Chapter_7_Create_Dataset_Scraping_Google_Play

January 29, 2022

1 Chapter 7: Scraping Google Play Reviews

Imports

[ ]:
```
!pip install -qq google_play_scraper
```

```
     |                  | 52 kB 824 kB/s
  Building wheel for google-play-scraper (setup.py) … done
```

[ ]:
```
# files
import json
import pandas as pd

#progress bar
from tqdm import tqdm

#formatters
from pygments import highlight
from pygments.lexers import JsonLexer
from pygments.formatters import TerminalFormatter

# scraper
from google_play_scraper import Sort, reviews, app

# plotting
import seaborn as sns
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 20,8

#linalg
import numpy as np
```

Defining Apps to Scrape

[ ]:
```
# we define the languages and the country
LANGUAGE,COUNTRY = 'en','us'
# we will use some of the top apps from the US
# this data comes from AppAnnie which is a paid service :(
```

```
app_packages=[
              'com.anydo','com.todoist', 'com.ticktick.task',
              'com.habitrpg.android.habitica', 'cc.forestapp',
              'com.oristats.habitbull', 'com.levor.liferpgtasks',
              'com.habitnow', 'com.microsoft.todos', 'prox.lab.calclock',
              'com.gmail.jmartindev.timetune', 'com.artfulagenda.app',
              'com.tasks.android', 'com.appgenix.bizcal', 'com.appxy.planner'
]
```

Testing the Google Play Scraper

[ ]:
```
# lets see what information the google play scraper gives us
# a random sample
import numpy as np
sample = np.random.choice(app_packages)
print(sample)
# info
info = app(sample, lang=LANGUAGE, country=COUNTRY)
# lets check how many keys we have --> 52 keys!
print(len(info.keys()))
del info['comments'] # delete the comments
info # look at the data
```

Extracting Google Play Reviews

[ ]:
```
def collect_info(LIST_OF_APPS,LANGUAGE,COUNTRY):
  '''
  Helper function which extracts the information from a given app-name
  returns a json file with 52 keys, we will use 51 because we are
  removing the comments section!
  '''
  app_info = []
  for apps in tqdm(LIST_OF_APPS):
    info = app(apps, lang=LANGUAGE, country=COUNTRY)
    del info['comments']
    app_info.append(info)
  return app_info

# also make a pretty printer

def pretty_print_json(JSON_OBJECT):
  '''
   Color codes the keys and the values in a JSON object
  '''
  json_file = json.dumps(
                          JSON_OBJECT,
                         indent=2,
```

```
                          sort_keys=True,
                          default=str
                             )
  print(highlight(json_file, JsonLexer(), TerminalFormatter()))
```

[ ]:
```
# creating the app info list
app_info = collect_info(app_packages, LANGUAGE, COUNTRY)
```

```
100%|      | 15/15 [00:02<00:00,  5.32it/s]
```

[ ]:
```
# now we can pretty print the file
# lets get a random sample
random_app = np.random.choice(app_info)

pretty_print_json(random_app)
```

Checking out the reviews part of GPS

[ ]:
```
# now we need to extract the reviews
revs,k = reviews(sample, lang=LANGUAGE, country=COUNTRY,sort=Sort.MOST_RELEVANT)
print(len(revs))

len(revs[0])
```

```
100
```

[ ]:
```
10
```

Extracting the Reviews - ACTUALLY!

[ ]:
```
def extract_gps_reviews(LIST_OF_APPS,LANGUAGE, COUNTRY):
  # instantiate an empty list
  l = []
  for apps in tqdm(LIST_OF_APPS):
    # we want to go in order for each of the scores
    for val in list(range(1,6)):
      for sort_order in [Sort.MOST_RELEVANT,Sort.NEWEST]:
        REVIEWS,_ = reviews(apps,
                            lang=LANGUAGE,
                            country=COUNTRY,
                            sort = sort_order,
                            count = 200 if val == 3 else 100,
                            filter_score_with = val)
        for k in REVIEWS:
          k['sortOrder'] = 'most_relevant' if sort_order == Sort.MOST_RELEVANT␣
↪else 'newest'
          k['appId'] = apps
        l.extend(REVIEWS)
```

```
  return l
```

[ ]:
```
app_reviews = extract_gps_reviews(app_packages, LANGUAGE, COUNTRY)
```

```
100%|      | 15/15 [01:06<00:00,  4.47s/it]
```

[ ]:
```
df = pd.DataFrame(app_reviews)
df.to_csv('GOOGLE_PLAY_REVIEWS.csv')
```

[ ]:
```
df.head()
```

[ ]:

	reviewId	…	appId
0	gp:AOqpTOH85sc18Ajgcgj6-IGmA7Gp34fVsrbyBJ274IZ…	…	com.anydo
1	gp:AOqpTOGxyMqOStnhbQ_mLfnLUfd1DHAt5uRXqDNArML…	…	com.anydo
2	gp:AOqpTOHgR1qnD3AZbHvKJ6-BbO4pMkokJS2JT1UvdVI…	…	com.anydo
3	gp:AOqpTOH_GtAiezLqm0tHyquE1arU2C_L__IFTeqJxsg…	…	com.anydo
4	gp:AOqpTOEvo7a7-HX3iYDQ-FPQOgQUw-kA5ajCkrf2ENy…	…	com.anydo

[5 rows x 12 columns]

EDA

[ ]:
```
# checking the number of samples we have
N_SAMPLES = df.shape[0]
txt = f"The number of samples is {N_SAMPLES}\n"
print(txt)

# also checking the columns that we have
[x for x in df.columns]
```

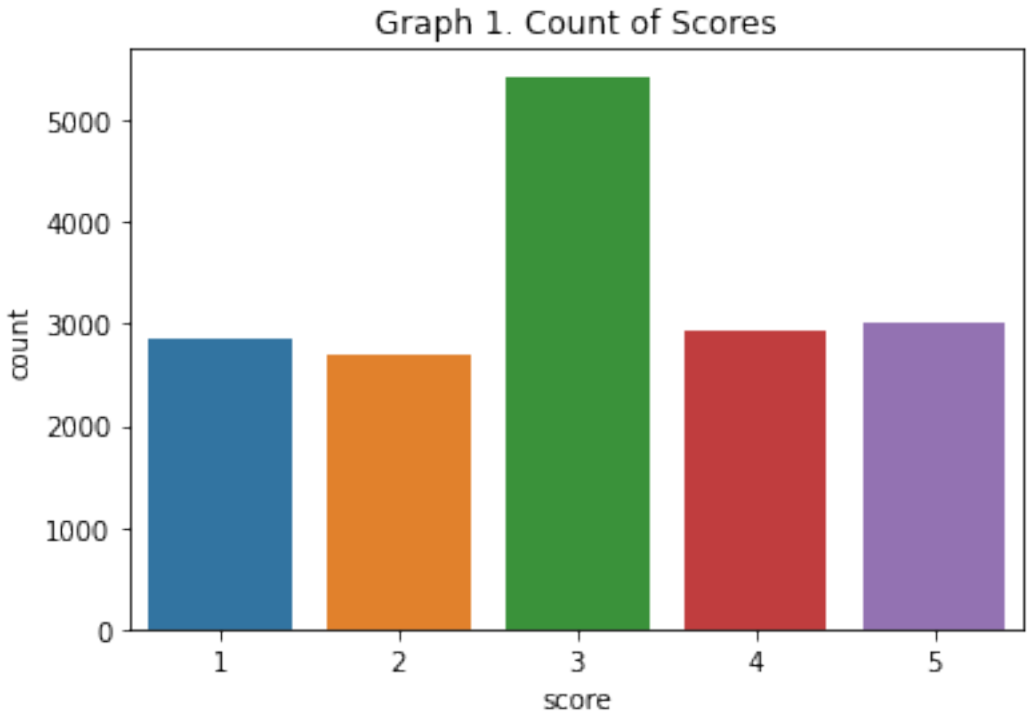
```
The number of samples is 16914
```

[ ]:
```
['reviewId',
 'userName',
 'userImage',
 'content',
 'score',
 'thumbsUpCount',
 'reviewCreatedVersion',
 'at',
 'replyContent',
 'repliedAt',
 'sortOrder',
 'appId']
```

[ ]:
```
# are all the usernames unique?
all_users = df['userName'].values
num_users = len(all_users)
print(f"The number of users is {num_users}")

# now lets check the unique users
num_users_unique = len(df['userName'].unique())
print(f"The number of unique users is {num_users_unique}")

# what is the difference
diffs = abs(num_users - num_users_unique)
print(f"The number of duplicated users is {diffs}")
```

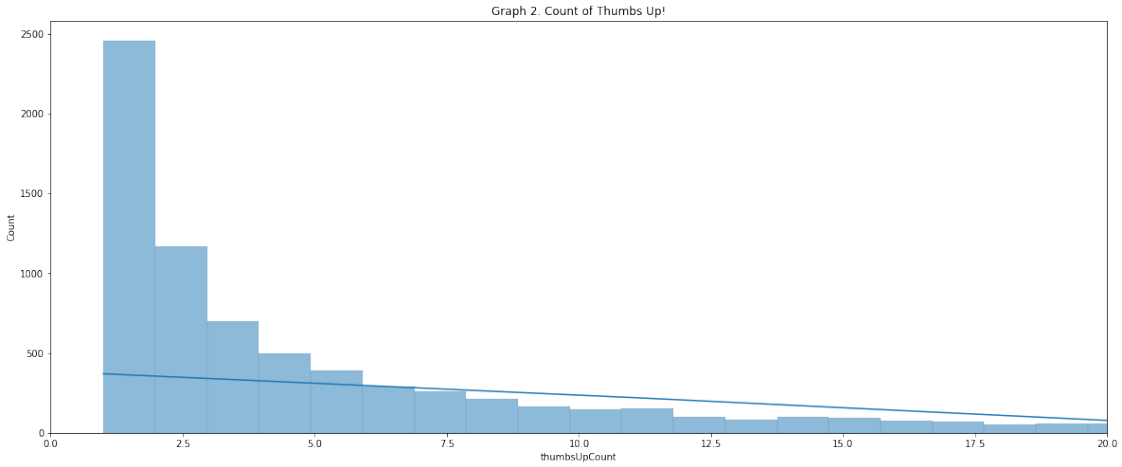
```
The number of users is 16914
The number of unique users is 12665
The number of duplicated users is 4249
```

[ ]:
```
g = sns.countplot(x='score',data=df)
g.set_title('Graph 1. Count of Scores')
plt.show()
```



[ ]:
```
# filter the df for values greater than 0
filtered_df = df[df['thumbsUpCount'] != 0]
g = sns.histplot(filtered_df['thumbsUpCount'],kde=True)
```

```
g.set_xlim([0,20])
g.set_title('Graph 2. Count of Thumbs Up!')
plt.show()
```

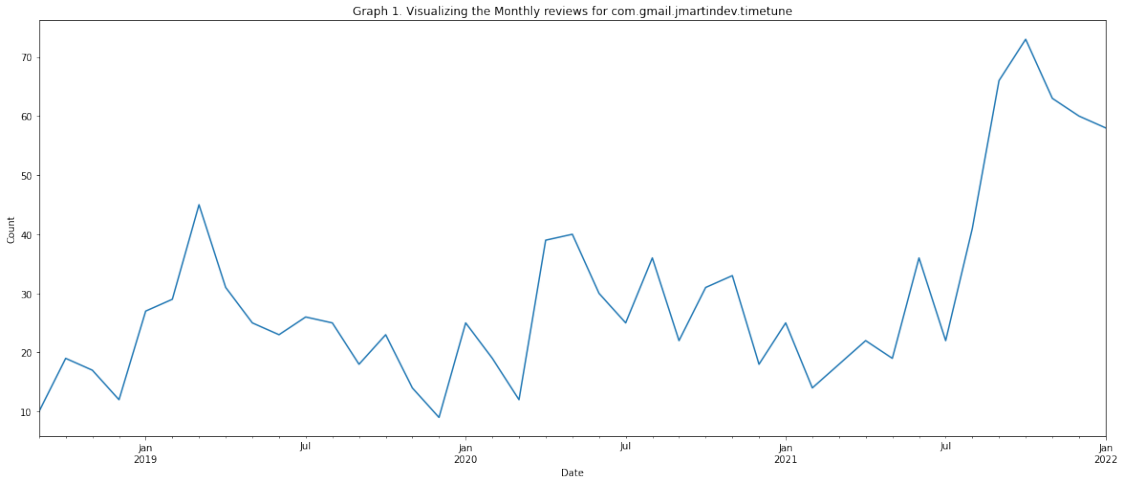


[ ]:
```
df.dtypes
```

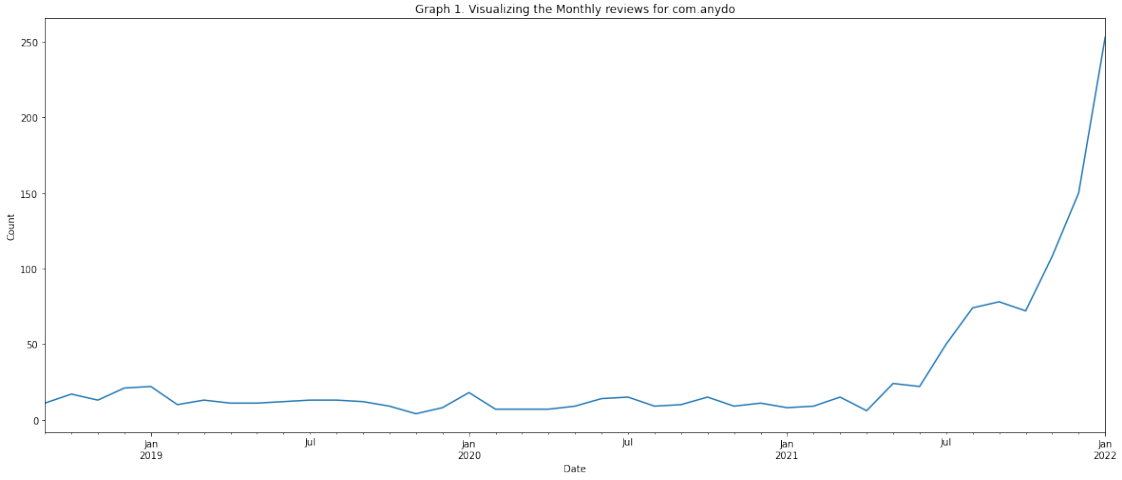
[ ]:
```
reviewId                    object
userName                    object
userImage                   object
content                     object
score                        int64
thumbsUpCount                int64
reviewCreatedVersion        object
at                  datetime64[ns]
replyContent                object
repliedAt           datetime64[ns]
sortOrder                   object
appId                       object
dtype: object
```

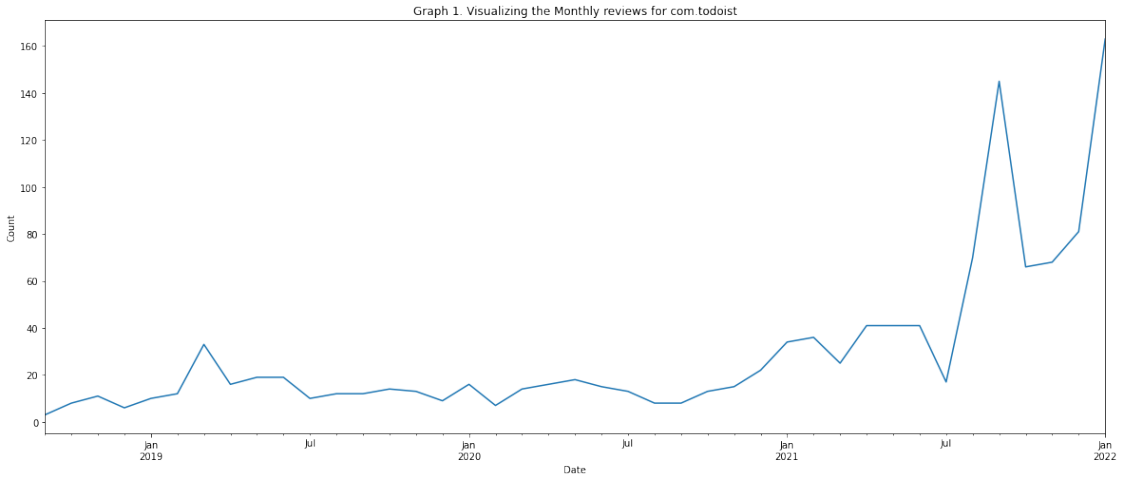
[ ]:
```
'''
RESAMPLING DOCUMENTATION HERE
https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.
 ↪html#offset-aliases
D = Day
W = Weekly
M = Monthly
A, Y = Annualy / Yearly
'''

def visualize_reviews(DATAFRAME, SAMPLING_TYPE, APP_ID):
```

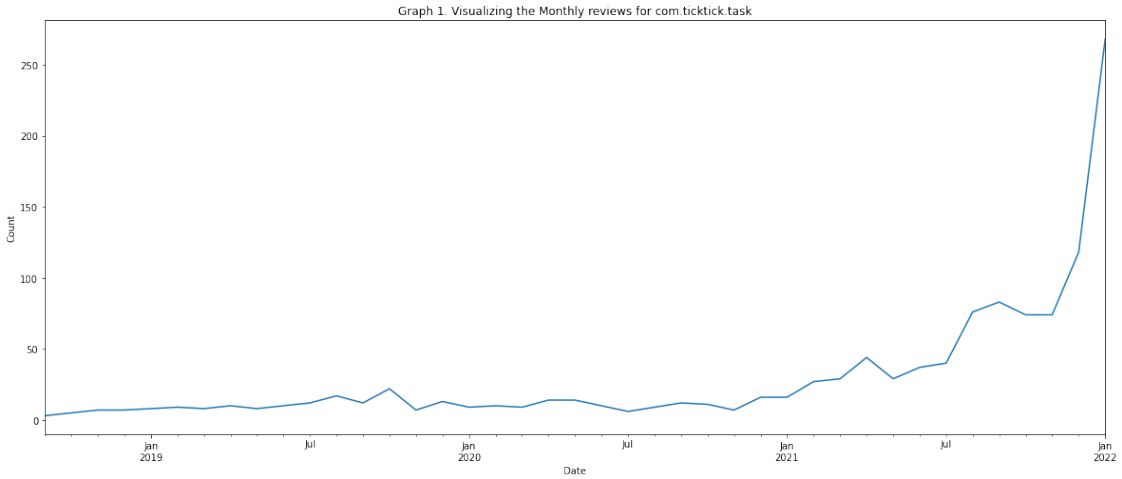
```
  # first we create a filtered dataframe
  NEW_DF = DATAFRAME[DATAFRAME['appId'] == APP_ID].copy()
  # resample into the desired type
  # title text
  d = {
       'D':'Daily',
       'W':'Weekly',
       'M':'Monthly',
       'A':'Yearly',
       'Y':'Yearly'
       }
  title_text = f'Graph 1. Visualizing the {d[SAMPLING_TYPE]} reviews for␣
↪{APP_ID}'
  NEW_DF.resample(SAMPLING_TYPE,on='at')['reviewId'].count().plot(
                                                                  ␣
↪title=title_text,
                                                                    xlabel='Date',
                                                                    ylabel='Count'
                                                                    )

  plt.show()
# visualizing it
visualize_reviews(df, 'M', sample)
```

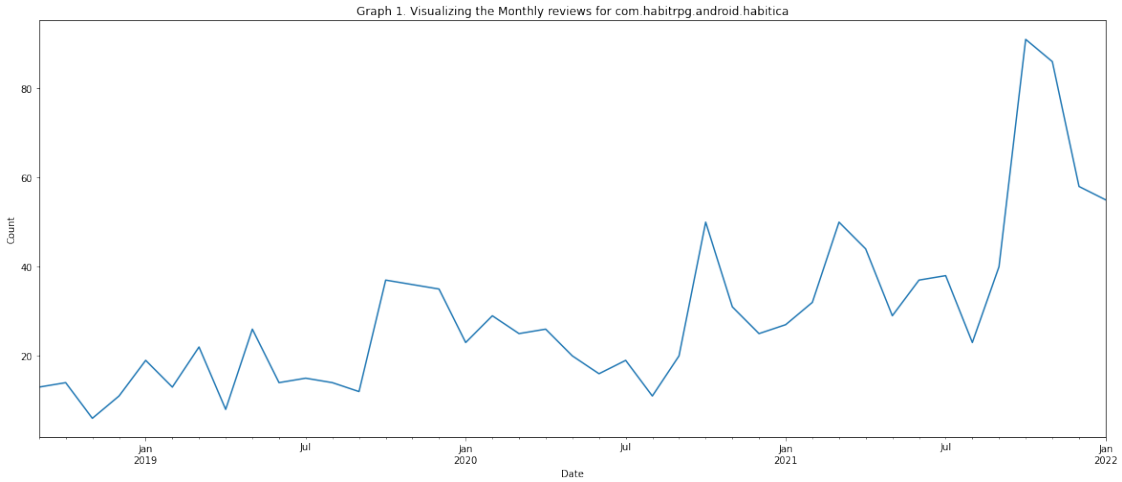


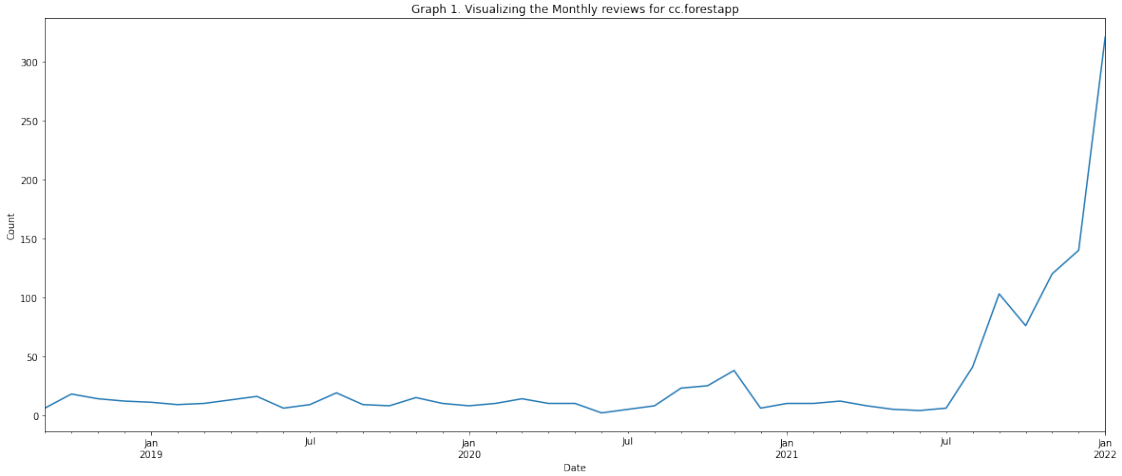
[ ]:
```
# looking at the graph for all the apps
[visualize_reviews(df,'M',x) for x in app_packages]
```

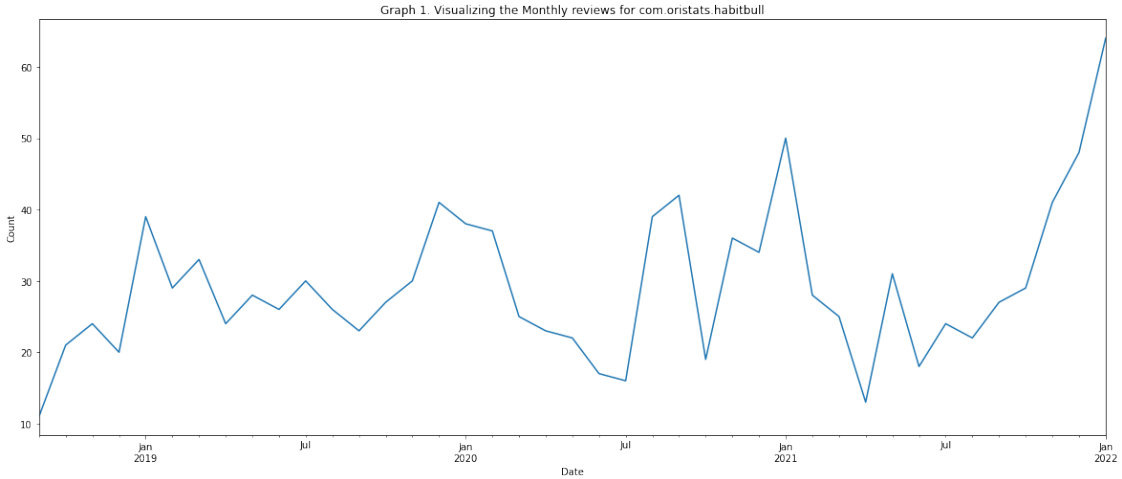

Graph 1. Visualizing the Monthly reviews for com.anydo
Count
Date
0
50
100
150
200
250
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

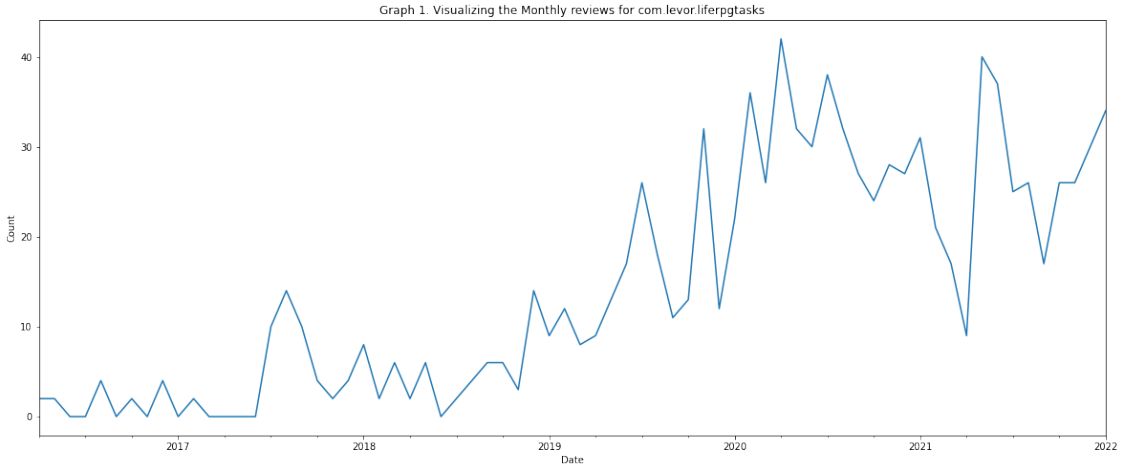

Graph 1. Visualizing the Monthly reviews for com.todoist
Count
Date
0
20
40
60
80
100
120
140
160
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

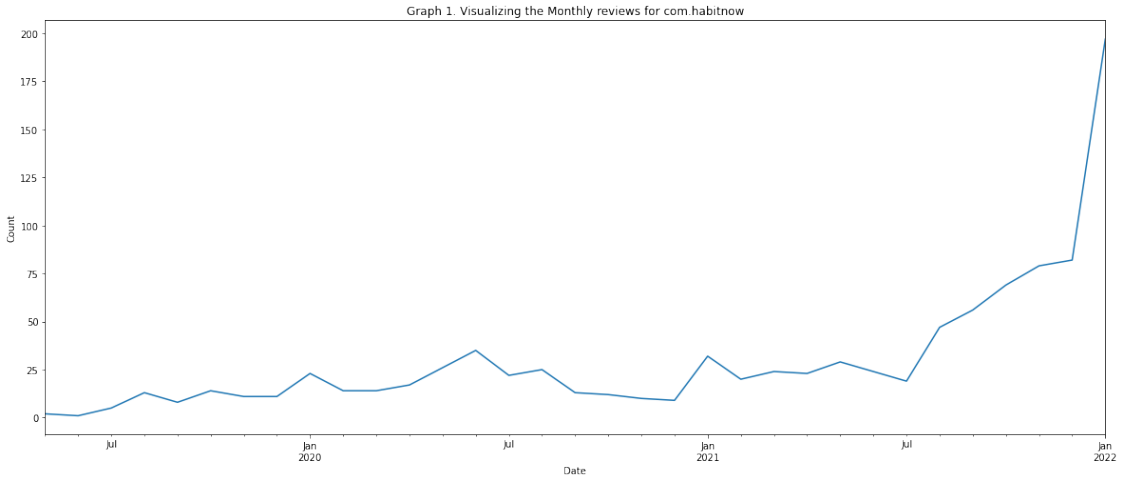

Graph 1. Visualizing the Monthly reviews for com.ticktick.task
Count
Date
0
50
100
150
200
250
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

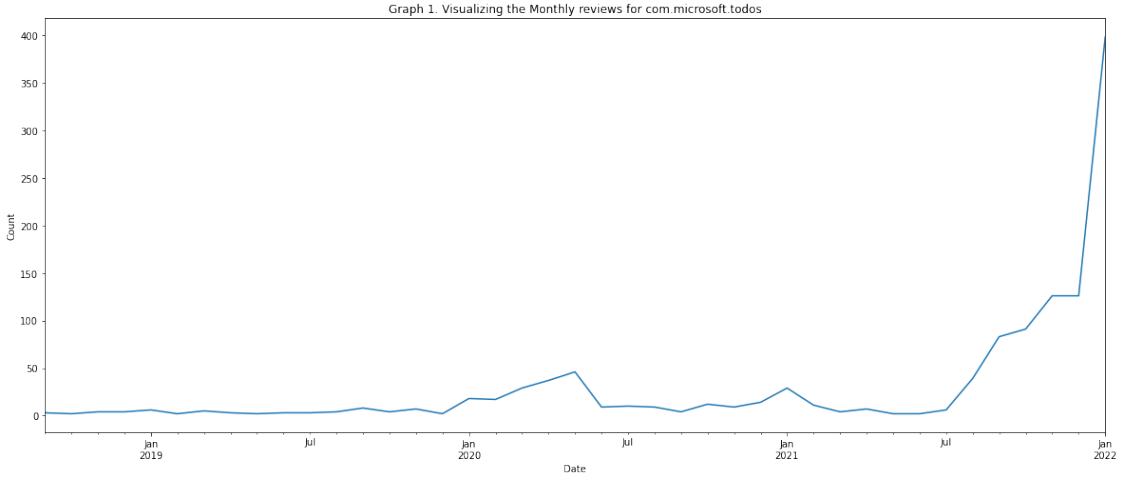

Graph 1. Visualizing the Monthly reviews for com.habitrpg.android.habitica
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

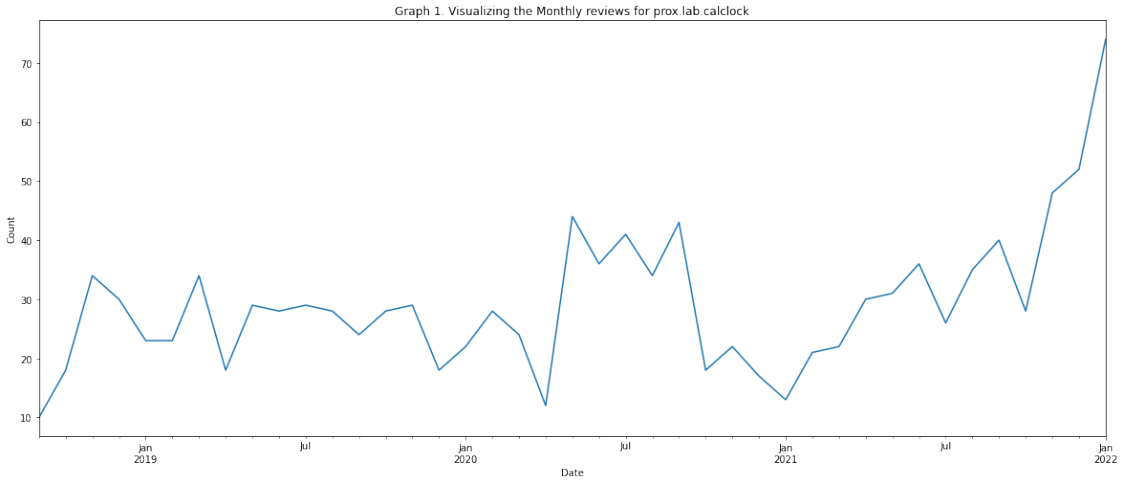

Graph 1. Visualizing the Monthly reviews for cc.forestapp
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

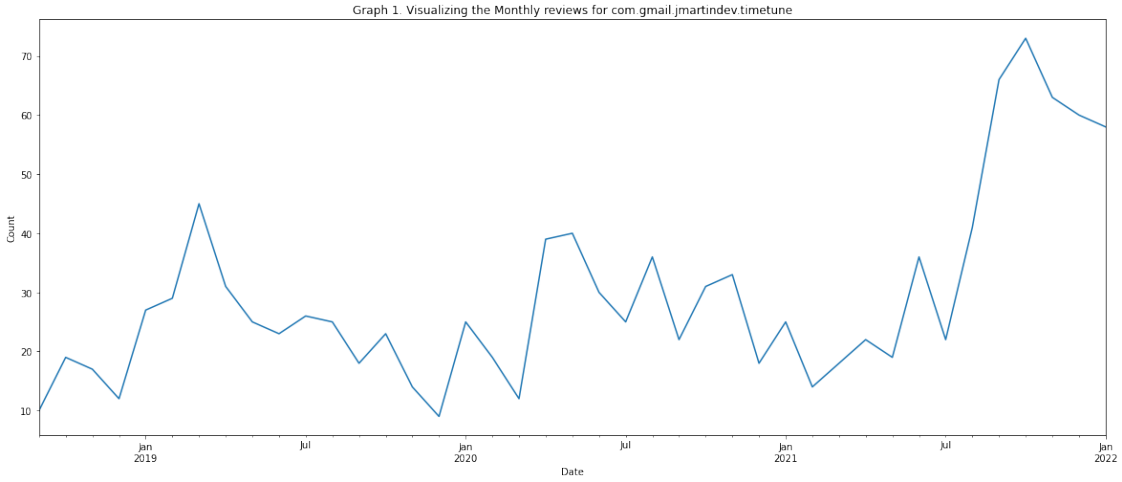

Graph 1. Visualizing the Monthly reviews for com.oristats.habitbull
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

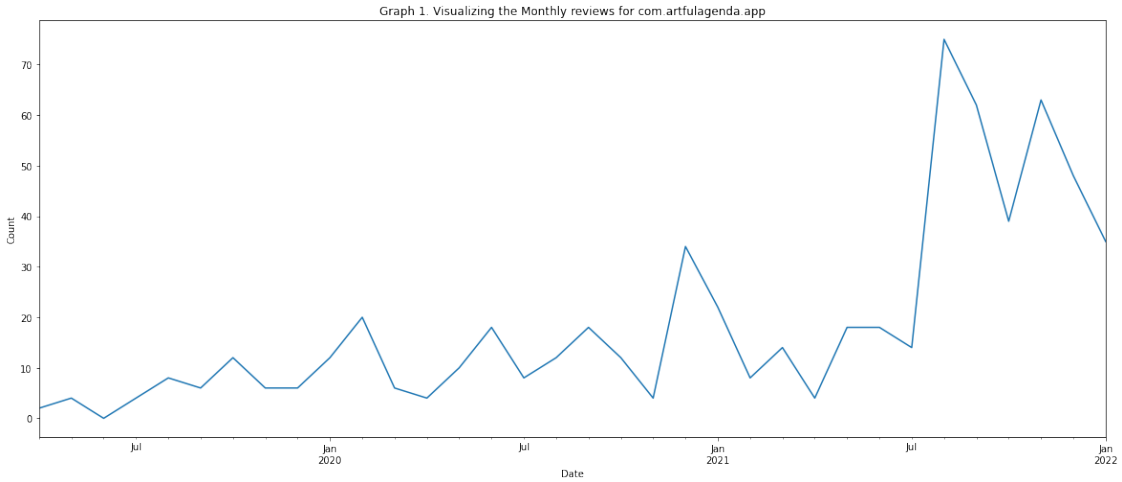

Graph 1. Visualizing the Monthly reviews for com.levor.liferpgtasks
Count
Date
0
10
20
30
40
2017
2018
2019
2020
2021
2022

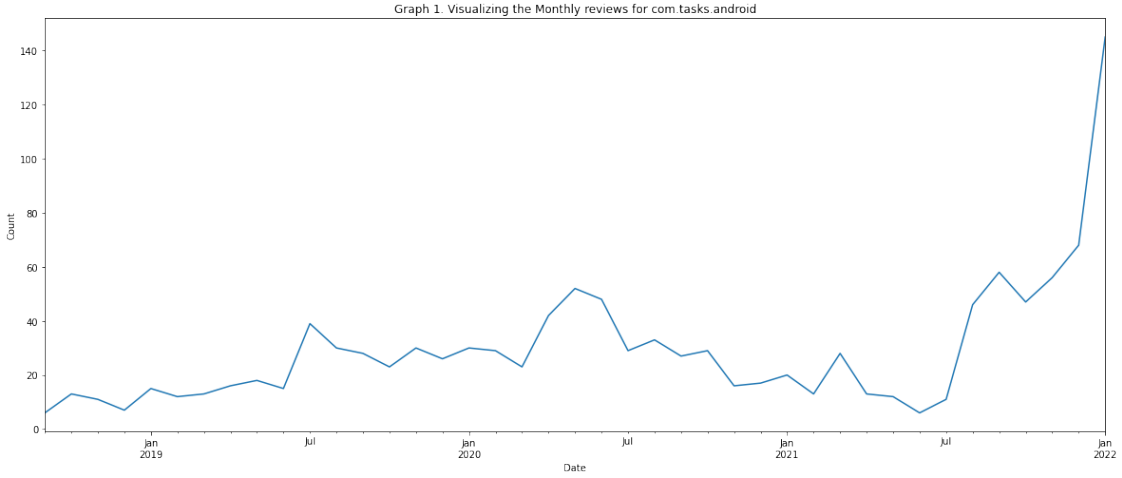

Graph 1. Visualizing the Monthly reviews for com.habitnow
Count
Date
0
25
50
75
100
125
150
175
200
Jul
Jan
2020
Jul
Jan
2021
Jul
Jan
2022

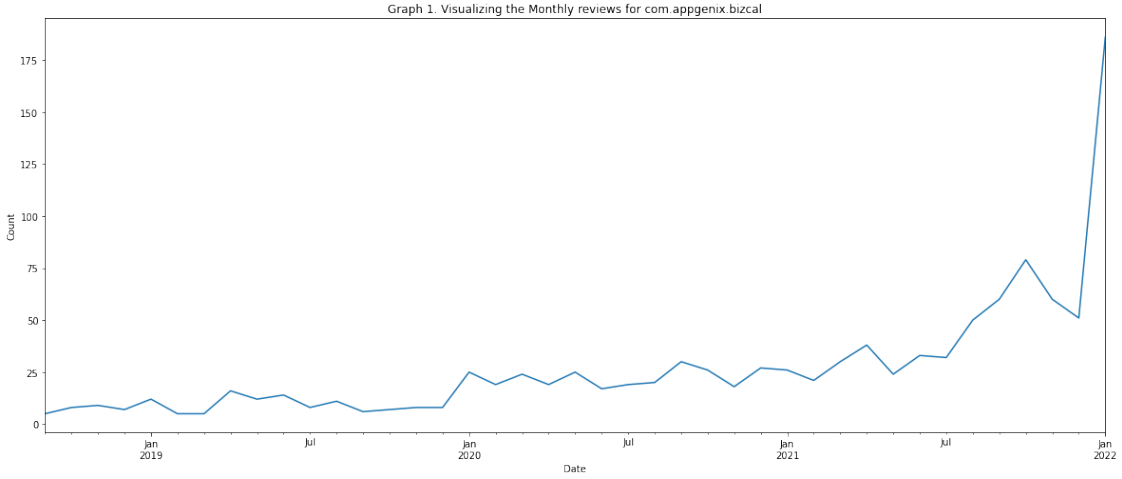

Graph 1. Visualizing the Monthly reviews for com.microsoft.todos
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022

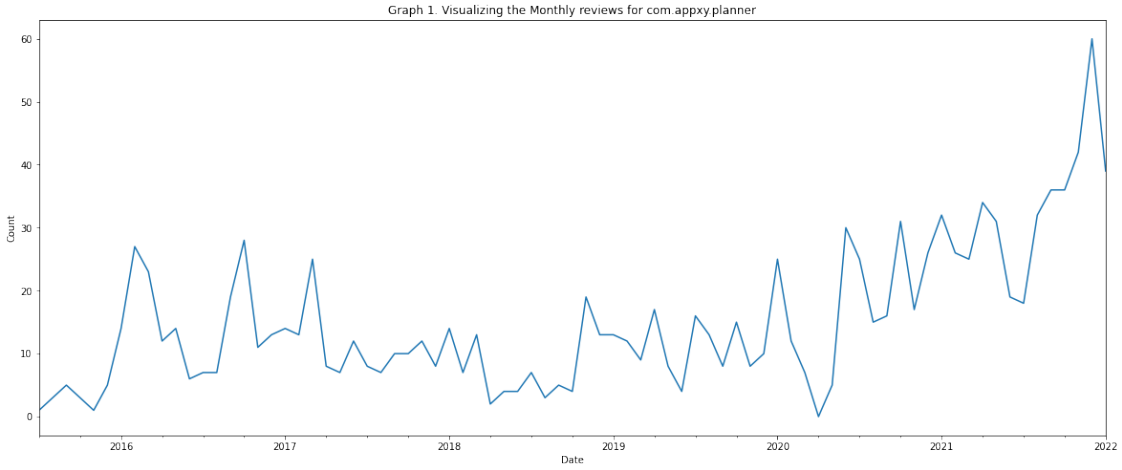

Graph 1. Visualizing the Monthly reviews for prox.lab.calclock
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022


Graph 1. Visualizing the Monthly reviews for com.gmail.jmartindev.timetune
Count
Date
Jan 2019
Jul
Jan 2020
Jul
Jan 2021
Jul
Jan 2022


Graph 1. Visualizing the Monthly reviews for com.artfulagenda.app
Count
Date
Jul
Jan
2020
Jul
Jan
2021
Jul
Jan
2022


Graph 1. Visualizing the Monthly reviews for com.tasks.android
Count
Date
Jan
2019
Jul
Jan
2020
Jul
Jan
2021
Jul
Jan
2022


Graph 1. Visualizing the Monthly reviews for com.appgenix.bizcal
Count
Date
Jan
2019
Jul
Jan
2020
Jul
Jan
2021
Jul
Jan
2022



[ ]:
```
[None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None]
```

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

[ ]:
```
16914
```

[ ]:
```

```