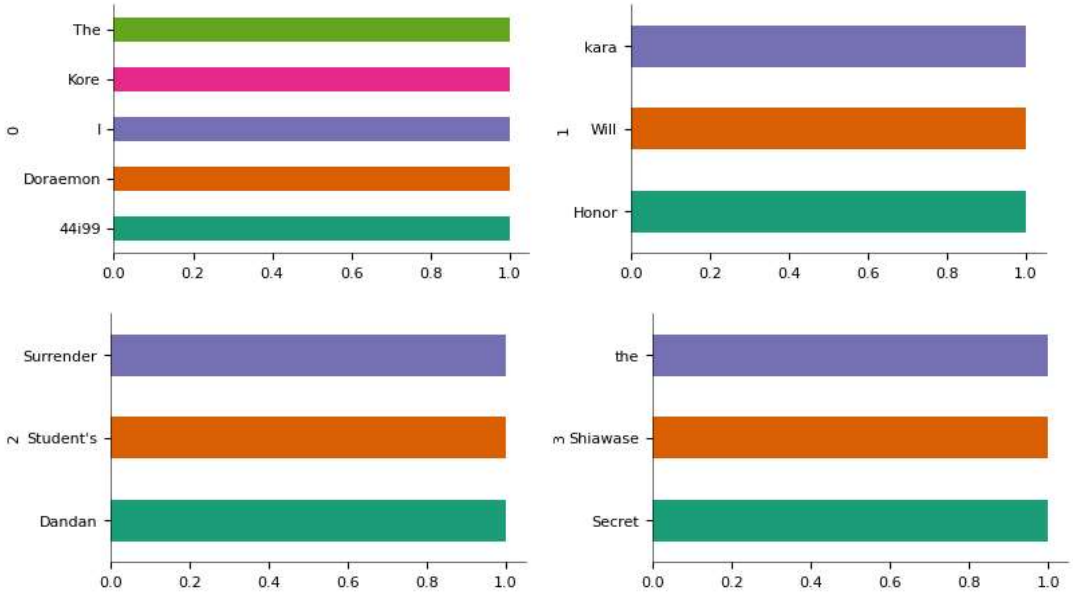
lets see the words

```
words = mangaData["name"].apply(lambda x: pd.Series(x.split(" ")))
# make the datafram with views set to zero so we can count
words_views = pd.DataFrame(0, index=words.index, columns=["views","frequency","score"])
# create a dataframe
print(f"words_views.shape: {words.shape}")
words.head(
)
```

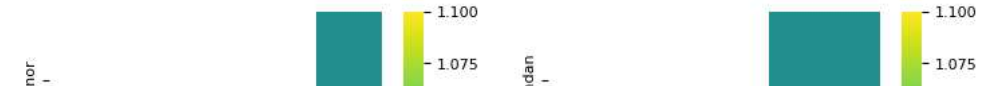
```
words_views.shape: (1784, 12)
```

	0	1	2	3	4	5	6	7	8	9	10	11
0	Doraemon	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	The	Honor	Student's	Secret	Job	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Kore	kara	Dandan	Shiawase	ni	Natte	Iku	Kowai	Onna	Joushi	NaN	NaN
3	I	Will	Surrender	the	Position	as	Empress	NaN	NaN	NaN	NaN	NaN
4	44i99	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Categorical distributions

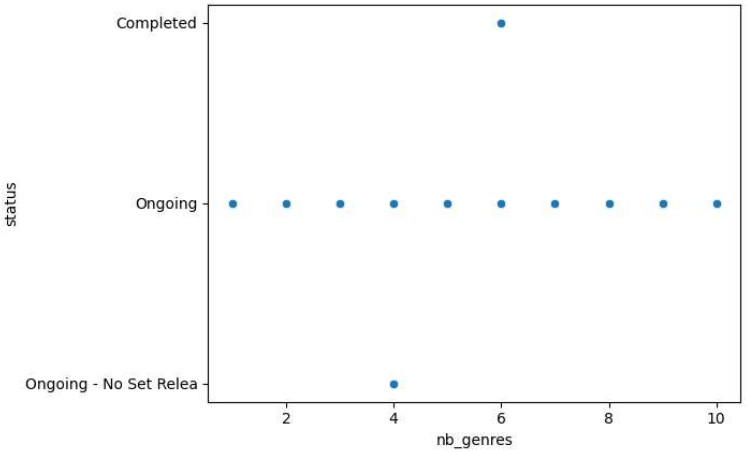


2-d categorical distributions




```
sns.scatterplot(x="nb_genres", y="status", data=mangaData)
```

```
<Axes: xlabel='nb_genres', ylabel='status'>
```

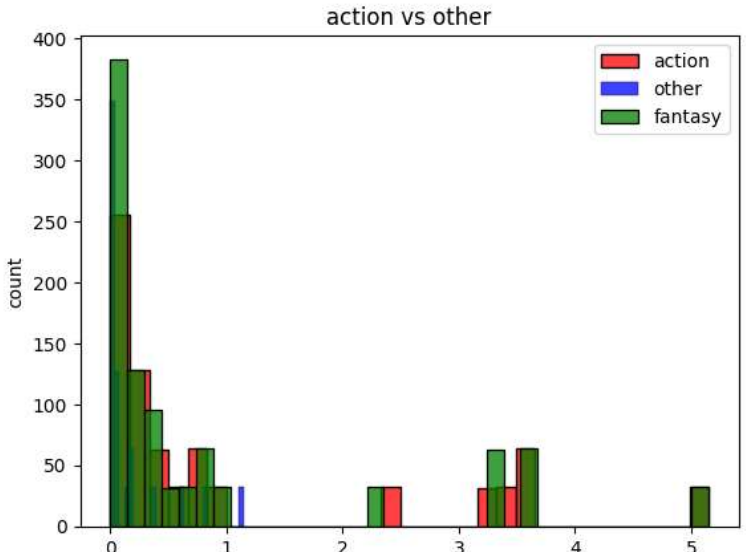


action genre vs the other genres

```
# views of the action genre compared to the other genres
# get the action genre
action = mangaData[mangaData["genre"].apply(lambda x: "Action" in x)]
# get the fantasy genre
fantasy = mangaData[mangaData["genre"].apply(lambda x: "Fantasy" in x)]
# get the other genres
other = mangaData[mangaData["genre"].apply(lambda x: "Action" not in x and "Fantasy" not in x)]
# plot the views of the action genre compared to the other genres
```

```
sns.histplot(action["views"], color="r", label="action", kde=False)
sns.histplot(other["views"], color="b", label="other", kde=False)
sns.histplot(fantasy["views"], color="g", label="fantasy", kde=False)
# add labels
plt.xlabel("views")
plt.ylabel("count")
plt.title("action vs other")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x799c4c818b80>
```



✓ 0 s terminée à 20:36