

▼ Créer un ensemble de données de vidéos YouTube

Problématique :

pouvez-vous écrire un programme pour extraire des informations sur les vidéos d'une page de chaîne YouTube ? Utilisez-le pour créer un ensemble de données des meilleures vidéos des chaînes populaires.



Auteurs du projet :

- Mohamed Diouf
- Eya Benalaya

Objectifs du projet :

- Apprendre à utiliser les différentes fonctionnalités de Youtube Data API
- Ecrire les fonctions nécessaires qui permettent de faire le scraping du contenu d'une chaîne youtube et d'en tirer les données nécessaires
- Stocker les données obtenues dans un format adéquat
- Faire une représentation graphique des données obtenus

Résultats attendu :

Dans le cadre de ce projet, on se propose d'étudier :

- Les vidéos ayant le plus de nombre de vues
- Les vidéo ayant le plus de "Like" et de "Dislike"
- L'évolution du nombre de réactions par vidéo
- Nombre de tags par vidéo
- [On peut rajouter d'autres graphes]

Important :

Avant de procéder à la réalisation du projet, il est important d'accéder à [la console développeur](#) et créer un nouveau projet afin de pouvoir obtenir une clé d'utilisateur qui va nous permettre de nous connecter au Youtube Data API/

▼ Implémentation du code de l'application

▼ Importer les nécessaires librairies

```
from googleapiclient.discovery import build
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

▼ Configuration de l'API

Pour le reste de notre projet nous allons utiliser chaîne youtube de [Jovian](#).

NB : Pour obtenir l'identifiant de la chaîne, il suffit de chercher la valeur attribuée à 'browse_id' dans le code source de la page d'accueil de la chaîne en question.

```
Id_Chaine = "UCmKaoNn00vxVAe7f_8sXYNQ"
Cle_API = "AIzaSyBAJdMyEKesVp54i2n5xS6WEMOzsD9I8dY"
youtube = build('youtube', 'v3', developerKey=Cle_API)
```

▼ Fonctions

Dans cette section, nous allons implémenter les différentes fonctions qui nous permettront de scraper la chaîne youtube tout en utilisant le 'Request' de l'API de Youtube

```
def statistiquesChaine(youtube, Id_Chaine):
    """
```

```

Objectifs : Cette fonction permet d'obtenir les statistiques d'une chaine youtube dont l
Méthode : Intéraction avec l'API de youtube afin d'obtenir les données demandées sous fo
Entrées : youtube , Id_Chaine
Sorties : reponse['items']
Hypothèses :-
"""
demande = youtube.channels().list(
    part="snippet,contentDetails,statistics",
    id=Id_Chaine
)
reponse = demande.execute()

return reponse['items']

```

```

def listeVideo(youtube, Id_upload):
    """
    Objectifs : Cette fonction permet d'obtenir une liste de vidéos à partir d'une playlist
    Méthode : Intéraction avec l'API de youtube afin d'obtenir les données demandées sous fo
    Entrées : youtube , Id_upload
    Sorties : liste_video
    Hypothèses : prochaine_page = True

    """
    liste_video = []
    demande = youtube.playlistItems().list(
        part="snippet,contentDetails",
        playlistId=Id_upload,
        maxResults=50 #parce que la page de résultats contient un maximum de 50 vidéo
    )
    #Nous allons parcourir (s'il existe) les prochaines pages jusqu'à l'obtention des donnée
    prochaine_page = True
    while prochaine_page:
        reponse = demande.execute()
        data = reponse['items']

        for video in data:
            video_id = video['contentDetails']['videoId']
            if video_id not in liste_video:
                liste_video.append(video_id)

        # Est ce qu'on a d'autres pages ?
        if 'nextPageToken' in reponse.keys():
            demande = youtube.playlistItems().list(
                part="snippet,contentDetails",
                playlistId=Id_upload,
                pageToken=reponse['nextPageToken'],
                maxResults=50
            )
        else:
            prochaine_page = False

    return liste_video

```

```

def detailsVideos(youtube, liste_video):
    """
    Objectifs : Cette fonction permet d'obtenir les détails d'une liste de vidéos passée en
    Méthode : Interaction avec l'API de youtube afin d'obtenir les données demandées sous fo
    Entrées : youtube , liste_video
    Sorties : liste_stats
    Hypothèses : prochaine_page = True

    """
    liste_stats=[]

    # On peut avoir uniquement 50 vidéos à la fois
    for i in range(0, len(liste_video), 50):
        demande = youtube.videos().list(
            part="snippet,contentDetails,statistics",
            id=liste_video[i:i+50]
        )

        data = demande.execute()
        for video in data['items']:
            titre=video['snippet']['title']
            publication=video['snippet']['publishedAt']
            description=video['snippet']['description']
            nbr_tag= len(video['snippet'].get('tags',[]))
            nbr_vue=video['statistics'].get('viewCount',0)
            nbr_like=video['statistics'].get('likeCount',0)
            nbr_dislike=video['statistics'].get('dislikeCount',0)
            nbr_comment=video['statistics'].get('commentCount',0)
            stats_dict=dict(titre=titre,
                            description=description,
                            publication=publication,
                            nbr_tag=nbr_tag,
                            nbr_vue=nbr_vue,
                            nbr_like=nbr_like,
                            nbr_dislike=nbr_dislike,
                            nbr_comment=nbr_comment)
            liste_stats.append(stats_dict)

    return liste_stats

```

▼ Récupération des données

Dans cette section, nous allons executer les fonctions implémentée ci-dessus afin de permettre le scraping des données.

Tout d'abord, nous allons consulter les statistiques de notre chaine avec la fonction "StatistiquesChaine"

```

stats = statistiquesChaine(youtube, Id_Chaine)
stats

```

```
[{'contentDetails': {'relatedPlaylists': {'likes': '',
    'uploads': 'UUmKaoNn0OvxVAe7f_8sXYNQ'}}},
  'etag': 'hAgMKAdJNSSX8L3y72r1ZTqImVs',
  'id': 'UCmKaoNn0OvxVAe7f_8sXYNQ',
  'kind': 'youtube#channel',
  'snippet': {'country': 'IN',
    'customUrl': 'jovianml',
    'description': 'Your cloud platform for hands-on learning.\n\n- Watch live hands-o
    'localized': {'description': 'Your cloud platform for hands-on learning.\n\n Watc
    'title': 'Jovian'},
    'publishedAt': '2018-10-12T09:58:11Z',
    'thumbnails': {'default': {'height': 88,
      'url': 'https://yt3.ggpht.com/ytc/AKedOLRodFBgvfPncrxEt2ZuILkk1wSwS6wvuXMIv1ix3g',
      'width': 88},
      'high': {'height': 800,
        'url': 'https://yt3.ggpht.com/ytc/AKedOLRodFBgvfPncrxEt2ZuILkk1wSwS6wvuXMIv1ix3g',
        'width': 800},
        'medium': {'height': 240,
          'url': 'https://yt3.ggpht.com/ytc/AKedOLRodFBgvfPncrxEt2ZuILkk1wSwS6wvuXMIv1ix3g',
          'width': 240}},
          'title': 'Jovian'},
    'statistics': {'hiddenSubscriberCount': False,
      'subscriberCount': '19700',
      'videoCount': '142',
      'viewCount': '573319'}}
```

```
type(stats)
```

list

Nous avons obtenu une liste de dictionnaires contenant plusieurs types de données à propos de la chaîne Jovian. Nous pouvons donc accéder aux informations qu'on veut tirer.

```
stats[0]['statistics']
```

```
{'hiddenSubscriberCount': False,
 'subscriberCount': '19700',
 'videoCount': '142',
 'viewCount': '573319'}
```

On a obtenu donc le nombre d'abonnées, le nombre de vus et le nombre de vidéo qui est égal à **142**

Essayons maintenant d'obtenir une liste des vidéos. Pour se faire, il faut d'abord accéder aux information de la chaine pour obtenir l'identificateur de la playlist contenant toutes les vidéos de la chaine

```
id_upload = stats[0]['contentDetails']['relatedPlaylists']['uploads']
id_upload
```

```
'lUllmKa0Nn00vxVAe7f 8sXYNO'
```

```
liste_video = listeVideo(youtube,id_upload)
liste_video
```

```
['tjUL1a61EQQ',
 'aMpVzUg3Ep0',
 'UEKNMFKxK0M',
 'IHfYF0ZqcSCA',
 'd6xH6k7_Zv4',
 'kqj8J6CVrfrw',
 'sjIzfC4A0I0',
 'Bk_Z1ATw8k0',
 'CVszSgTWODE',
 '06BrfFb6Pkc',
 'Hs18wnVqViY',
 'pS-FZe5Hd3Q',
 'F4wyQSN1_1M',
 'CVqsdTTKfd8',
 'RlGelsx7U28',
 'Y-YMLJ5FkKg',
 'NK6UYg3-Bxs',
 'ebR7x6t1xBk',
 'aaKg7WaRMyw',
 'KZf2mRNGA4w',
 'iZ83gWUZNXA',
 'RKsLLG-bzEY',
 'ZO-YwkVk8Vo',
 'tJM6-atFYM0',
 'ahMRFwphi3s',
 'lEtgfIfRhsU',
 'h6XRPmSBEM4',
 'mMQiucAukQI',
 'SmOrBW22R2k',
 'avSKR73MqBE',
 'kLDTbavcmd0',
 'bCPsBxEyQgc',
 'c6AY15bONRI',
 '9gEvswBA2d4',
 '4zcScm6Sp5U',
 '6xX45qZ3mwY',
 'nZ01Pj0KTW4',
 'M6NJUfT14aY',
 'W0R6m1sX1sY',
 'HRhGDc6Qe9k',
 '9Dpk_mYsqJc',
 '45njMELBwWI',
 'Jh4t9o2y_pw',
 'c1TW41ydwOU',
 'DyUQT06S1Zk',
 'am_hzAG0dWI',
 '2eeGWpQi9ME',
 '2vtD9Yt5hzM',
 '_vDP95xeEzo',
 'MQGH13E8QA0']
```

```
len(liste_video)
```

```
50
```

Problème : ici on ai supposé trouver une liste de 142 vidéo (fonction **listeVideo** à revoir)

Maintenant, obtenon les détails des vidéo de notre chaine youtube, afin de pouvoir les analyser.




```
donnees = detailsVideos(youtube, liste_video)
donnees
```

```
[{'description': 'We are excited to bring this AMA with Aditya Prasad, Lead Data
  'nbr_comment': '18',
  'nbr_dislike': '0',
  'nbr_like': '101',
  'nbr_tag': 22,
  'nbr_vue': '3534',
  'publication': '2021-07-29T15:30:13Z',
  'titre': 'AMA with Aditya Prasad | Lead Data Scientist at Dream11'},
{'description': '📺 Lecture 6 of the Machine Learning with Python: Zero to GBMs
  'nbr_comment': '20',
  'nbr_dislike': '2',
  'nbr_like': '96',
  'nbr_tag': 23,
  'nbr_vue': '3992',
  'publication': '2021-07-24T15:30:13Z',
  'titre': 'Unsupervised Learning and Recommendations (6/6) | Machine Learning wi
{'description': '📺 Lecture 5 of the Machine Learning with Python: Zero to GBMs
  'nbr_comment': '12',
  'nbr_dislike': '2',
  'nbr_like': '161',
  'nbr_tag': 26,
  'nbr_vue': '5643',
  'publication': '2021-07-17T15:30:11Z',
  'titre': 'Gradient Boosting with XGBoost (5/6) | Machine Learning with Python:
{'description': '📺 Lecture 4 of the Machine Learning with Python: Zero to GBMs
  'nbr_comment': '19',
  'nbr_dislike': '1',
  'nbr_like': '136',
  'nbr_tag': 27,
  'nbr_vue': '4774',
  'publication': '2021-07-10T15:30:10Z',
  'titre': 'Random Forests and Regularization (4/6) | Machine Learning with Pytho
{'description': '📺 In this lesson, we learn how to use decision trees and hyper
  'nbr_comment': '20',
  'nbr_dislike': '2',
  'nbr_like': '174',
  'nbr_tag': 25,
  'nbr_vue': '6945',
  'publication': '2021-07-03T15:30:11Z',
  'titre': 'Decision Trees and Hyperparameters | Solving a real-world problem fro
{'description': 'Join us for an introductory session on "MLOps + Feature Store: I
  'nbr_comment': '0',
  'nbr_dislike': '0',
  'nbr_like': '90',
  'nbr_tag': 30,
  'nbr_vue': '2600',
  'publication': '2021-07-01T17:17:37Z',
  'titre': 'MLOps + Feature Store: Move your models from development to productio
{'description': '📺 In this lesson we will learn about using Logistic Regression
  'nbr_comment': '29',
```

```
'nbr_dislike': '1',
'nbr_like': '294',
'nbr_tag': 22,
'nbr_vue': '12553',
'publication': '2021-06-26T15:30:12Z',
'titre': 'Logistic Regression for Classification | Working with a real-world da
{'description': "Assignment 1 of the Machine Learning with Python: Zero to GBMs
```

Maintenant que nous avons une liste de dictionnaires contenant les details des videos de notre chaine, la dernière étapes serait de les mettre dans un pandas dataframe puis dans un fichier CSV afin de pouvoir les visualiser.

```
df=pd.DataFrame(donnees)
df["nbr_vue"] = pd.to_numeric(df["nbr_vue"])
df["nbr_like"] = pd.to_numeric(df["nbr_like"])
df["nbr_dislike"] = pd.to_numeric(df["nbr_dislike"])
df["nbr_dislike"] = pd.to_numeric(df["nbr_dislike"])
df["nbr_comment"] = pd.to_numeric(df["nbr_comment"])
# les réactions sont la somme du nombre de like, dislike et commentaires
df["reactions"] = df["nbr_like"] + df["nbr_dislike"] + df["nbr_comment"]
df.to_csv("Donnees.csv")
df.head()
```

	titre	description	publication	nbr_tag	nbr_vue	nbr_lik
0	AMA with Aditya Prasad Lead Data Scientist a...	We are excited to bring this AMA with Aditya P...	2021-07-29T15:30:13Z	22	3534	10
1	Unsupervised Learning and Recommendations (6/6...	 Lecture 6 of the Machine Learning with Pytho...	2021-07-24T15:30:13Z	23	3992	9
2	Gradient Boosting with XGBoost (5/6) Machine...	 Lecture 5 of the Machine Learning with Pytho...	2021-07-17T15:30:11Z	26	5643	16
3	Random Forests and Regularization (4/6) Mach...	 Lecture 4 of the Machine Learning with Pytho...	2021-07-10T15:30:10Z	27	4774	13

#fonction permettant d'enregistrer les details des videos dans un fichier CSV

```
def ecrire_csv(items,chemain):
    with open(chemain,"w",encoding="utf-8") as f:
        if len(items)==0:
            return 0
        #recuperer les entetes
        entetes=list(items[0].keys())
        f.write(';'.join(entetes)+"\n")
        for item in items:
            valeur=[]
            for entete in entetes:
                valeur.append(str(item.get(entete, "")))
            f.write(';'.join(valeur)+"\n")
```



```
donnees_d=donnees
```

```
donnees_d[1]['description'][:50]
```

```
'📺 Lecture 6 of the Machine Learning with Python: Z'
```

```
#selectionner uniquement les 50 premier caractères de la description
i=0
for data in donnees_d:
    data['description']=data['description'][:50]+"..."
    i+=1
```

```
ecrire_csv(donnees_d,'donnee_videos.csv')
```

```
data_jovian=pd.read_csv('donnee_videos.csv',sep=";")
```

```
data_jovian.head()
```

	titre	description	publication	nbr_tag	nbr_vue	nbr_like	nbr_dislik
0	AMA with Aditya Prasad Lead Data Scientist a...	We are excited to bring this AMA with Aditya P...	2021-07-29T15:30:13Z	22	3534	101	
1	Unsupervised Learning and Recommendations (6/6...	📺 Lecture 6 of the Machine Learning with Pytho...	2021-07-24T15:30:13Z	23	3992	96	
2	Gradient Boosting with XGBoost	📺 Lecture 5 of the Machine	2021-07-17T15:30:13Z	26	5643	161	

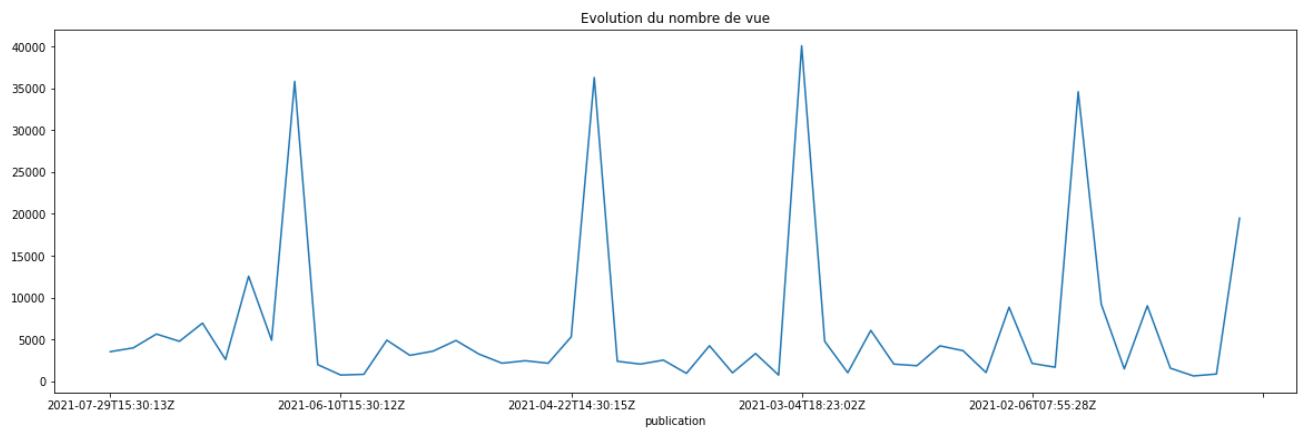
```
data_jovian1 =data_jovian.set_index('publication')
```

```
jovian_vue =data_jovian1['nbr_vue']
```

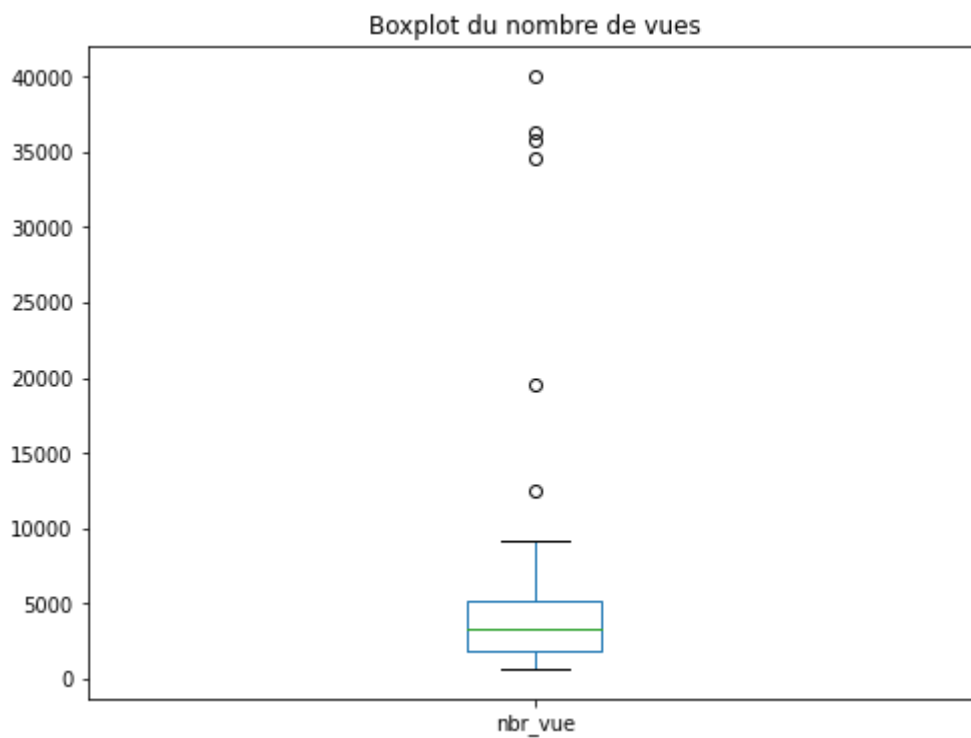
```
jovian_vue.sort_index(ascending=False,inplace=True)
```

```
jovian_vue.plot(kind='line', figsize=(20, 6))
plt.title ('Evolution du nombre de vue')
```

```
plt.show()
```



```
jovian_vue.plot(kind = 'box',figsize=(8, 6))  
plt.title('Boxplot du nombre de vues')  
plt.show()
```



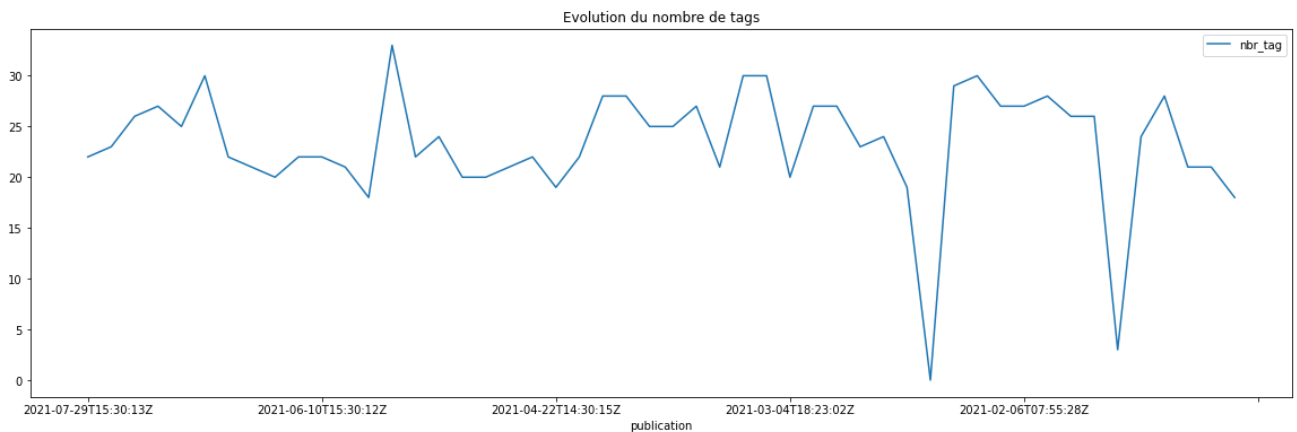
```
jovian_vue.hist(figsize=(15,6))  
plt.title('Histogramme des vues')
```

```
Text(0.5, 1.0, 'Histogramme des vues')
```

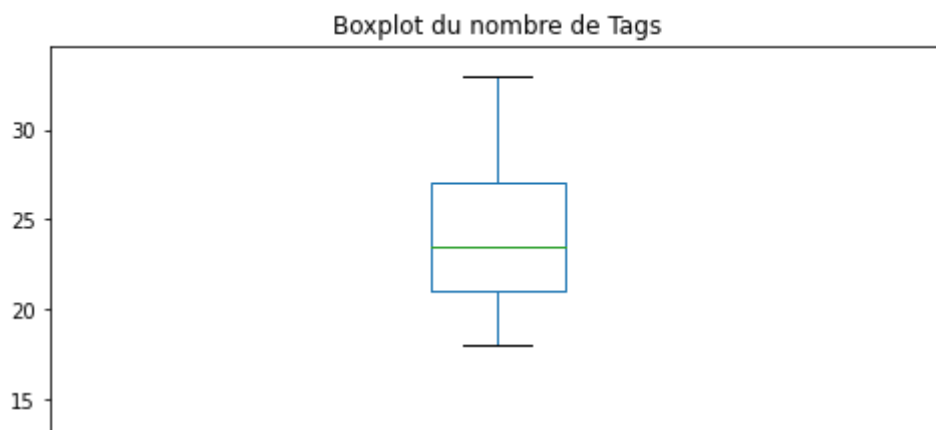


```
jovian_tags=data_jovian1.loc[:,['nbr_tag']]
jovian_tags.plot(kind='line', figsize=(20, 6))
plt.title ('Evolution du nombre de tags')
```

```
plt.show()
```

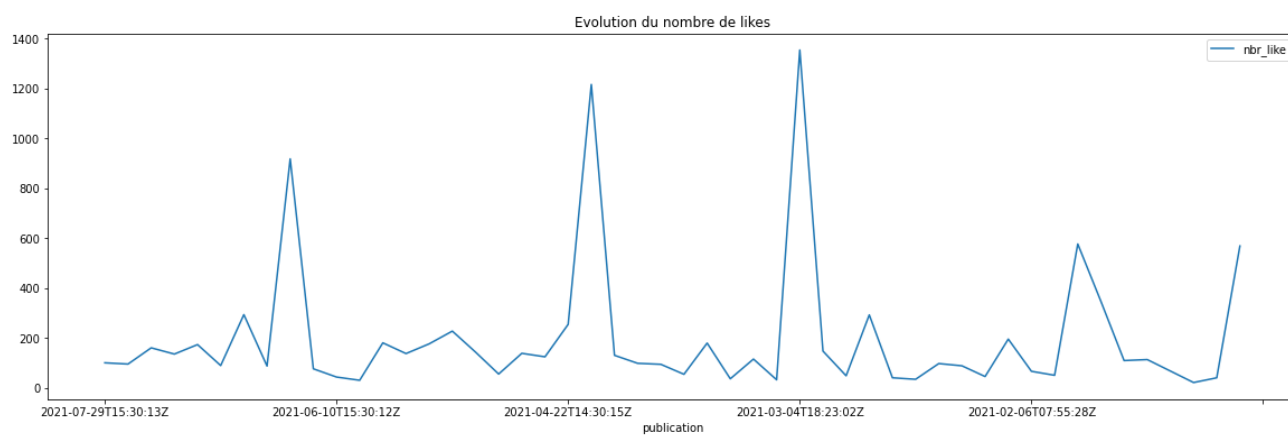


```
jovian_tags.plot(kind = 'box',figsize=(8, 6))
plt.title('Boxplot du nombre de Tags')
plt.show()
```



```
jovian_likes=data_jovian1.loc[:,['nbr_like']]
jovian_likes.plot(kind='line', figsize=(20, 6))
plt.title ('Evolution du nombre de likes')
```

```
plt.show()
```



```
jovian_likes.plot(kind='box', figsize=(8, 6))
plt.title ('box du nombre de likes')
```

```
plt.show()
```



```
jovian_likes.hist(figsize=(15,6))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7faabeb34150>]],  
      dtype=object)
```

