Untitled

February 17, 2018

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import ast
import datetime as dt
```

In [6]:
```
ted=pd.read_csv("ted_main.csv")
#Importing the csv formatted dataset into the kernel environment
```

In [7]:
```
df=ted[['comments', 'description', 'duration', 'event', 'film_date',
       'languages', 'main_speaker', 'name', 'num_speaker',
       'published_date', 'ratings', 'related_talks', 'speaker_occupation',
       'tags', 'title', 'url', 'views']]
#Data framing of all the attributes into the environemnt with name df
```

In [8]:
```
len(df)
```

Out[8]: 2550

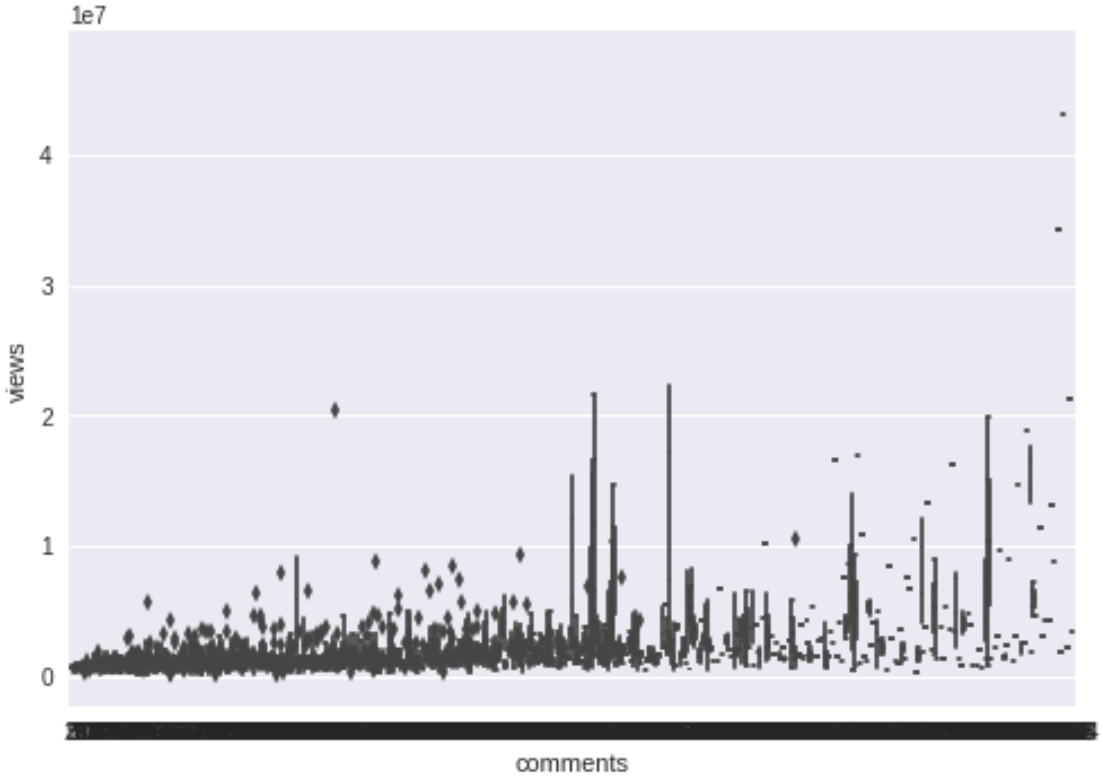
In [9]:
```
df.head()
#chekcking the upper values of the dataset
```

Out[9]:

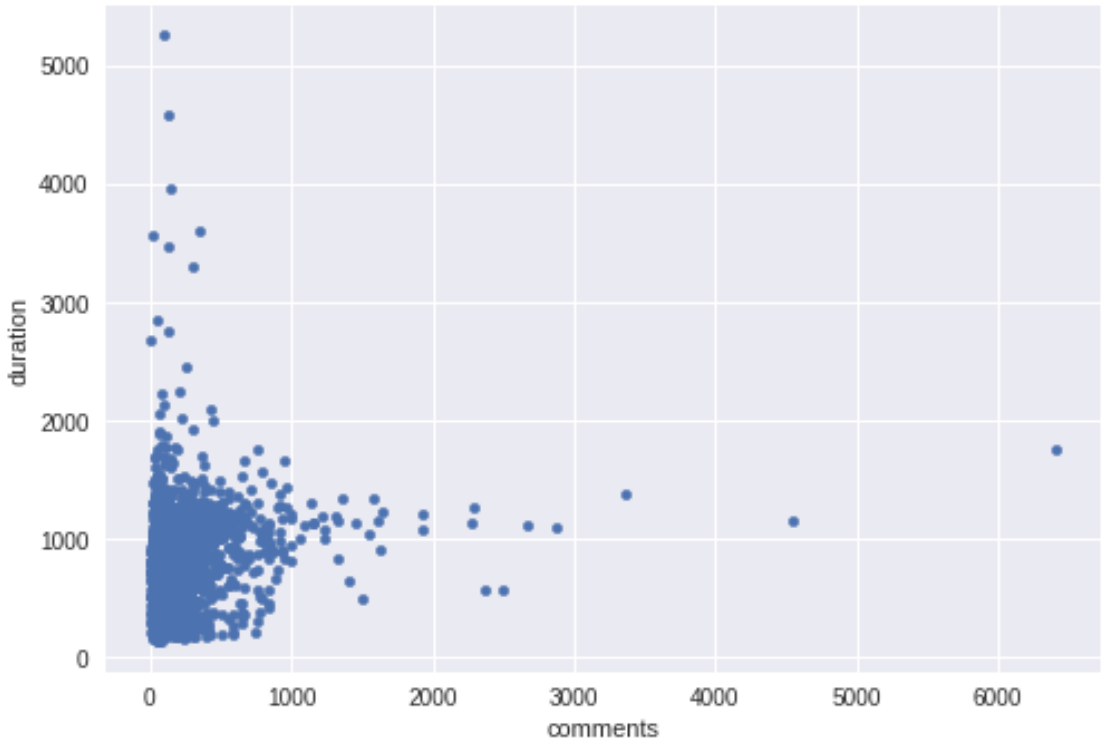
	comments	description	duration	event	film_date	languages	main_speaker	name	num_speaker	published_date
0	4553	Sir Ken Robinson makes an entertaining and pro...	1164	TED2006	1140825600	60	Ken Robinson			
1	265	With the same humor and humanity he exuded in ...	977	TED2006	1140825600	43	Al Gore			
2	124	New York Times columnist David Pogue takes aim...	1286	TED2006	1140739200	26	David Pogue			
3	200	In an emotionally charged talk, MacArthur-winn...	1116	TED2006	1140912000	35	Majora Carter			
4	593	You've never seen data presented like this. Wi...	1190	TED2006	1140566400	48	Hans Rosling			

```
0      Ken Robinson: Do schools kill creativity?           1      1151367060
1           Al Gore: Averting the climate crisis           1      1151367060
2                 David Pogue: Simplicity sells            1      1151367060
3            Majora Carter: Greening the ghetto            1      1151367060
4  Hans Rosling: The best stats you've ever seen           1      1151440680

                                              ratings  \
0  [{'id': 7, 'name': 'Funny', 'count': 19645}, {...
1  [{'id': 7, 'name': 'Funny', 'count': 544}, {'i...
2  [{'id': 7, 'name': 'Funny', 'count': 964}, {'i...
3  [{'id': 3, 'name': 'Courageous', 'count': 760}...
4  [{'id': 9, 'name': 'Ingenious', 'count': 3202}...

                                         related_talks  \
0  [{'id': 865, 'hero': 'https://pe.tedcdn.com/im...
1  [{'id': 243, 'hero': 'https://pe.tedcdn.com/im...
2  [{'id': 1725, 'hero': 'https://pe.tedcdn.com/i...
3  [{'id': 1041, 'hero': 'https://pe.tedcdn.com/i...
4  [{'id': 2056, 'hero': 'https://pe.tedcdn.com/i...

                   speaker_occupation  \
0                      Author/educator
1                     Climate advocate
2                 Technology columnist
3   Activist for environmental justice
4  Global health expert; data visionary

                                               tags  \
0  ['children', 'creativity', 'culture', 'dance',...
1  ['alternative energy', 'cars', 'climate change...
2  ['computers', 'entertainment', 'interface desi...
3  ['MacArthur grant', 'activism', 'business', 'c...
4  ['Africa', 'Asia', 'Google', 'demo', 'economic...

                          title  \
0     Do schools kill creativity?
1     Averting the climate crisis
2                Simplicity sells
3              Greening the ghetto
4  The best stats you've ever seen

                                                 url     views
0  https://www.ted.com/talks/ken_robinson_says_sc...  47227110
1  https://www.ted.com/talks/al_gore_on_averting_...   3200520
2  https://www.ted.com/talks/david_pogue_says_sim...   1636292
3  https://www.ted.com/talks/majora_carter_s_tale...   1697550
4  https://www.ted.com/talks/hans_rosling_shows_t...  12005869
```

```
In [10]: dateconv=np.vectorize(dt.datetime.utcfromtimestamp)
         df['film_date']=dateconv(df['film_date'])
         #converting the film_date attribute from Unix time stamp to human readbale form

In [11]: dateconv=np.vectorize(dt.datetime.utcfromtimestamp)
         df['published_date']=dateconv(df['published_date'])
         #converting the published_date attribute from Unix time stamp to human readbale form

In [12]: sns.boxplot(x=df['comments'],y=df['views'])

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2bb832ed68>

In [13]: sns.plt.show()
         #Checking the spread of data and the peak values
```

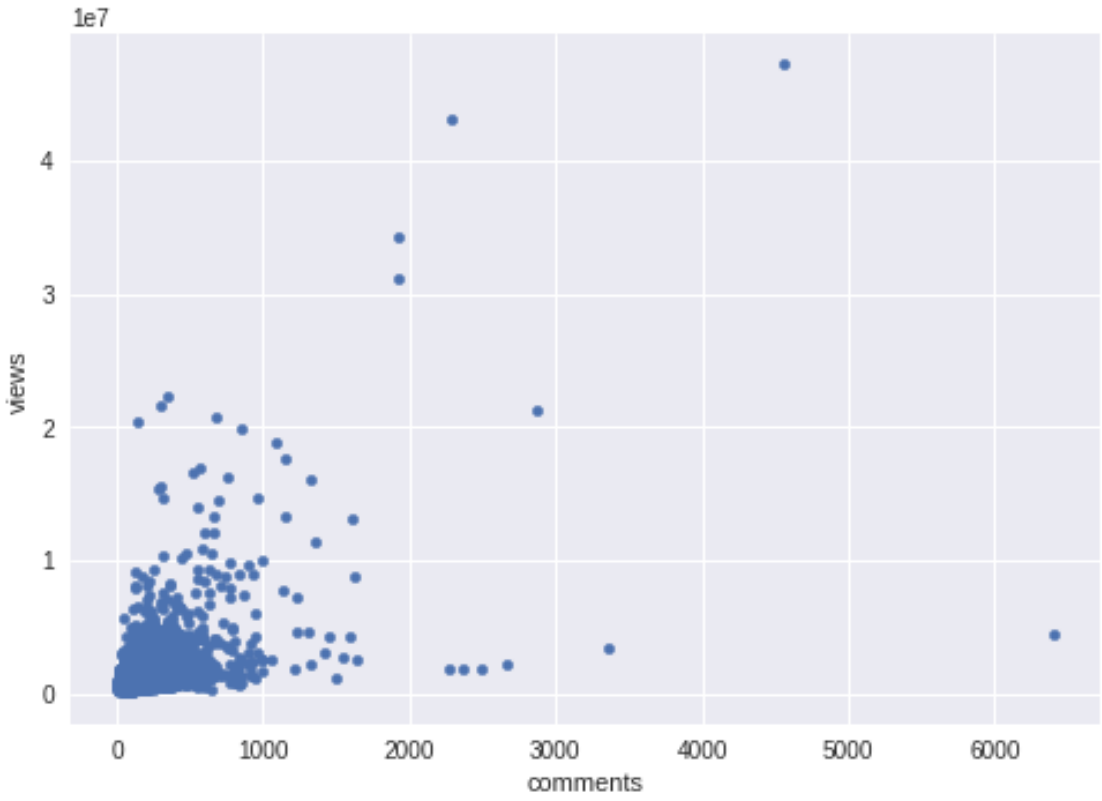



```
In [14]: df.plot.scatter("comments","duration")
         plt.xlabel("comments")
         plt.ylabel("duration")
         plt.show()
         #visualizng the scattering of data and outliers
```




```
In [15]: max(df['comments'])
         #maximum comments out of total

Out[15]: 6404

In [16]: max(df['views'])

Out[16]: 47227110

In [17]: max(df['duration'])

Out[17]: 5256

In [18]: np.corrcoef(df['comments'],df['duration'])[0,1]
         #finding corelation coefficient between comments and duration

Out[18]: 0.14069364760321112

In [19]: np.corrcoef(df['comments'],df['views'])[0,1]

Out[19]: 0.53093870062136805

In [20]: np.corrcoef(df['duration'],df['views'])[0,1]

Out[20]: 0.04874042904795986
```

In [21]:
```
df.plot.scatter("comments","views")
plt.show()
```



In [22]:
```
df[['title','main_speaker']][df.comments==max(df.comments)]
#check to see who got maximum number of comments
```

Out[22]:

	title	main_speaker
96	Militant atheism	Richard Dawkins

In [23]:
```
df[['title','main_speaker']][df.views==max(df.views)]
#check to see who got maximum number of views
```

Out[23]:

	title	main_speaker
0	Do schools kill creativity?	Ken Robinson

In [24]:
```
df[['title','main_speaker','views','comments','duration']][df['main_speaker'].str.conta
#check to see the tabular form of views,comments,duration of Ken Robinson
```

Out[24]:

	title	main_speaker	views	\
0	Do schools kill creativity?	Ken Robinson	47227110	
692	Bring on the learning revolution!	Ken Robinson	7266316	
833	Changing education paradigms	Ken Robinson	1854997	

```
1502  How to escape education's death valley  Ken Robinson   6657858

      comments  duration
0         4553      1164
692       1234      1008
833        473       700
1502       634      1151
```

In [25]:
```
df[['title','main_speaker','views','comments','duration']][df['main_speaker'].str.conta
#of Richard Dawkins
```

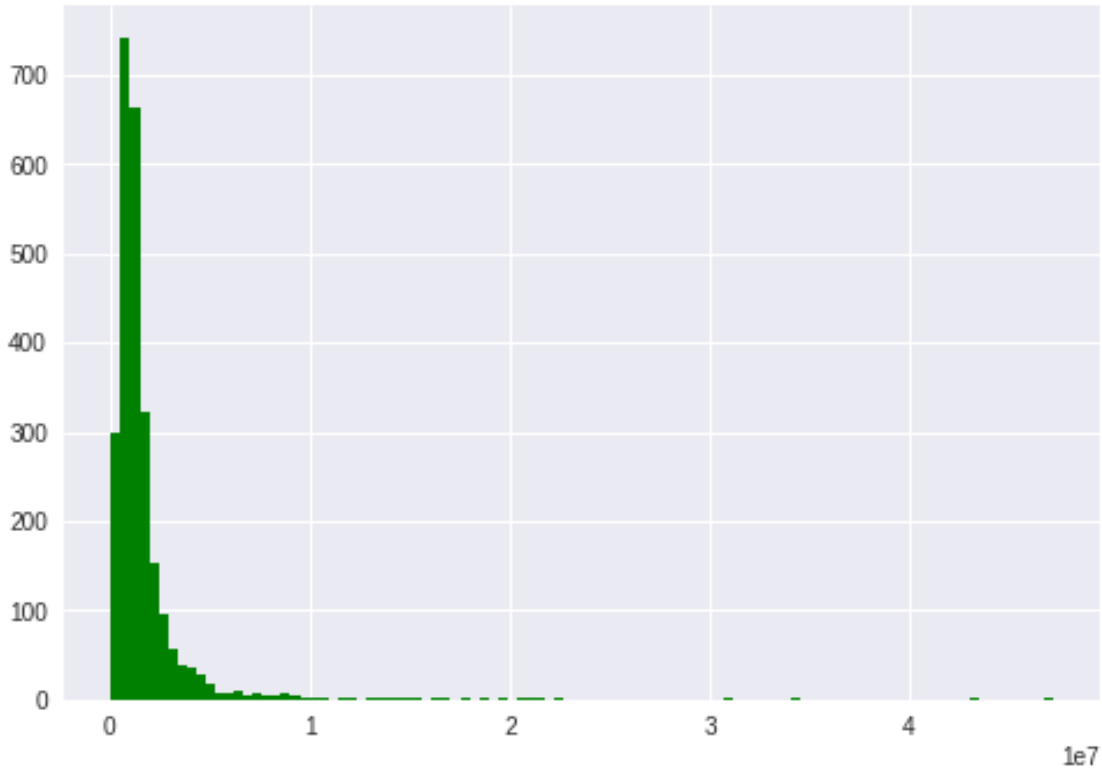
Out[25]:
```
                                 title     main_speaker    views  comments  \
26   Why the universe seems so strange  Richard Dawkins  2885999       507
96                    Militant atheism  Richard Dawkins  4374792      6404
600         Growing up in the universe  Richard Dawkins   318423       133

     duration
26       1316
96       1750
600      3475
```

In [26]:
```
plt.hist(df['views'],bins=100,color='green')
#histogram of views to see the skewness of the data
```

Out[26]:
```
(array([ 300.,  741.,  664.,  322.,  154.,   95.,   57.,   38.,   36.,
          29.,   18.,    8.,    8.,   11.,    4.,    8.,    5.,    5.,
           7.,    4.,    2.,    3.,    3.,    0.,    1.,    2.,    0.,
           1.,    2.,    1.,    1.,    2.,    2.,    0.,    2.,    2.,
           0.,    1.,    0.,    1.,    0.,    1.,    0.,    2.,    1.,
           1.,    0.,    1.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    1.,    0.,    0.,    0.,    0.,    0.,    0.,
           1.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    1.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    1.]),
 array([    50443.        ,    522209.67      ,    993976.34      ,
         1465743.01      ,   1937509.68      ,   2409276.35      ,
         2881043.02      ,   3352809.69      ,   3824576.36      ,
         4296343.03      ,   4768109.7       ,   5239876.37      ,
         5711643.04      ,   6183409.71      ,   6655176.38      ,
         7126943.05      ,   7598709.72      ,   8070476.39      ,
         8542243.06      ,   9014009.73      ,   9485776.4       ,
         9957543.07      ,  10429309.74      ,  10901076.41      ,
        11372843.08      ,  11844609.75      ,  12316376.42      ,
        12788143.09      ,  13259909.76      ,  13731676.43      ,
        14203443.1       ,  14675209.77      ,  15146976.44      ,
        15618743.11      ,  16090509.78      ,  16562276.45      ,
        17034043.12      ,  17505809.79      ,  17977576.46      ,
        18449343.13      ,  18921109.8       ,  19392876.47      ,
```

```
        19864643.14      ,  20336409.81      ,  20808176.48      ,
        21279943.15      ,  21751709.82      ,  22223476.49      ,
        22695243.16      ,  23167009.83      ,  23638776.5       ,
        24110543.17      ,  24582309.84      ,  25054076.51      ,
        25525843.18      ,  25997609.85      ,  26469376.52      ,
        26941143.19      ,  27412909.86      ,  27884676.53      ,
        28356443.2       ,  28828209.87      ,  29299976.54      ,
        29771743.21      ,  30243509.88      ,  30715276.55      ,
        31187043.22      ,  31658809.89      ,  32130576.56      ,
        32602343.23      ,  33074109.9       ,  33545876.57      ,
        34017643.24      ,  34489409.91      ,  34961176.58      ,
        35432943.25      ,  35904709.92      ,  36376476.59      ,
        36848243.26      ,  37320009.93      ,  37791776.6       ,
        38263543.27      ,  38735309.94      ,  39207076.61      ,
        39678843.28      ,  40150609.95      ,  40622376.62      ,
        41094143.29      ,  41565909.96      ,  42037676.63      ,
        42509443.3       ,  42981209.97      ,  43452976.64      ,
        43924743.30999999,  44396509.98      ,  44868276.65      ,
        45340043.32      ,  45811809.99      ,  46283576.66      ,
        46755343.33      ,  47227110.        ]),
 <a list of 100 Patch objects>)
```

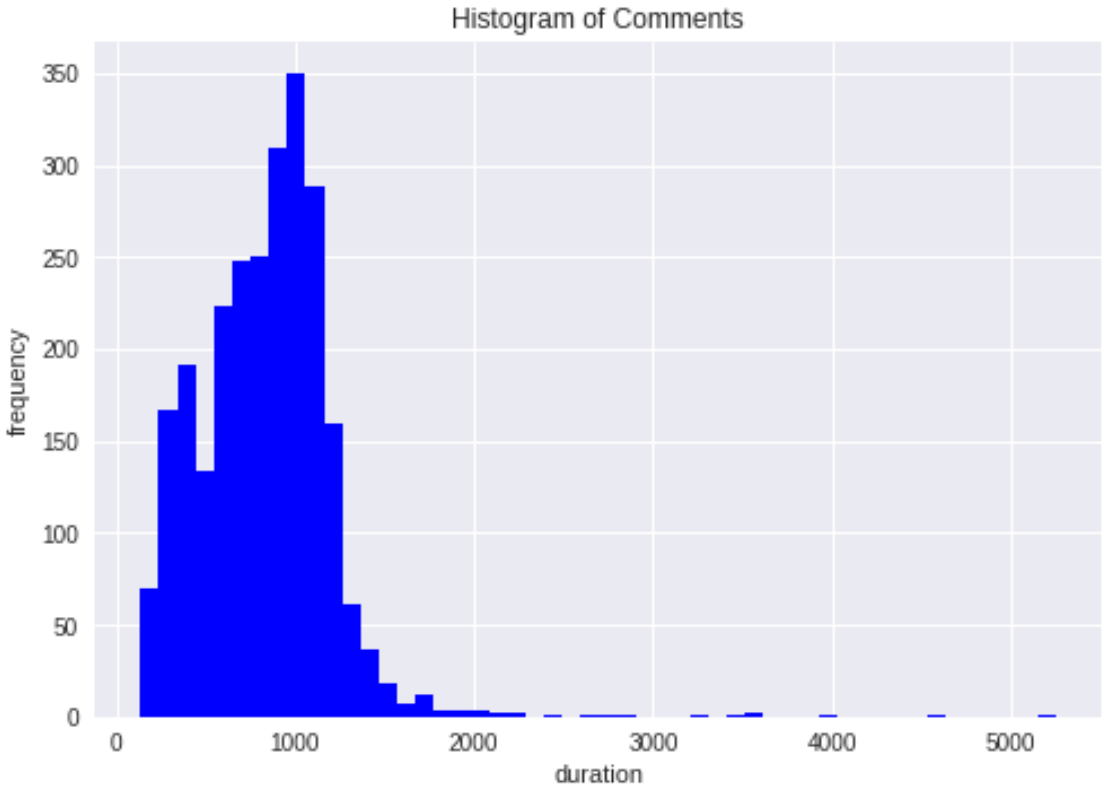
In [27]:
```
plt.show()
```



```
In [28]: plt.hist(df['duration'],bins=50,color='blue')
```

```
Out[28]: (array([  70.,  167.,  192.,  134.,  223.,  248.,  250.,  309.,  350.,
                  288.,  160.,   61.,   37.,   18.,    7.,   12.,    3.,    3.,
                    3.,    2.,    2.,    0.,    1.,    0.,    1.,    1.,    1.,
                    0.,    0.,    0.,    1.,    0.,    1.,    2.,    0.,    0.,
                    0.,    1.,    0.,    0.,    0.,    0.,    0.,    1.,    0.,
                    0.,    0.,    0.,    0.,    1.]),
          array([  135.  ,   237.42,   339.84,   442.26,   544.68,   647.1 ,
                   749.52,   851.94,   954.36,  1056.78,  1159.2 ,  1261.62,
                  1364.04,  1466.46,  1568.88,  1671.3 ,  1773.72,  1876.14,
                  1978.56,  2080.98,  2183.4 ,  2285.82,  2388.24,  2490.66,
                  2593.08,  2695.5 ,  2797.92,  2900.34,  3002.76,  3105.18,
                  3207.6 ,  3310.02,  3412.44,  3514.86,  3617.28,  3719.7 ,
                  3822.12,  3924.54,  4026.96,  4129.38,  4231.8 ,  4334.22,
                  4436.64,  4539.06,  4641.48,  4743.9 ,  4846.32,  4948.74,
                  5051.16,  5153.58,  5256.  ]),
          <a list of 50 Patch objects>)
```

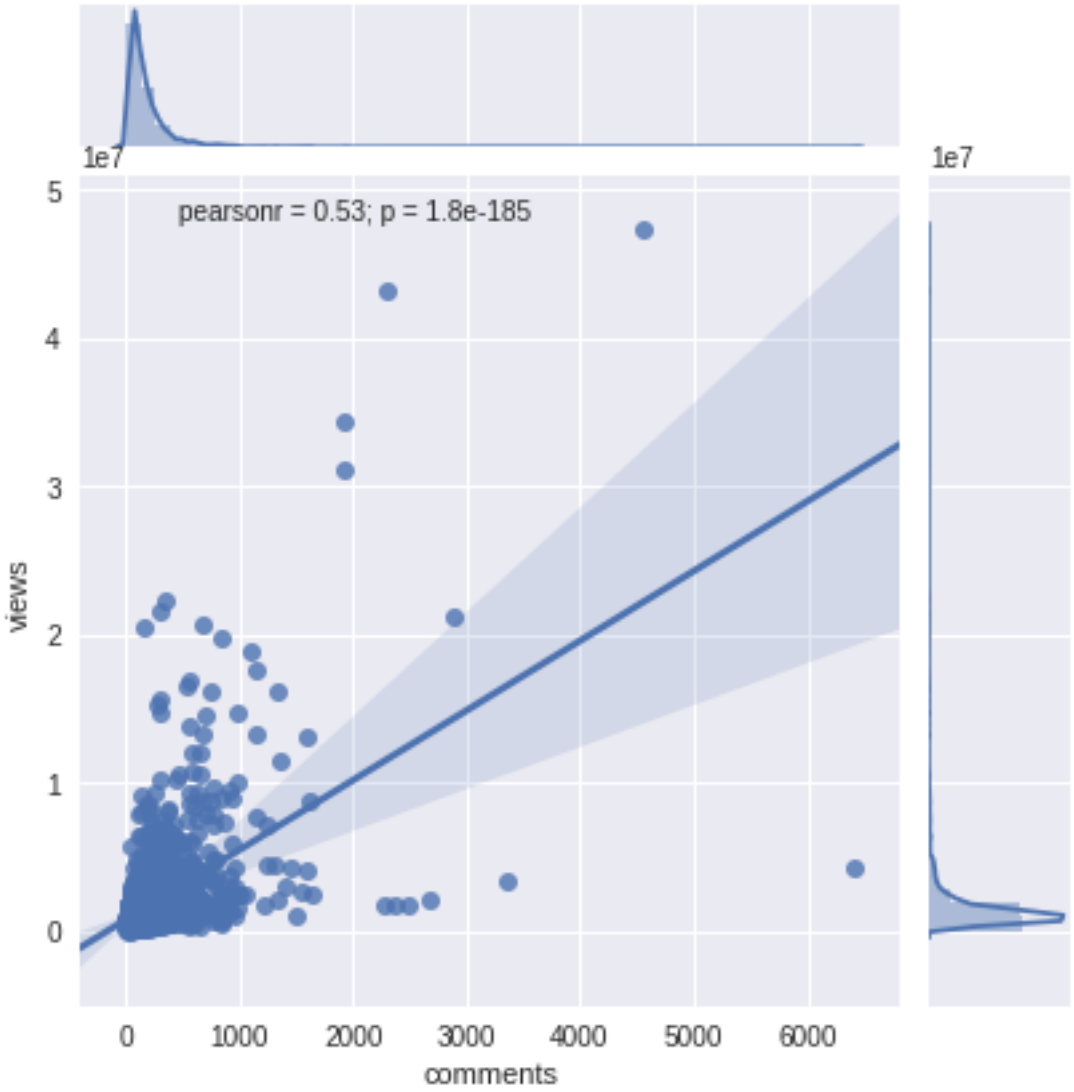
```
In [29]: plt.xlabel("duration")
         plt.ylabel("frequency")
         plt.title("Histogram of Comments")
         plt.show()
```



In [30]:
```
sns.jointplot(x=df['comments'],y=df['views'],kind='reg')
#joint plot to see the corelation and spread of data all together
```

Out[30]:
```
<seaborn.axisgrid.JointGrid at 0x7f2b3c3235c0>
```

In [31]:
```
sns.plt.show()
```

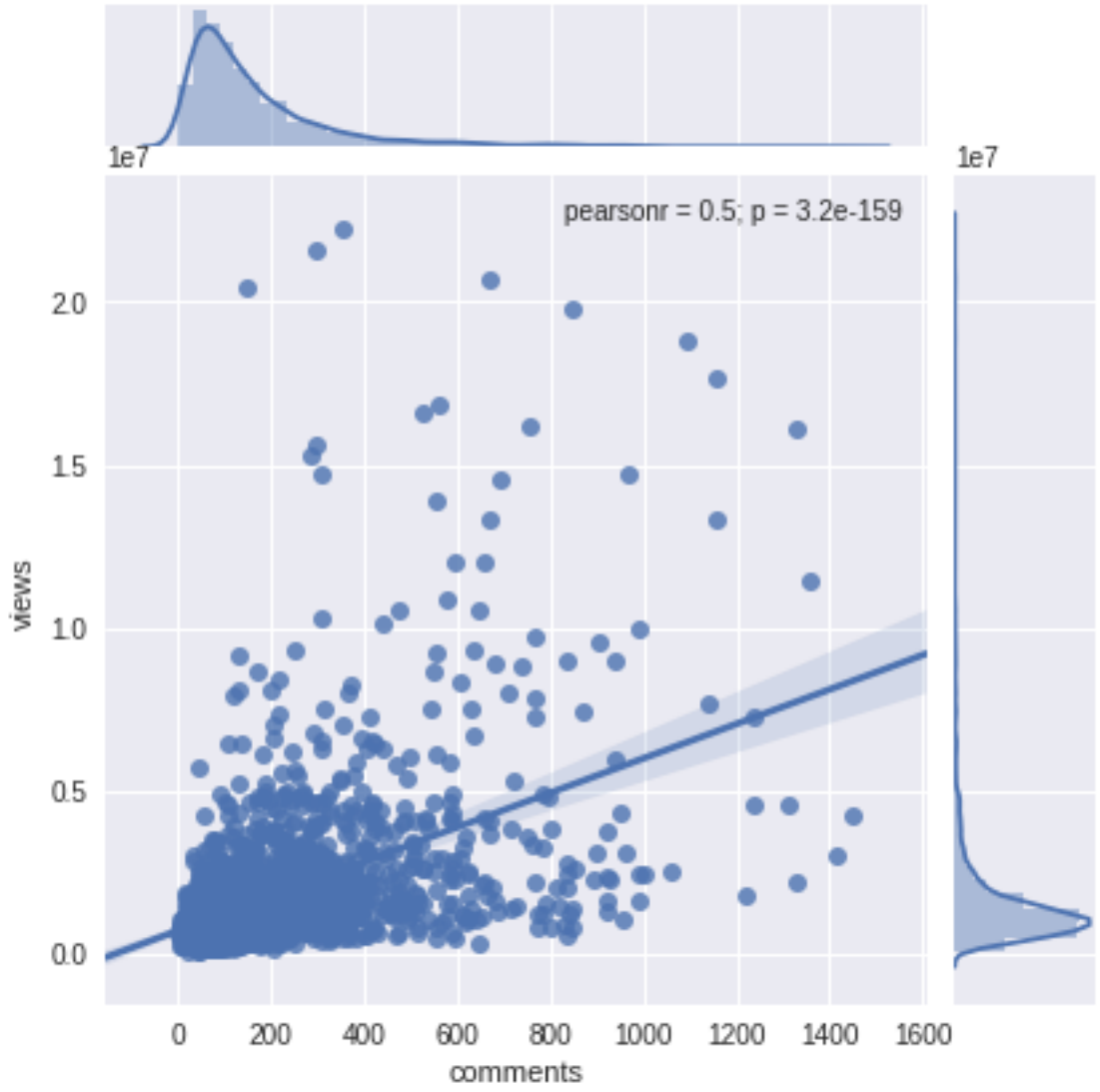


In [32]:
```
df_comments=df[df['comments']<1500]
#deleting the some outbound values
```

In [33]:
```
sns.jointplot(x=df_comments['comments'],y=df['views'],kind='reg')
#check to see the effect on corelation after deleting outbound values
```

Out[33]:
```
<seaborn.axisgrid.JointGrid at 0x7f2b3c21a668>
```

In [34]: sns.plt.show()



In [35]: df['ratings'][1]
#checking the dictionary in ratings column

Out[35]: "[{'id': 7, 'name': 'Funny', 'count': 544}, {'id': 3, 'name': 'Courageous', 'count': 13

In [36]: categories=set([])

In [37]: def split_ratings(ratings):
    val=ast.literal_eval(ratings)
    for rating in val:
        categories.add(rating['name'])

```
In [38]: df['ratings'].apply(split_ratings)
```

```
Out[38]: 0       None
         1       None
         2       None
         3       None
         4       None
         5       None
         6       None
         7       None
         8       None
         9       None
         10      None
         11      None
         12      None
         13      None
         14      None
         15      None
         16      None
         17      None
         18      None
         19      None
         20      None
         21      None
         22      None
         23      None
         24      None
         25      None
         26      None
         27      None
         28      None
         29      None
                 ...
         2520    None
         2521    None
         2522    None
         2523    None
         2524    None
         2525    None
         2526    None
         2527    None
         2528    None
         2529    None
         2530    None
         2531    None
         2532    None
         2533    None
         2534    None
```

```
2535    None
2536    None
2537    None
2538    None
2539    None
2540    None
2541    None
2542    None
2543    None
2544    None
2545    None
2546    None
2547    None
2548    None
2549    None
Name: ratings, dtype: object
```

```
In [39]: categories
```

```
Out[39]: {'Beautiful',
          'Confusing',
          'Courageous',
          'Fascinating',
          'Funny',
          'Informative',
          'Ingenious',
          'Inspiring',
          'Jaw-dropping',
          'Longwinded',
          'OK',
          'Obnoxious',
          'Persuasive',
          'Unconvincing'}
```

```
In [40]: def get_count_from_ratings(ratings,col_name):
             val=ast.literal_eval(ratings)
             for rating in val:
                 if rating['name']==col_name:
                         return rating['count']
```

```
In [41]: for name in categories:
             df[name]=df['ratings'].apply(lambda rating: get_count_from_ratings(rating,col_name=
```