Anonymous_youtube

Unknown Author

January 22, 2014

1 Exploration of the Anonymous videos on Youtube.

1.1 How the initial data was generated - Davide's dataset – generate in July 2013

Here's what Davide wrote in his email:

> The query: "anonymous internet freedom"
>
> Some notes: I have just discovered that some changes have been made to YT api, so the script was not able to retreive keywords associated with videos (that can be interesting for semantic network analysis purposed) and that videos descriptions are truncated. I will try to figure out whether is possible to fix the first shortcome; for the second, luckly the "related videos" descriptions are still complete..and usually there is wide redundancy, so maybe it can be easily overcame (consider that in this sample I restricted related videos to 25 for each); the comments are also in this case limited to 25; videos are ordered by relevance computed by Youtube itself; users are limited to original content submitters; the string I input as keyword was only "anonymous internet freedom".
>
> (1) retrieve up to 1000 videos queried by keywords (including username,date,category,n of favorites, n of views,duration, tags and description)-maybe it is possible to extend the 1000 limit, or by the way it can be somehow done with the following module-
>
> (2) reconstruct the network of "related videos" for each video retrieved (with the metadata above)-the alghoritm for matching related videos is of course not known, but as far as I understood it should be both related to tags and to users' behaviour: the more two videos are viewed in sequence, the more related they'll be... so it can provide interesting insights on network of content-
>
> (3) scrape comments related to the videos/related videos retrieved (including user name, time and parent comments) The geo info are quite tricky cause there is no "nationality" associated to the video, but the language...at least not provided with the api! But maybe something can be inferred with a heuristic approach (nations mentioned in the title/description/comments)

1.2 Generating the data from Youtube API – Adrian's dataset

Davide's dataset is really useful, but I found it hard to replicate the same results. The scripts I wrote to query the YT api bring back different results. This needs to be investigated. Also because Google/Youtube only returns a maximum of 1000 results, we need to think of ways of 'tiling' the queries to get around this limitation. At least, so that we can be reasonably satisfied we are getting all the relevant videos.

Felipe: use list of campaigns to script a set of queries that build a bigger dataset

In [1]:
```
%load_ext autoreload
%autoreload 2
```

In [2]:
```
import pandas as pd
import YT_api_generate as yt
from pylab import *
import seaborn
from IPython.display import HTML
from IPython.display import YouTubeVideo
pd.set_option("display.max_columns", 6)
pd.set_option("display.max_rows", 15)
pd.set_option("display.notebook_repr_html", True)
from optparse import OptionParser
```

In [86]:
```
an_f = pd.ExcelFile('data/anonymouys internet freedom - VIDEOS.xls')
an_dict = {name:an_f.parse(name) for name in an_f.sheet_names}
video_df = an_dict['VIDEOS']
#clean up data a bit
video_df.VIEWS = video_df.VIEWS.replace('None', 0)
print(video_df.shape)
print(video_df.head(10))
```

```
(908, 11)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10 entries, 0 to 9
Data columns (total 11 columns):
ID              10  non-null values
TITLE           10  non-null values
URL             10  non-null values
USER            10  non-null values
DATA            10  non-null values
CATEGORY        10  non-null values
FAVOURITES      10  non-null values
VIEWS           10  non-null values
DURATION        10  non-null values
KEYWORDS        0  non-null values
DESCRIPTION     10  non-null values
dtypes: float64(2), object(9)
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
```

The dataset has about a 1000 rows, but many seem duplicates. That is, the rows are literally identical in every field So cutting those out: ...

In [49]:
```
video_df = video_df.drop_duplicates()
print(video_df.shape)
print('Users: ', len(video_df.USER.unique()))
print('Titles: ', len(video_df.TITLE.str.encode('utf-8').unique()))
print('IDS: ', len(video_df.ID.unique()))
```

```
(289, 11)
('Users: ', 246)
('Titles: ', 283)
('IDS: ', 289)
```

There are 908 titles in the Beraldo dataset, of which **289** are unique. There are 246 users – people (robots?) who upload videos.

Comparing Davide's results with the ones I get from the Youtube API:

In [7]:
```
df_am = yt.youtube_search(query='anonymous,internet,freedom', max_results=1000, with_s
getting another page of results ... 50
getting another page of results ... 100
getting another page of results ... 150
getting another page of results ... 200
getting another page of results ... 250
getting another page of results ... 300
getting another page of results ... 350
getting another page of results ... 400
getting another page of results ... 450
getting another page of results ... 500
getting another page of results ... 540
getting another page of results ... 590
getting another page of results ... 640
getting another page of results ... 690
getting another page of results ... 740
getting another page of results ... 790
getting another page of results ... 840
getting another page of results ... 890
getting another page of results ... 940
getting video statistics: 50
getting video statistics: 100
getting video statistics: 150
getting video statistics: 200
getting video statistics: 250
getting video statistics: 300
getting video statistics: 350
getting video statistics: 400
getting video statistics: 450
getting video statistics: 500
getting video statistics: 550
getting video statistics: 600
getting video statistics: 650
getting video statistics: 700
getting video statistics: 750
getting video statistics: 800
getting video statistics: 850
getting video statistics: 900
getting video statistics: 950
getting video statistics: 990
(990, 14)
(990, 8)
```

In [79]:
```
df_am.columns
df_am.drop_duplicates(inplace=True, cols='videoId')
print(df_am.shape)
print 'Unique ids: ', len(df_am.videoId.unique())
```

```
(525, 23)
Unique ids:  525
```

So not a huge difference in numbers – 525 vs 289. But are they the same videos more or less?

In [80]:
```
davide_set = set(video_df.ID.tolist())
adrian_set = set(df_am.videoId.tolist())
comb_id = davide_set.union(adrian_set)
intersecting_id = adrian_set.intersection(davide_set)
adrian_unique = adrian_set.difference(davide_set)
davide_unique = davide_set.difference(adrian_set)
print('Set of combined video ids: ' + str(len(comb_id)))
print('They have ' + str(len(intersecting_id)) + ' in common')
print('Davide datatset has ' + str(len(davide_unique)) + ' that Adrian doesnt')
print('Adrian has ' + str(len(adrian_unique)) + ' that Davide doesnt')
```

```
Set of combined video ids: 707
They have 107 in common
Davide datatset has 182 that Adrian doesnt
Adrian has 418 that Davide doesnt
```

So it looks like there is in **poor overlap** between them. Altogether there are around 700 unique video ids, but 25% are common to both. The rest are different. How to explain this? If Davide used the Youtube API like I did, does this mean that Google is giving different results to different people? Or that the results depend on when/where you run the query? The question of getting the same data is maybe only an interesting methodological wrinkle, or it might be something worth investigating in its own right.

I'll leave this aside for the moment, and just combine them both into one dataset.

In [91]:
```
#combine all the ids and titles from Davide and Adrian's results
video_df['videoId'] = video_df.ID
video_df['title'] = video_df.TITLE.str.encode('utf-8')
video_df['duration_second'] = video_df.DURATION


video_comb = pd.concat([df_am[['videoId', 'title', 'duration_second']], video_df[['vid
video_comb = video_comb.drop_duplicates()
video_comb.shape
```

```
(715, 3)
```

Out [91]:

2 The uniqueness of the videos

But once all the dupicates are removed, it seems we are left with a unique list of videos, and **they are not mirrored very much at all.** Is that right? Video-IDs means that there is no real mirroring? A key question, because it would undermine the whole mirroring strategy.

Here are the mirrored videos:

In [92]:
```
vc=video_comb.title.value_counts()
vc[vc>1]
```

Out [92]:
```
Freedom                                    5
Anonymous Declaration of Freedom           4
A message from Anonymous Security          3
Internet Censorship                        3
Anonymous - Operation ANTI-ACTA            3
...
Is Hollywood undermining the free Internet?                    2
```

```
ANONYMOUS NO MORE! Anonymous Comments Online to be BANNED from the
Internet    2
LulzSec & Anonymous Declare War Against Government! Operation Anti
Security '#AntiSec'! Leak Info!    2
Christopher Hitchens  Wishful Thinking vs Evidence 2008)        2
freedom                                                         2
EU & 22 Countries Sign ACTA! Anonymous Start #TwitterCensored! Twitter
Start Censorship 2012!    2
Length: 31, dtype: int64
```

That's it in combined's dataset: 5 copies of 'Anonymous Declaration of Freedom,' and 2 of the rest. **5** mirrored videos, and not highly mirrored.Here there is a bit more duplication, and some overlap. There are also some time-specific results – PRISM and Snowden probably arrives after Davide has generated his dataset. But still only 8 mirrored videos.

I guess we should discuss this Adam. Should we look for more hidden duplicates? Is it something to do with the youtube search? Are there other ways of identifying videos associated with Anonymous that need to look into.

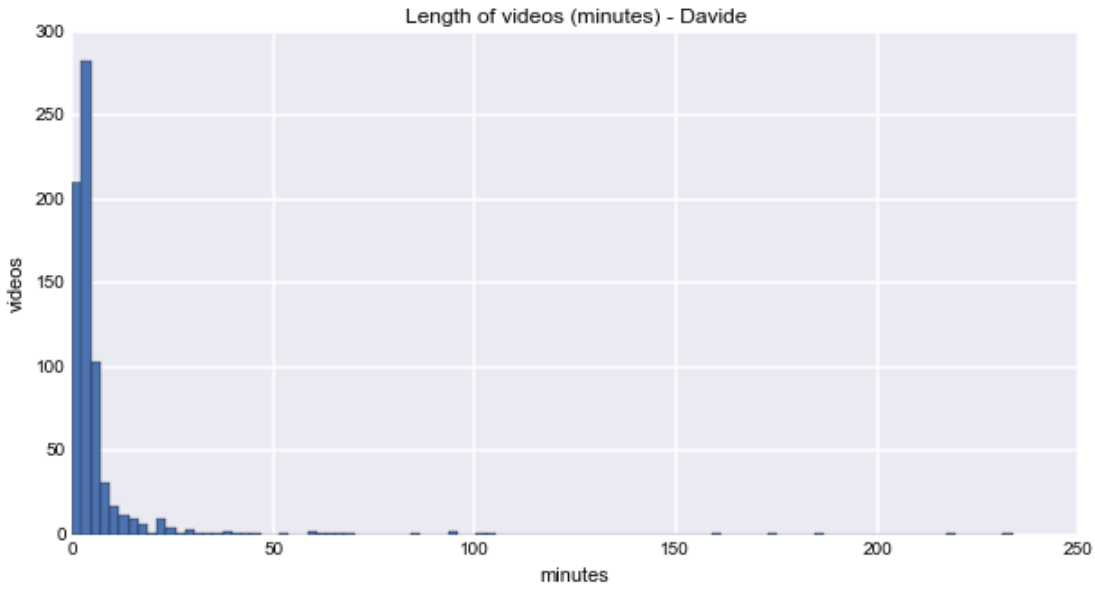
2.1 The length of the videos

Not sure what the length of videos tells us. Something about the kind of object we are dealing with. Also, I guess length affects mirroring. Would many people upload a 60 minute documentary?

In [95]:
```
# renaming dataframes
video_df = video_comb

duration = video_df.duration_second.order().tolist()
duration = [d/60 for d in duration]
f = pylab.figure(figsize=(10,5))
sp=f.add_subplot(1,1,1)

h=sp.hist(duration, bins=100)
print('Less than 10 minutes: '+ str(float(sum(video_df.duration_second/60 < 10))/video
sp.set_title('Length of videos (minutes) - Davide')
sp.set_xlabel('minutes')
sp.set_ylabel('videos')
#adrian's videos
# df_am = yt.format_durations(df_am)
# df_am.columns
# #df_am['duration_second'].hist(bins=100)
# sp2=f.add_subplot(1,2,2)
# sp2.set_title('Length of videos (minutes) - Adrian')
# sp2.set_xlabel('minutes')
# sp2.set_ylabel('videos')
# dur_min = df_am.duration_second/60
# h2 = sp2.hist(dur_min, bins=100)
YouTubeVideo(video_df.videoId.iloc[1])
```

```
Less than 10 minutes: 88.951048951%
<IPython.lib.display.YouTubeVideo at 0x6cec650>
```

Out [95]:



Again, the differences between the datasets. But can't see any pattern here.

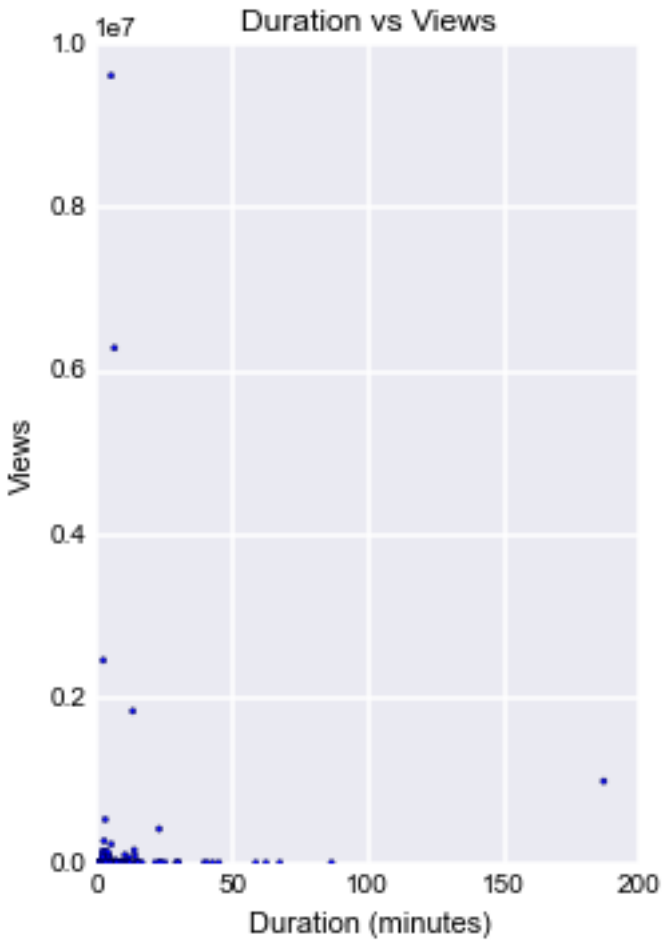
In general, it looks like most of the videos are relatively short (less than 10 minutes). Does duration correlate with either viewing or duplication?

2.2 Felipe: EVERYTHING BELOW HERE IS NOT RELEVANT AT THE MOMENT OR NEEDS FIXING

In [18]:
```
fig=figure(dpi=200)
sp = fig.add_subplot(1,2,1)
sp.scatter(video_df.DURATION/60, video_df.VIEWS, s=5,marker='o')
sp.set_title('Duration vs Views')
sp.set_xlabel('Duration (minutes)')
sp.set_ylabel('Views')
sp.set_xlim(0, 200)
sp.set_ylim(0, 1e7)

hms =[re.sub('H|M', ':', s) for s in [re.sub('PT|S', '', t)   for t in df_am.duration]

sp = fig.add_subplot(1,2,2)
sp.scatter(video_df.DURATION/60, video_df.VIEWS, s=5,marker='o')
sp.set_title('Duration vs Views')
sp.set_xlabel('Duration (minutes)')
sp.set_ylabel('Views')
sp.set_xlim(0, 200)
sp.set_ylim(0, 1e7)
```

```
---------------------------------------------------------------------
-------------
    NameError                                 Traceback (most recent
call last)



<ipython-input-18-657643217319> in <module>()
  8 sp.set_ylim(0, 1e7)
  9
```

```
    ---> 10 hms =[re.sub('H|M', ':', s) for s in [re.sub('PT|S', '',
t)   for t in df_am.duration]]
         11
         12 sp = fig.add_subplot(1,2,2)


NameError: name 're' is not defined
```



At a glance, it looks like the longer videos are viewed much less. Very short videos are not viewed much either. But this doesn't take into account the fact that there are many more short videos.

3 Viewing figures

Another way to look at videos – more in terms of reception. Does anything stand out from the viewing figures?

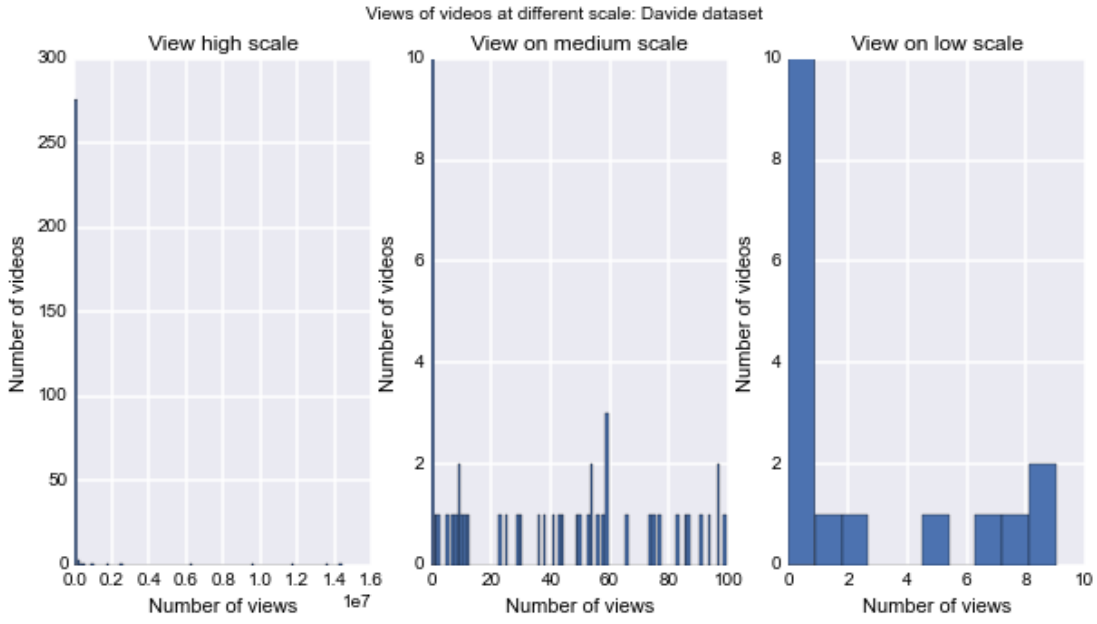
As usually, totally different results from the Adrian dataset – maximum view is 66 million, not 14 million

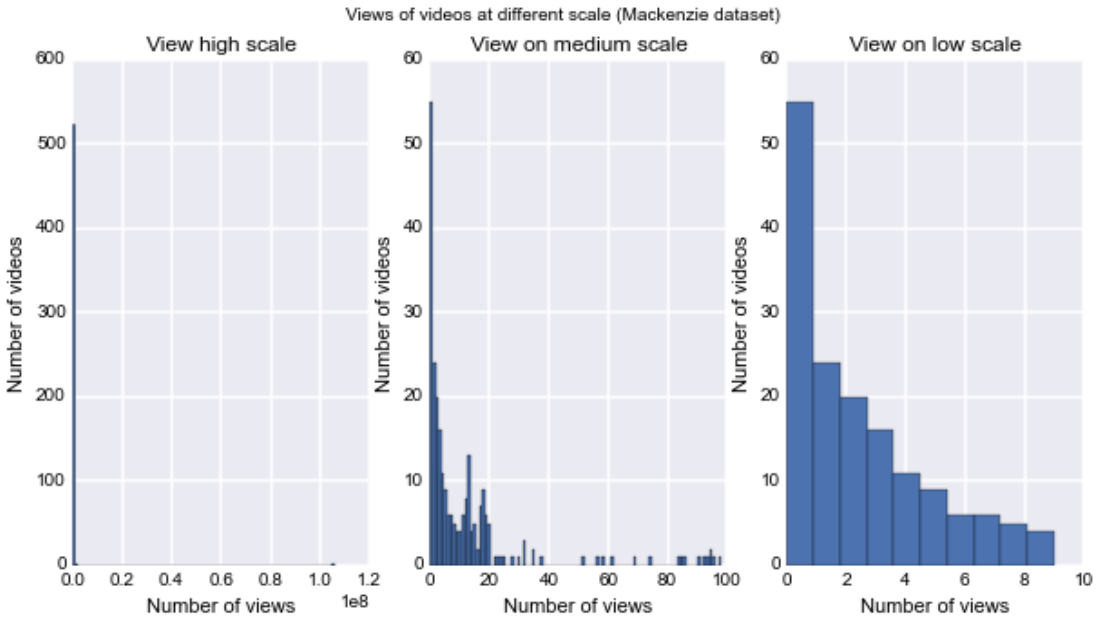
In [19]:
```
video_df.VIEWS = video_df.VIEWS.replace('None', 0)
views = video_df.VIEWS.order()
print('Maximum views: ' + str(views.max()))
```

```
f=figure(figsize(10,5))
suptitle('Views of videos at different scale: Davide dataset')
sp1 = f.add_subplot(1,3,1)
h1=sp1.hist(views, bins=100)
sp1.set_ylabel('Number of videos')
sp1.set_xlabel('Number of views')
sp1.set_title('View high scale')
sp2 = f.add_subplot(1,3,2)
h2 = sp2.hist(views[views<100], bins=100)
sp2.set_title('View on medium scale')
sp2.set_ylabel('Number of videos')
sp2.set_xlabel('Number of views')
sp3 = f.add_subplot(1,3,3)
h3 = sp3.hist(views[views<10], bins = 10)
sp3.set_title('View on low scale')
sp3.set_ylabel('Number of videos')
sp3.set_xlabel('Number of views')

#Adrian dataset
views = df_am.viewCount.astype('int').order()
print('Max views in adrian dataset: ' + str(views.max()))
f=figure(figsize=(10,5))
suptitle('Views of videos at different scale (Mackenzie dataset)')
sp1 = f.add_subplot(1,3,1)
h1=sp1.hist(views, bins=100)
sp1.set_ylabel('Number of videos')
sp1.set_xlabel('Number of views')
sp1.set_title('View high scale')
sp2 = f.add_subplot(1,3,2)
h2 = sp2.hist(views[views<100], bins=100)
sp2.set_title('View on medium scale')
sp2.set_ylabel('Number of videos')
sp2.set_xlabel('Number of views')
sp3 = f.add_subplot(1,3,3)
h3 = sp3.hist(views[views<10])
sp3.set_title('View on low scale')
sp3.set_ylabel('Number of videos')
sp3.set_xlabel('Number of views')
```

```
Maximum views: 14381492.0
Max views in adrian dataset: 105657789
<matplotlib.text.Text at 0x91325d0>
```

Out [19]:





As usual, most videos are only viewed by a small number of people, and a couple are viewed by 100,000s up to 14 million. Actually, viewing figures are quite widely distributed even for the less-viewed videos. It might be interesting to look at the videos that are viewed 8, 30 or 80 times as well as the ones viewed 14 millon times. This analysis *does not* take into account view figures for copies of videos.

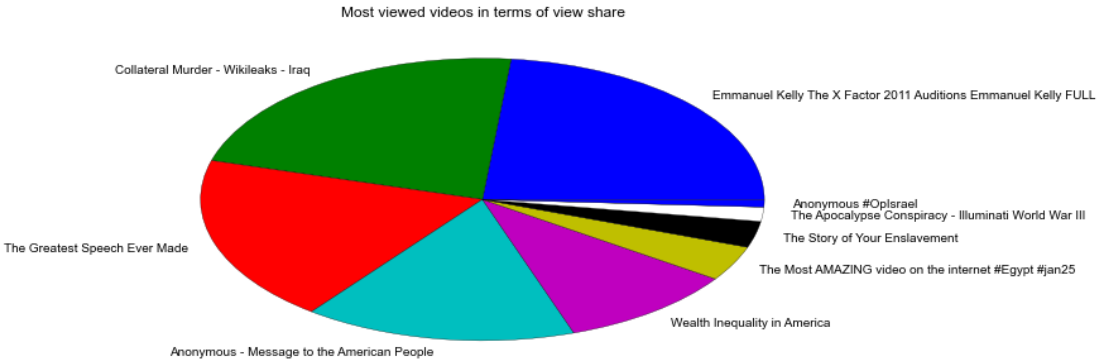
3.1 The most viewed videos

If we aggregate views for videos with the same title, things start to look different.

In [21]:
```
top_views = video_df.loc[video_df.VIEWS>100000, ['TITLE', 'VIEWS']]
gby=top_views.groupby(by='TITLE')['VIEWS']
print('total viewings:'+str(video_df['VIEWS'].sum()))
print('total views of top videos:', sum(gby.sum()))
print('total views of top 5 videos:', sum(gby.sum()[:4]))

print(gby.ngroups)
gbyo=gby.sum().order(ascending=False)
pie(gbyo[0:9], labels=gbyo.keys()[0:9])
pylab.title('Most viewed videos in terms of view share')
pd.DataFrame(gbyo).head(n=5)
```

```
total viewings:64494886.0
('total views of top videos:', 63247980.0)
('total views of top 5 videos:', 1128680.0)
18
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
```

Out [21]:
```
VIEWS
TITLE
Emmanuel Kelly The X Factor 2011 Auditions Emmanuel Kelly FULL
14381492
Collateral Murder - Wikileaks - Iraq
13640192
The Greatest Speech Ever Made
11744180
Anonymous - Message to the American People
9612949
Wealth Inequality in America
6287314
```

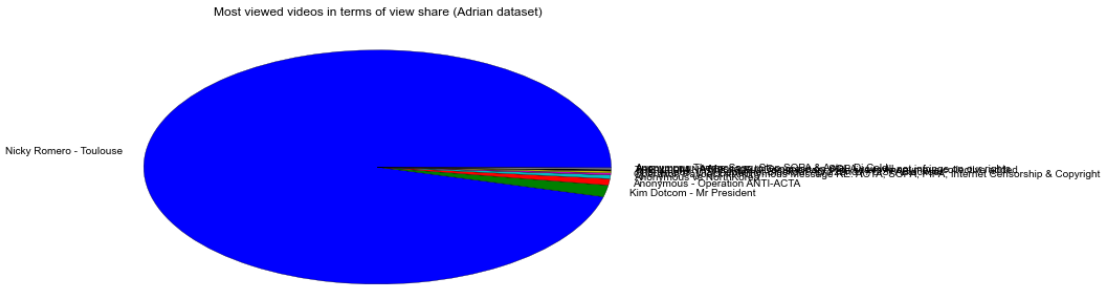


In [22]:
```
# for the Adrian dataset
df_am.viewCount = df_am.viewCount.astype('int32')
top_views_am = df_am.loc[df_am.viewCount>100000, ['title', 'viewCount']]
gby_am=top_views_am.groupby(by='title')['viewCount']
print('total viewings:'+str(df_am['viewCount'].sum()))
print('total views of top videos:', sum(gby_am.sum()))
print('total views of top 5 videos:', sum(gby_am.sum()[:4]))

print(gby_am.ngroups)
gbyo_am=gby_am.sum().order(ascending=False)
pie(gbyo_am[0:9], labels=gbyo_am.keys()[0:9])
pylab.title('Most viewed videos in terms of view share (Adrian dataset)')
pd.DataFrame(gbyo_am).head(n=5)
```

```
total viewings:112748149
('total views of top videos:', 110920802)
('total views of top 5 videos:', 1393879)
15
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
```

Out [22]:
```
                      viewCount
title
Nicky Romero - Toulouse
105657789
Kim Dotcom - Mr President
```

```
1770309
Anonymous - Operation ANTI-ACTA
973731
Anonymous vs NorthKorea
568330
Operation Payback #Anonymous Message RE: ACTA, SOPA, PIPA, Internet
Censorship & Copyright     308797
```



There are 18 videos that each have more 100,00 views. There have been around 480 million views in total of Anonymous-related videos. The top 18 videos account for 477 million of the views. The top 5 videos account for 460 million of the 480 million. Looking at the titles, is the 'Emmanuel Kelly X Factor 2011 Auditions' video 'noise'? Maybe not – he's an Iraqi orphan living in Australia, and 'Collateral Murder - Wikileaks - Iraq' (#2) doesn't look like that. The top four videos after Kelly's all have more than 50 million viewings. So they dominate the total views of Anonymous videos.

In [12]:
```
#video_df.TITLE.
emmanuel_kelly = video_df.ID[video_df.TITLE == gbyo.index[0]]
YouTubeVideo(emmanuel_kelly.iloc[0])
```

Out [12]:
```
<IPython.lib.display.YouTubeVideo at 0x4c13a90>
```

3.2 Are video duplicates viewed?

Are views related to duplication/mirroring practices? First, we need to find the duplicates. Title is an ok starting point. Presumably, videos with the same title are likely to be duplicates.

In [13]:
```
title = video_df.TITLE.tolist()
video_df.TITLE = [t.encode('utf8', errors='ignore') for t in title]
print('distinct titles:', len(video_df.TITLE.unique()))
print('distinct ids', len(video_df.ID.unique()))
print('distinct durations:', len(video_df.DURATION.unique()))
```

```
('distinct titles:', 283)
('distinct ids', 289)
('distinct durations:', 221)
```

While there are 908 videos listed, there are only 283 distinctly titled videos and 289 distinct video IDS. That means that potentially ~620 are copies. But if copies have the same ID, does that mean that they have been uploaded? Or are they simply different views on the same video? This is a crucial question - are we dealing here with duplicates or just references to one instance?

Things are a bit more complicated in the duration data. There are only 220 distinct durations. But it could be that different videos happen to have the same duration. What are the most commonly duplicated videos?

In [14]:
```
dup_video_counts =video_df.TITLE.value_counts()
dup_df = pd.DataFrame(dup_video_counts.head())
dup_df
```

Out [14]:
```
                                                                  0
Christopher Hitchens  Wishful Thinking vs Evidence 2008)   16
Anonymous is Freedom                                       15
Anonymous Declaration of Freedom                           10
Anonymous - Truth is a Virus                               10
Anonymous Exposed for Crimes Against Humanity               9
```

In [15]:
```
top_dup_video_counts=dup_video_counts[dup_video_counts>5]
top_dup_video_counts = top_dup_video_counts.order(ascending=False)
top_dup_videos = dup_video_counts.index
top_dup_video_counts
```

Out [15]:
```
Christopher Hitchens  Wishful Thinking vs Evidence 2008)    16
Anonymous is Freedom                                        15
Anonymous - Truth is a Virus                                10
Anonymous Declaration of Freedom                            10
The Truth About The Anonymous Mask.                          9
...
Tor tutorial "How to be anonymous on the internet"                6
Anonymous: S.O.P.A.                                               6
Anonymous - NDAA Bill Signed                                      6
OpIsrael Anonymous launches massive cyber assault                 6
Arranca la operación de Anonymous contra Israel para borrarlos de
Internet (06/04/13)    6
Anonymous Current Event Project                                   6
Length: 84, dtype: int64
```

Does duplication correlate with viewing figures? Do these duplicates get the same number of views? Or are some versions much more widely viewed than others?

In [16]:
```
dup_views=video_df.loc[video_df.TITLE.isin(top_dup_videos), ['TITLE', 'VIEWS']]
dup_views['copies'] = top_dup_video_counts
dv =pd.DataFrame(dup_views.groupby(['TITLE'])['VIEWS'].sum().order(ascending=False))
dv.head()
```

Out [16]:
```
VIEWS
TITLE
Emmanuel Kelly The X Factor 2011 Auditions Emmanuel Kelly FULL
115051936
Collateral Murder - Wikileaks - Iraq
109121536
The Greatest Speech Ever Made
93953440
Anonymous - Message to the American People
76903592
Wealth Inequality in America
50298512
```

Some interesting points here. High rates of duplication does not equate with high view counts. What is the Christopher Hitchens piece? It is the 4th most duplicated video, but shows 0 views - is that possible? More importantly, the most duplicated video 'Anonymous - Truth is a Virus' only has 1380 views.

3.3 Some stuff on users

In [23]:
```
video_df.columns
video_df['USER'].unique().shape
```

```
(246,)
```

Out [23]:

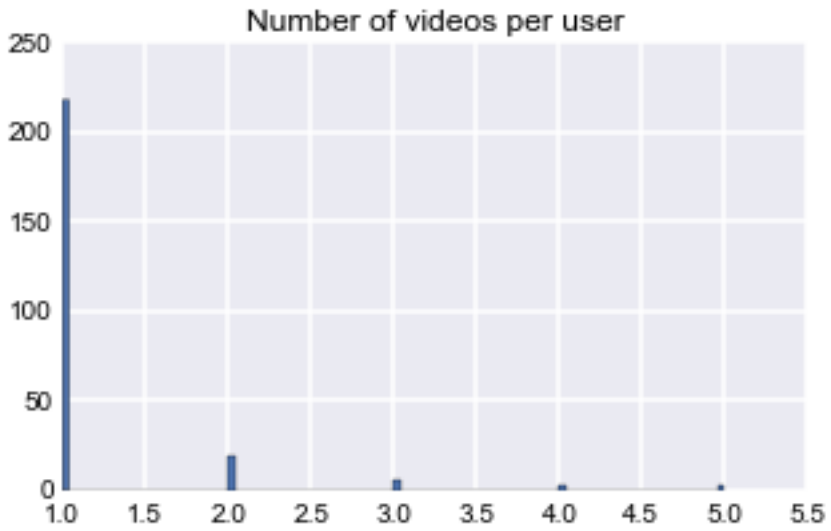
So there are 246 users (people?) who upload videos. How much do they upload?

In [24]:

```
user_counts = video_df['USER'].value_counts()
f3 = figure(figsize=(5,3))
sp=f3.add_subplot(111)
h4 = sp.hist(video_df['USER'].value_counts(), bins=100)
sp.set_title('Number of videos per user')
top_users = user_counts[user_counts>5].index
top_users
print(user_counts.describe())
user_counts.quantile(0.6)
```

```
count    246.000000
mean       1.174797
std        0.576703
min        1.000000
25%        1.000000
50%        1.000000
75%        1.000000
max        5.000000
dtype: float64
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/compat/scipy.py:68: DeprecationWarning: using a non-
integer number instead of an integer will result in an error in the
future
  score = values[idx]
1
```

Out [24]:



So, more than 60% of people upload less than 2 videos.

3.4 How top contributing users upload top-viewed videos – that is, do they duplicate them?

I'm calling 'top users' anyone who uploads more than 5 videos, and 'top videos' any video that is duplicated more than 5 times.

In [32]:
```
topuser_toptitle=pd.crosstab(video_df[video_df.USER.isin(top_users)]['USER'], video_df
top_user_title_df = pd.DataFrame(topuser_toptitle.sum(axis=1).order(ascending=False))
top_user_title_df.head()
```

Out [32]:
```
                        0
USER
YourRightsVigilante    18
Truthloader            17
wikispeak10            16
Tom Craig              16
MatriXCr3w             16
```

This suggests that there is some link between between uploading a lot and duplicating a lot. I'm not sure about this actually – needs further thought. There is still the question of whether the duplicates are viewed a lot. And do these high duplicating users duplicate the same videos?

3.5 What is viewed

Many questions could be asked here. For highly duplicated videos, are all copies viewed at similar levels or not?

In [25]:
```
date = pd.to_datetime(video_df.DATA)
video_df.DATA = date
video_df.groupby(by=['TITLE'])['VIEWS'].sum().order()
```

Out [25]:
```
TITLE
Aaron Swartz - Wiki Article                                          0
Are your civil liberties being taken away? LIVE Debate               0
Christopher Hitchens  Wishful Thinking vs Evidence 2008)             0
Cyber Intelligence Sharing and Protection Act - Wiki Article         0
Freedom on the Net 2012 Release: Shifting Methods of Internet Control
0
...
The Most AMAZING video on the internet #Egypt #jan25
2474050
Wealth Inequality in America
6287314
Anonymous - Message to the American People
9612949
The Greatest Speech Ever Made
11744180
Collateral Murder - Wikileaks - Iraq
13640192
Emmanuel Kelly The X Factor 2011 Auditions Emmanuel Kelly FULL
14381492
Name: VIEWS, Length: 283, dtype: float64
```

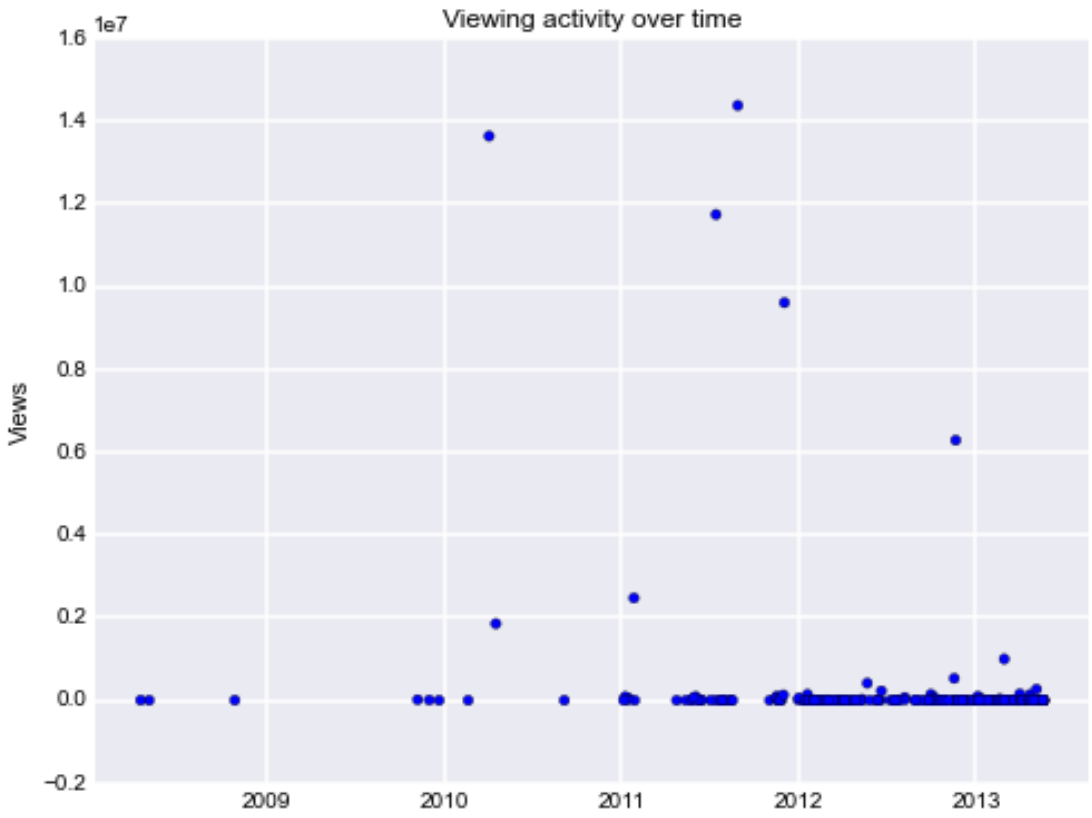
In [26]:
```
f = figure(figsize=(8,6), dpi=400)
s = f.add_subplot(111)
s.scatter(date, video_df.VIEWS)
s.set_title('Viewing activity over time')
s.set_ylabel('Views')

top_views.head()
```

```
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
/usr/local/lib/python2.7/dist-packages/pandas-0.12.0-py2.7-linux-
```

```
x86_64.egg/pandas/core/config.py:570: DeprecationWarning: height has
been deprecated.

  warnings.warn(d.msg, DeprecationWarning)
```

Out [26]:

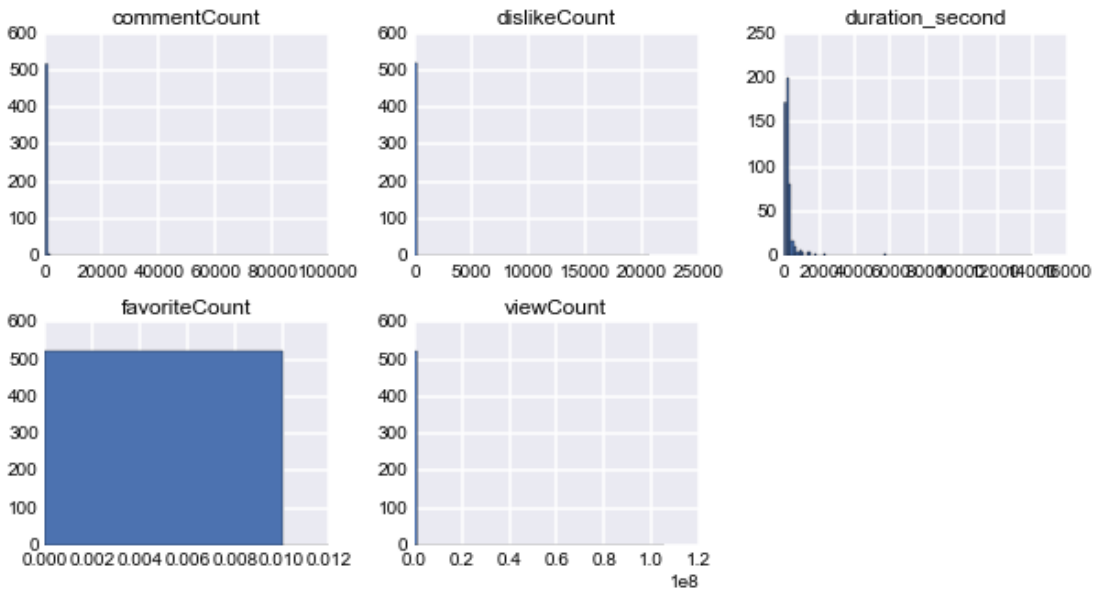
	TITLE	VIEWS
21	Anonymous Philippines message to Cybercrime La...	143349
45	Table Talk: Cons, Internet Freedom, & Spoiler ...	155988
58	Anonymous Calls for CISPA Internet Blackout Day	138356
60	F2C2012: Aaron Swartz keynote - "How we stopp...	415163
76	Anonymous - A Message to Congress on SOPA	101092



In []:
```
## Overviews of the data
```

In [27]:
```
df_am.columns
# df_am.commentCount.astype('int64').hist()

df_am[['commentCount', 'dislikeCount', 'favoriteCount', 'viewCount', 'duration_second'
```

Out [27]:
```
array([[<matplotlib.axes.AxesSubplot object at 0x7397f90>,
        <matplotlib.axes.AxesSubplot object at 0x954dad0>,
        <matplotlib.axes.AxesSubplot object at 0x6e86590>],
       [<matplotlib.axes.AxesSubplot object at 0x7c515d0>,
        <matplotlib.axes.AxesSubplot object at 0x81c0e10>,
        <matplotlib.axes.AxesSubplot object at 0x82aa890>]],
dtype=object)
```



Hard to see what is happening in the viewing of these videos.