Digifest 2016 analysis

Reading the dataset with pandas

http://pandas.pydata.org/ (http://pandas.pydata.org/)

In [1]:

```
import pandas as pd
```

We read in the input file to get a pandas DataFrame which we use in our analysis.

In [2]:

```
df = pd.read_csv('digifest2016dataset.csv') # dataframe object
```

Let's have a quick look at the DataFrame.

In [3]:

```
df
```

Out[3]:

	id_str	from_user	text	create
0	705802747971543040	laurajisc	More power to you' - @Jisc's opening video fro...	Fri Ma 17:11: +0000 2016
1	705802727276847104	mark8n	More power to you' - @Jisc's opening video fro...	Fri Ma 17:10: +0000 2016
2	705800984661004289	JournalArchives	RT @Jisc: Timelapse action from #digifest16. S...	Fri Ma 17:04: +0000 2016
3	705800366370250753	dr_mike_jones	Presentations, podcasts & resources from @...	Fri Ma 17:01: +0000 2016
4	705800366328307712	JiscLondon	Presentations, podcasts & resources from @...	Fri Ma 17:01: +0000 2016
5	705800203752882176	HistoricalTexts	RT @Jisc: Slides and resources from two very b...	Fri Ma 17:00: +0000 2016
6	705800083930005504	wagjuer	RT @coolcatteacher: Get #Digifest16 slides and...	Fri Ma 17:00: +0000 2016
7	705799329001422848	MobileTechExp	RT @ResourceProds: We have lots of digital/cre...	Fri Ma 16:57: +0000 2016
8	705798453847322624	g_fielding	RT @Jisc: Here it is - our list of the UK's to...	Fri Ma 16:53: +0000 2016
9	705798322263556097	hammel_rachel	RT @Jisc: Slides and resources from two very b...	Fri Ma 16:53: +0000 2016
				Fri Ma

10	705797988057219072	bobharrisonset	RT @MoodleMcKean: 'More power to you' - @Jisc'...	16:52: +0000 2016
11	705797827344146433	MoodleMcKean	More power to you' - @Jisc's opening video fro...	Fri Ma 16:51: +0000 2016
12	705797327806701568	HistoricalTexts	Jisc's investment in digital content for human...	Fri Ma 16:49: +0000 2016
13	705796891573952512	HistoricalTexts	Impacts of Digital Collections: EEBO & Hou...	Fri Ma 16:47: +0000 2016
14	705796639961710592	Borschtwithanna	RT @willrich45: Just Posted "Stop Innovating i...	Fri Ma 16:46: +0000 2016
15	705796032714690560	kevhickeyuk	Presentations, podcasts & resources from @...	Fri Ma 16:44: +0000 2016
16	705792935628050433	alstarkey	The challenge of international open access - J...	Fri Ma 16:32: +0000 2016
17	705792530177236993	JiscNorth	Presentations, podcasts & resources from @...	Fri Ma 16:30: +0000 2016
18	705792206859345921	coolcatteacher	Get #Digifest16 slides and presentations here:...	Fri Ma 16:29: +0000 2016
19	705791020504637441	morphospace	#digifest16 ART https://t.co/hK7E9RLzmF	Fri Ma 16:24: +0000 2016
20	705790875658543104	LeysDigital	RT @WarwickLanguage: Enhance your practice, ta...	Fri Ma 16:23: +0000 2016

21	705790858210250752	Jisc	#digifest16 might be over, but #networkshop44 ...	Fri Ma 16:23: +0000 2016
22	705790559902941184	JiscLondon	More power to you' - @Jisc's opening video fro...	Fri Ma 16:22: +0000 2016
23	705790261352394752	morphospace	RT @jamesclay: Find out more on #digitalcapabi...	Fri Ma 16:21: +0000 2016
24	705790261352394752	morphospace	RT @jamesclay: Find out more on #digitalcapabi...	Fri Ma 16:21: +0000 2016
25	705788250967957504	StrathclydeOA	RT @JuliaTaylorJisc: Presentations, podcasts &...	Fri Ma 16:13: +0000 2016
26	705787579589894144	RebseyOH	RT @EricStoller: Get in the sandbox of learnin...	Fri Ma 16:10: +0000 2016
27	705787470756061184	JuliaTaylorJisc	Presentations, podcasts & resources from @...	Fri Ma 16:10: +0000 2016
28	705787202958172161	megancodling	RT @Collabco: our new CEO Mark enjoying @Jisc ...	Fri Ma 16:09: +0000 2016
29	705786947546030086	martin_hamilton	RT @Jisc: We spoke to lastminute-dot-com found...	Fri Ma 16:08: +0000 2016
...	...	...	...	...
12469	701805686158663680	dkernohan	Yes - you can still book to attend #digifest16...	Mon F 22 16:28: +0000 2016
				Mon F 22

12470	701801548901289984	WileyLibINFO	RT @Jisc: "We've gone from having very little ...	16:11: +0000 2016
12471	701794685581402112	ThembaniMalapel	"We're overwhelmed with #data" - interview wit...	Mon F 22 15:44: +0000 2016
12472	701792419201871875	flaterik3	RT @Jisc: "We've gone from having very little ...	Mon F 22 15:35: +0000 2016
12473	701791445351587840	nopiedra	RT @Jisc: "We've gone from having very little ...	Mon F 22 15:31: +0000 2016
12474	701791090169528321	Jisc	"We've gone from having very little data to ha...	Mon F 22 15:30: +0000 2016
12475	701776326752149505	RedCityLab	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 14:31: +0000 2016
12476	701771353599426560	ciaran_talbot	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 14:11: +0000 2016
12477	701771306862321665	CinziaPG	RT @JiscScotland: Still time to book free plac...	Mon F 22 14:11: +0000 2016
12478	701766711540850688	ukinserbia	RT @i_gvs: A busy week for @i_gvs as we prepar...	Mon F 22 13:53: +0000 2016
				Mon F

12479	701765804774334465	vhmdale	RT @JiscScotland: Still time to book free plac...	22 13:49: +0000 2016
12480	701760794749444096	DominiqueVanpee	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:29: +0000 2016
12481	701757573838540800	tashtom	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:16: +0000 2016
12482	701756460854878208	JiscScotland	Still time to book free place at #digifest16 -...	Mon F 22 13:12: +0000 2016
12483	701755503257505792	AlisonMcNab	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:08: +0000 2016
12484	701752435216359424	Archaeology_UoS	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:56: +0000 2016
12485	701750912960897024	jrannali	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:50: +0000 2016
12486	701750556302430209	BigStorageETN	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:49: +0000 2016
12487	701749368576204800	idafensp	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:44: +0000 2016

12488	701740321063968770	Domicus	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:08: +0000 2016
12489	701738601516822528	frankbenneker	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:01: +0000 2016
12490	701737409768243202	kosson	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:56: +0000 2016
12491	701736038818062336	chriscb	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:51: +0000 2016
12492	701735483068563456	NHarvatt	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:49: +0000 2016
12493	701734313872777216	GleefulKaz	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:44: +0000 2016
12494	701734147220512769	Jisc	"We're overwhelmed with data" - interview with...	Mon F 22 11:43: +0000 2016
12495	701723914213261312	i_gvs	A busy week for @i_gvs as we prepare for 3 #ed...	Mon F 22 11:03: +0000 2016
12496	701715325738491904	Jisc_SnE	Still time to book free place at #digifest16 -...	Mon F 22 10:29: +0000 2016

12497	701712688116846592	Jisc_SnE	Still time to book free place at #digifest16 -...	Mon F 22 10:18: +0000 2016
12498	701712688116846592	Jisc_SnE	Still time to book free place at #digifest16 -...	Mon F 22 10:18: +0000 2016

12499 rows × 17 columns

We can see that a lot of the information we want is either in the text column or the entities_str column. Now we have to refine the dataset to extract some more information before we can do our analysis.

1. Refining the dataset

We refine the twitter dataset by using regular expressions and adding more columns to the pandas DataFrame that we can use for analysis.

We want to drop any duplicate rows from our dataset just in case.

In [4]:

```
df = pd.DataFrame.drop_duplicates(df)
```

Use regular expressions and json to further refine our dataset. **re_functions.py** contains functions that use regular expressions to match with the dataset and return a more refined result.

In [5]:

```
%run re_functions.py
```

Now let's add new columns to the dataframe by mapping the regular expression functions to get columns with more meaningful and easy to calculate data.

In [6]:

```
df['source_name'] = df['source'].map(source_name)
df['hashtags'] = df['entities_str'].map(get_hashtags)
df['user_mentions'] = df['entities_str'].map(get_mentions)
df['retweeted_user'] = df['text'].map(get_retweeted)
df['retweeted_user_list'] = df['retweeted_user'].map(to_list)
df['replied_to_user_list'] = df['in_reply_to_screen_name'].map(to_list)
```

```
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  from ipykernel import kernelapp as app
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  app.launch_new_instance()
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
```

For tweet types, we take 'Tweets' as anything that is not a reply or retweet, otherwise every single row of the dataset would count as a tweet.

In [7]:

```
df['tweet_type'] = df.apply(lambda row: tweet_type (row), axis = 1)
```

```
/cs/home/sy35/.local/lib/python3.4/site-packages/ipykernel/__main_
_.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

Now we have lots of new columns in the dataset we can use to analyse it easily! Let's look at our DataFrame again.

In [8]:

```
df
```

Out[8]:

	id_str	from_user	text	create
0	705802747971543040	laurajisc	More power to you' - @Jisc's opening video fro...	Fri Ma 17:11: +0000 2016
1	705802727276847104	mark8n	More power to you' - @Jisc's opening video fro...	Fri Ma 17:10: +0000 2016
2	705800984661004289	JournalArchives	RT @Jisc: Timelapse action from #digifest16. S...	Fri Ma 17:04: +0000 2016
3	705800366370250753	dr_mike_jones	Presentations, podcasts & resources from @...	Fri Ma 17:01: +0000 2016
4	705800366328307712	JiscLondon	Presentations, podcasts & resources from @...	Fri Ma 17:01: +0000 2016
5	705800203752882176	HistoricalTexts	RT @Jisc: Slides and resources from two very b...	Fri Ma 17:00: +0000 2016
6	705800083930005504	wagjuer	RT @coolcatteacher: Get #Digifest16 slides and...	Fri Ma 17:00: +0000 2016
7	705799329001422848	MobileTechExp	RT @ResourceProds: We have lots of digital/cre...	Fri Ma 16:57: +0000 2016
8	705798453847322624	g_fielding	RT @Jisc: Here it is - our list of the UK's to...	Fri Ma 16:53: +0000 2016
9	705798322263556097	hammel_rachel	RT @Jisc: Slides and resources from two very b...	Fri Ma 16:53: +0000 2016
				Fri Ma

10	705797988057219072	bobharrisonset	RT @MoodleMcKean: 'More power to you' - @Jisc'...	16:52: +0000 2016
11	705797827344146433	MoodleMcKean	More power to you' - @Jisc's opening video fro...	Fri Ma 16:51: +0000 2016
12	705797327806701568	HistoricalTexts	Jisc's investment in digital content for human...	Fri Ma 16:49: +0000 2016
13	705796891573952512	HistoricalTexts	Impacts of Digital Collections: EEBO & Hou...	Fri Ma 16:47: +0000 2016
14	705796639961710592	Borschtwithanna	RT @willrich45: Just Posted "Stop Innovating i...	Fri Ma 16:46: +0000 2016
15	705796032714690560	kevhickeyuk	Presentations, podcasts & resources from @...	Fri Ma 16:44: +0000 2016
16	705792935628050433	alstarkey	The challenge of international open access - J...	Fri Ma 16:32: +0000 2016
17	705792530177236993	JiscNorth	Presentations, podcasts & resources from @...	Fri Ma 16:30: +0000 2016
18	705792206859345921	coolcatteacher	Get #Digifest16 slides and presentations here:...	Fri Ma 16:29: +0000 2016
19	705791020504637441	morphospace	#digifest16 ART https://t.co/hK7E9RLzmF	Fri Ma 16:24: +0000 2016
20	705790875658543104	LeysDigital	RT @WarwickLanguage: Enhance your practice, ta...	Fri Ma 16:23: +0000 2016

21	705790858210250752	Jisc	#digifest16 might be over, but #networkshop44 ...	Fri Ma 16:23: +0000 2016
22	705790559902941184	JiscLondon	More power to you' - @Jisc's opening video fro...	Fri Ma 16:22: +0000 2016
23	705790261352394752	morphospace	RT @jamesclay: Find out more on #digitalcapabi...	Fri Ma 16:21: +0000 2016
25	705788250967957504	StrathclydeOA	RT @JuliaTaylorJisc: Presentations, podcasts &...	Fri Ma 16:13: +0000 2016
26	705787579589894144	RebseyOH	RT @EricStoller: Get in the sandbox of learnin...	Fri Ma 16:10: +0000 2016
27	705787470756061184	JuliaTaylorJisc	Presentations, podcasts & resources from @...	Fri Ma 16:10: +0000 2016
28	705787202958172161	megancodling	RT @Collabco: our new CEO Mark enjoying @Jisc ...	Fri Ma 16:09: +0000 2016
29	705786947546030086	martin_hamilton	RT @Jisc: We spoke to lastminute-dot-com found...	Fri Ma 16:08: +0000 2016
30	705786674295480321	EdTechRetweet	RT @MoodleMcKean: Presentations, podcasts &amp...	Fri Ma 16:07: +0000 2016
...	...	...	...	...
12468	701816866352672772	GawaliKiran	RT @Jisc: "We've gone from having very little ...	Mon F 22 17:12: +0000 2016
				Mon F 22

12469	701805686158663680	dkernohan	Yes - you can still book to attend #digifest16...	16:28: +0000 2016
12470	701801548901289984	WileyLibINFO	RT @Jisc: "We've gone from having very little ...	Mon F 22 16:11: +0000 2016
12471	701794685581402112	ThembaniMalapel	"We're overwhelmed with #data" - interview wit...	Mon F 22 15:44: +0000 2016
12472	701792419201871875	flaterik3	RT @Jisc: "We've gone from having very little ...	Mon F 22 15:35: +0000 2016
12473	701791445351587840	nopiedra	RT @Jisc: "We've gone from having very little ...	Mon F 22 15:31: +0000 2016
12474	701791090169528321	Jisc	"We've gone from having very little data to ha...	Mon F 22 15:30: +0000 2016
12475	701776326752149505	RedCityLab	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 14:31: +0000 2016
12476	701771353599426560	ciaran_talbot	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 14:11: +0000 2016
12477	701771306862321665	CinziaPG	RT @JiscScotland: Still time to book free plac...	Mon F 22 14:11: +0000 2016
				Mon F

12478	701766711540850688	ukinserbia	RT @i_gvs: A busy week for @i_gvs as we prepar...	22 13:53: +0000 2016
12479	701765804774334465	vhmdale	RT @JiscScotland: Still time to book free plac...	Mon F 22 13:49: +0000 2016
12480	701760794749444096	DominiqueVanpee	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:29: +0000 2016
12481	701757573838540800	tashtom	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:16: +0000 2016
12482	701756460854878208	JiscScotland	Still time to book free place at #digifest16 -...	Mon F 22 13:12: +0000 2016
12483	701755503257505792	AlisonMcNab	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 13:08: +0000 2016
12484	701752435216359424	Archaeology_UoS	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:56: +0000 2016
12485	701750912960897024	jrannali	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:50: +0000 2016
12486	701750556302430209	BigStorageETN	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:49: +0000 2016

12487	701749368576204800	idafensp	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:44: +0000 2016
12488	701740321063968770	Domicus	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:08: +0000 2016
12489	701738601516822528	frankbenneker	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 12:01: +0000 2016
12490	701737409768243202	kosson	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:56: +0000 2016
12491	701736038818062336	chriscb	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:51: +0000 2016
12492	701735483068563456	NHarvatt	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:49: +0000 2016
12493	701734313872777216	GleefulKaz	RT @Jisc: "We're overwhelmed with data" - inte...	Mon F 22 11:44: +0000 2016
12494	701734147220512769	Jisc	"We're overwhelmed with data" - interview with...	Mon F 22 11:43: +0000 2016
12495	701723914213261312	i_gvs	A busy week for @i_gvs as we prepare for 3 #ed...	Mon F 22 11:03: +0000 2016

12496	701715325738491904	Jisc_SnE	Still time to book free place at #digifest16 -...	Mon F 22 10:29: +0000 2016
12497	701712688116846592	Jisc_SnE	Still time to book free place at #digifest16 -...	Mon F 22 10:18: +0000 2016

12438 rows × 24 columns

We can see to the far right of the table, there are many new columns which we can use for analysis such as hashtags, user_mentions, tweet_type.

Save the refined dataset to 'digifest16_refined.csv'

In [9]:

```
df.to_csv('digifest16_refined.csv')
```

2. Structure of the dataset

We can now look at the structure of the dataset.

This import allows visualisations to be displayed inline.

In [10]:

```
get_ipython().magic('matplotlib inline')
```

In [11]:

```
df['tweet_type'].value_counts()
```

Out[11]:

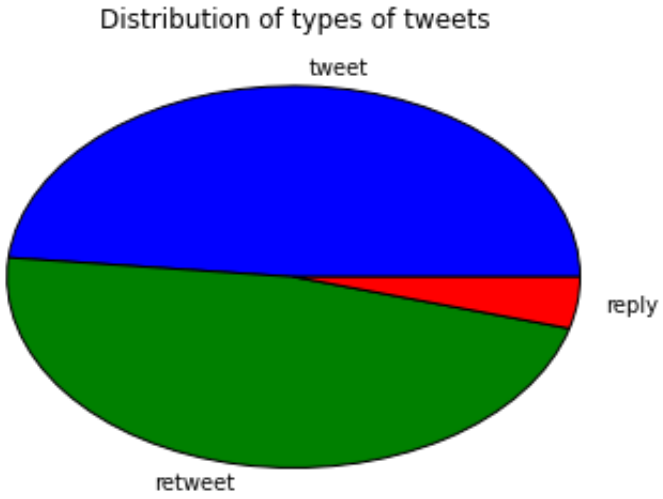
```
tweet      6019
retweet    5877
reply       542
Name: tweet_type, dtype: int64
```

In [12]:

```
tweet_type_pie = df['tweet_type'].value_counts().plot(kind = 'pie',
                                                      title = 'Distribution o
f types of tweets'
                                                     )
tweet_type_pie.set_ylabel('')
```

Out[12]:

<matplotlib.text.Text at 0x7f4ceae8a470>



This shows that tweets are the majority of the messages sent, followed relatively closely by retweets, with replies being the least prominent form of communication by a large amount.

3. Means and standard deviations

Using our refined dataset, we can easily calculated the mean and standard deviations for replies, retweets and tweets.

In [13]:

```
typegroup = df.groupby(['tweet_type', 'from_user'])
refinedtg = typegroup.size().reset_index()
refinedtg.rename(columns = {0:'Number'}, inplace = True)
```

Mean and standard deviation for replies

In [14]:

```
replies = refinedtg.loc[refinedtg['tweet_type'] == 'reply']
rpmean = replies.mean()['Number']
rpstd = replies.std()['Number']
```

Mean and standard deviation for retweets

In [15]:

```
retweets = refinedtg.loc[refinedtg['tweet_type'] == 'retweet']
rtmean = retweets.mean()['Number']
rtstd = retweets.std()['Number']
```

Mean and standard deviation for tweets Tweets here are

In [16]:

```
tweets = refinedtg.loc[refinedtg['tweet_type'] == 'tweet']
tmean = tweets.mean()['Number']
tstd = tweets.std()['Number']
```

Putting it all together into a dictionary

In [17]:

```
tweetdict = {'Mean per user':{'Tweets':tmean, 'Retweets':rtmean, 'Replies':rpm
ean},
             'Standard deviation per user':{'Tweets':tstd, 'Retweets':rtstd,
'Replies':rpstd}}
```

In [18]:

```
pd.DataFrame.from_dict(tweetdict)
```

Out[18]:

	Mean per user	Standard deviation per user
Replies	2.509259	4.291680
Retweets	3.187093	8.249384
Tweets	7.542607	18.821824

This table shows that most users replied to tweets a fairly low number of times (~2.5), and all users replied a fairly similar amount, the number of re-tweets was slightly higher at around 3.2 but there was a higher standard deviation so some users will of re tweeted a lot which will have skewed the data. The number of tweets per user on average was more than double the re-tweets at ~7.5 but it also had a far higher standard deviation so there were some users which tweeted at a far higher rate than that, as 99% of users are likely to of tweeted less than 47 times due to the normal distribution of data.

In [19]:

```
retweeted = df['retweeted_user'].value_counts()
replied = df['in_reply_to_screen_name'].value_counts()
```

In [20]:

```
responsedict = {'Mean':{'Times retweeted'
                        :retweeted.mean(),
                        'Times replied to'
                        :replied.mean()},
                'Standard deviation':{'Times retweeted'
                                      :retweeted.std(),
                                      'Times replied to'
                                      :replied.std()}}
```

Here we are only including users who were retweeted/replied to at least once

In [21]:

```
pd.DataFrame.from_dict(responsedict)
```

Out[21]:

	Mean	**Standard deviation**
Times replied to	2.326180	3.639587
Times retweeted	12.043033	54.833444

This table was created by removing the data from all users who had 0 re-tweeted/replied to tweets and running mean and standard deviation on the remaining data. Replies were largely similar in mean and standard deviation terms to the previous data, the small losses being fairly obviously due to the reduction of data points so there was a higher number of replies per tweet that was replied to. The re-tweet information shows a large spread of data, the mean is shown as 12 but due to having such a high standard deviation this could be considered unreliable statistically speaking, the high standard deviation could be put down to users like jisc who will of received a huge number of re-tweets, skewing the dataset.

4. Timeline of activity

Here we look at the timeline of all tweets over the period of time covered by our dataset. This can show us when was the time that #digifest16 was being tweeted a lot.

In [22]:

```
tweet_times = []
```

We get the tweet times using the 'created_at' column in our DataFrame

In [23]:

```
for time_tweeted in df['created_at']:
    tweet_times.append(time_tweeted)
```

Using the pandas DatetimeIndex, we can plot the sums of each time interval.

In [24]:

```
ones = [1]*len(tweet_times)
idx = pd.DatetimeIndex(tweet_times)
ts = pd.Series(ones, index=idx)
```

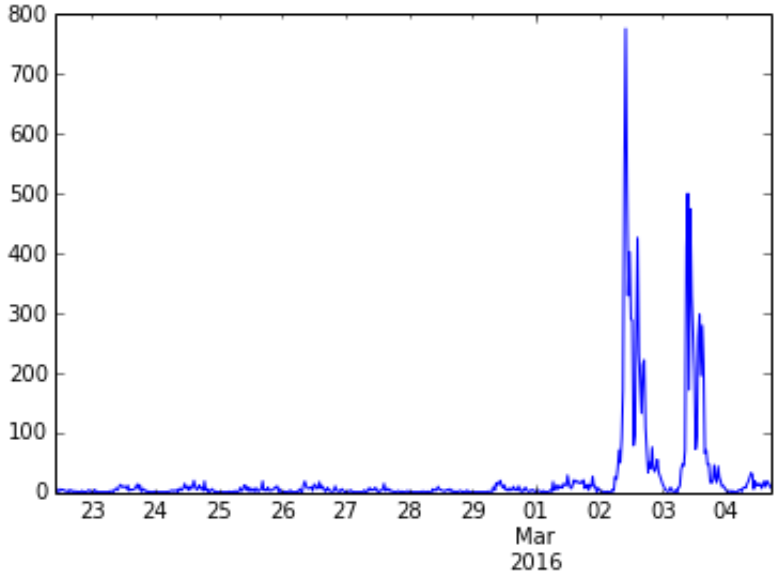
Let's resample every 30 minutes, since this dataset covers a period of over a week.

In [25]:

```
resampled = ts.resample('30Min').sum().fillna(0)
```

In [26]:

```
ax1 = resampled.plot()
```



As we can see, the twitter activity for #digifest16 was rather low the week before it started, with the peak of the activity being the 2nd and 3rd of March, the actual days of the event.

Let's look closer at the activity on those days.

First we have to import the time module to create a unix timestamp of the date: 02 March 2016.

In [27]:

```
import time
import datetime
```

In [28]:

```
date_str = '02/03/2016'
datestart = time.mktime(datetime.datetime.strptime(date_str, '%d/%m/%Y').timet
uple())/60
```

Now we plot the graph at a shorter interval (5 minutes) for more detail on the activity.

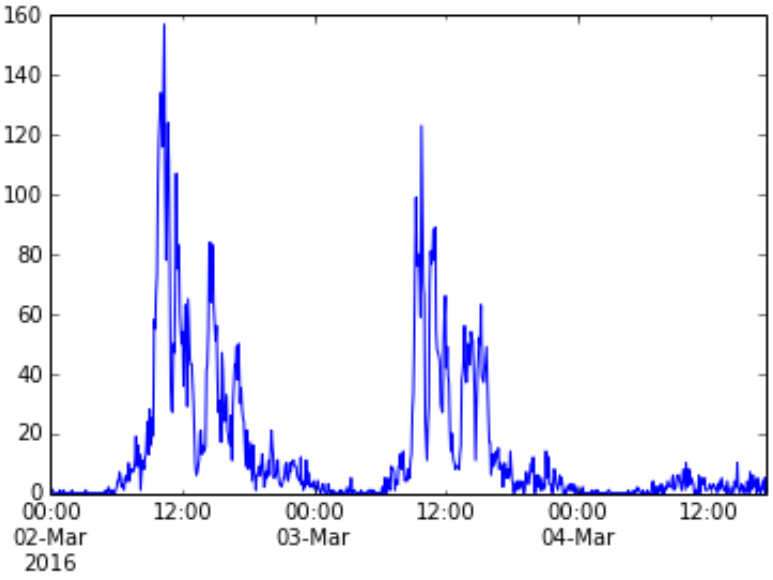
In [29]:

```
import matplotlib.ticker as ticker

resampled = ts.resample('5Min').sum().fillna(0)
ax1 = resampled.plot()
ax1.set_xlim(left = datestart)
```

Out[29]:

```
(24281280.0, 24285190.0)
```



From the graph we can see the pattern of twitter activity during the two days of digifest.

- The activity is highest in the morning, from about 9am to 12pm
- There is a brief lull in activity after noon as everyone is having lunch
- During the afternoon, the activity is less than during the morning
- The activity is less on the second day of the event

Hashtag timeline

We use Counter to count the number of hashtags and store the result in a tuple.

In [30]:

```
from collections import Counter
```

Get the most common hashtags from the dataset

In [31]:

```
hts_list = [ht for ht in df.hashtags]
hts = [ht for sublist in hts_list for ht in sublist]
ht_count = Counter()
```

Map using str.lower to turn all hashtags to lower case

In [32]:

```
ht_count.update(map(str.lower, hts))
```

Now we can see the most common hashtags and their counts!

In [33]:

```
ht_count.most_common(10)
```

Out[33]:

```
[('digifest16', 12456),
 ('edtech', 327),
 ('digitalcapability', 214),
 ('feltag', 170),
 ('jisc50social', 154),
 ('learninganalytics', 105),
 ('jiscrdm', 77),
 ('wednesdaywisdom', 58),
 ('hulldtn', 56),
 ('socialmedia', 54)]
```

Let's make functions to create a plot of the activity given a hashtag

In [34]:

```
%run hashtag_plot.py
```

```
<matplotlib.figure.Figure at 0x7f4cf12049e8>
```

Now let's look at the 2nd, 3rd and 4th most popular hashtags together. We are intentionally ignoring the most popular hashtag (#digifest16) because almost every single tweet in the dataset uses this hashtag and it would dwarf the other hashtags on the graph.

In [35]:

```
h1 = 'edtech'
h2 = 'digitalcapability'
h3 = 'feltag'
```

In [36]:

```
h1_plot = get_hashtag_plot(h1, df)
h2_plot = get_hashtag_plot(h2, df)
h3_plot = get_hashtag_plot(h3, df)
```

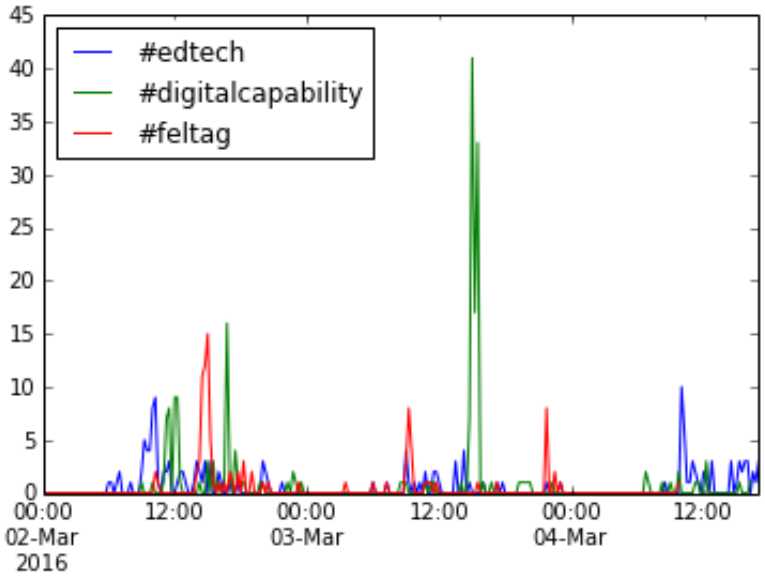
We will again just focus on the 2 days of digifest to see the kind of patterns these hashtags have.

In [37]:

```
df2 = pd.DataFrame({'#'+h1: h1_plot, '#'+h2: h2_plot, '#'+h3: h3_plot},
                   columns=['#'+h1, '#'+h2, '#'+h3])
ax2 = df2.plot()
ax2.set_xlim(left = datestart)
```

Out[37]:

```
(24281280.0, 24285180.0)
```



We can see from this date that there are clear spikes for some of the hashtags during certain periods of the event. #digitalcapability has a large spike during the afternoon of the 3rd of March. From a quick search we can see that digitalcapability (https://www.jisc.ac.uk/rd/projects/building-digital-capability) is a Jisc project so it is likely that there was some big annoucement on the 3rd of March.

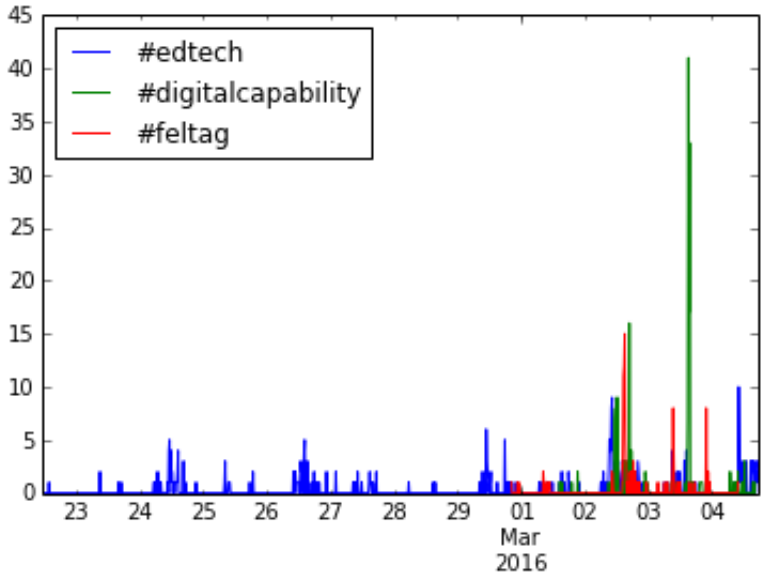
Even though **#edtech** was the second most popular hashtag on the list, it is not used very often during the 2 days of digifest, so let's look at this again with the time of the whole dataset.

In [38]:

```
df2.plot()
```

Out[38]:

```
<matplotlib.axes.AxesSubplot at 0x7f4ceadb7c18>
```



Now we can see that **#edtech** was very consistently tweeted during the whole period of the dataset. Edtech (http://www.edtechmagazine.com/) is a tech magazine so they likely would have been doing coverage of digifest and posting articles about the event the week before it started. There is also a small spike of activity of #edtech on the 4th of March, one day after digifest. Again this is probably due to Edtech providing coverage articles after the event.

5. Hashtag cloud

We use the wordcloud (http://amueller.github.io/word_cloud/) library to create a nice visualisation of the most common hashtags.

In [39]:

```
%run word_cloud.py
```

```
<matplotlib.figure.Figure at 0x7f4cf1161c18>
```

Using this library, we are able to generate a very colourful visualisation of the most popular hashtags used in the dataset. The library also scales the tuples well, so that even #digifest16, which we had to ignore in the section above is not overwelmingly larger than the other hashtags. The library can also be used on

In [40]:

```
hashtag_wordcloud = make_wc(ht_count.most_common(100))
```

In [41]:

```
import matplotlib.pyplot as plt
```

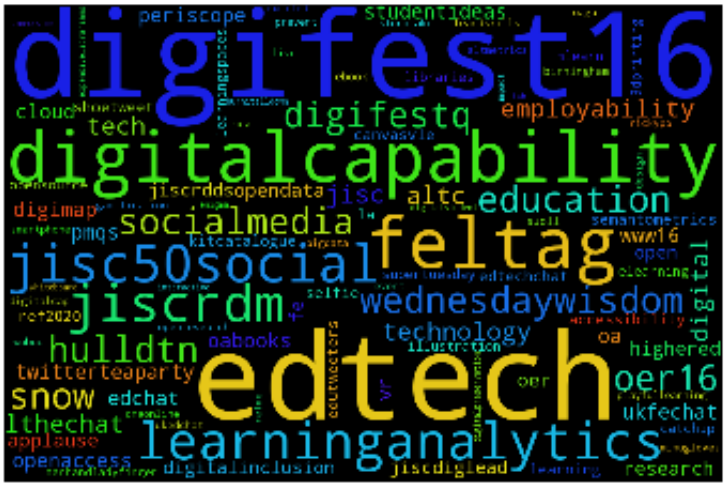
Let's make the visualisation cleaner by removing gridlines and print the visualisation to a file.

In [42]:

```
plt.axis('off')
plt.imshow(hashtag_wordcloud)
```

Out[42]:

```
<matplotlib.figure.AxesImage at 0x7f4ceacf7400>
```



This visualisation shows the most common hashtags by increasing the size as the frequency increased with the most populat hashtags being the biggest. The colours help to draw the users eye to increase recognition of the cloud at a glance, this type of graphic is more suited to display rather than analysis as it lacks any real numerical data.

The library also provides a way to mask the word cloud given a coloured masked image for an even nicer visualisation. Below is the word cloud for the most popular users that were mentioned during digifest masked using the digifest logo.

In [43]:

```
mentions = [m for sublist in [m for m in df.user_mentions] for m in sublist]
mentions_count = Counter()
mentions_count.update(mentions)
```

In [44]:

```
mentions_wordcloud = make_wc(mentions_count.most_common(100), 'logo.png', Tru
e)
```

Set the size for the visualisation and remove gridlines to make it more clearly readable.

In [45]:

```
plt.figure(figsize=(20,20))
plt.axis('off')
plt.imshow(mentions_wordcloud)
```

Out[45]:

```
<matplotlib.image.AxesImage at 0x7f4cea8768d0>
```

We can see from this visualisation quite clearly that the users Jisc, EricStoller, DonnaLanclos... are the more mentioned users during digifest 2016.

6. Applications used to send tweets

We can see the different sources from our refined column 'source_name'

In [46]:

```
sources = df['source_name'].value_counts()
```

Now we can see all the different sources that users tweeted from thanks to our refined dataset.

In [47]:

```
sources
```

Out[47]:

```
Twitter Web Client             2713
Twitter for iPhone             2555
Twitter for Android            2016
TweetDeck                      1567
Twitter for iPad               1265
Hootsuite                       771
iOS                             421
GaggleAMP                       274
Twitter for Windows Phone       103
RoundTeam                       101
Tweetbot for iOS                 60
DoubleDutch                      59
Buffer                           56
TweetCaster for Android          51
Twitter for Mac                  38
Twitterrific                     38
Echofon                          35
Instagram                        17
tweecha for android              16
Periscope                        11
HubSpot                          10
Twitter for Windows               9
Lissted                           9
Tweetium for Windows              8
IFTTT                             7
Twitter for Android Tablets       7
Talon Plus                        6
twicca                            5
TweetCaster for iOS               5
GIS Sharer                        5
                                ...
Carbon v2                         2
VR retweet                        2
Richard V                         1
Sprout Social                     1
sergperu                          1
TrendTweeter                      1
LollyDaskal_Roomee                1
Ickebot 3000                      1
Choqok                            1
Twuffer                           1
Polly                             1
NoSQLDigest                       1
new astoncliton_rt                1
Retweet rndteam                   1
Percolate                         1
UberSocial for Android            1
MeTweets for Windows Phone        1
PrayApp                           1
Plume for Android                 1
What the Trend                    1
Lowlands CPH Project              1
TipTop App                        1
Tweet Suite Plugin                1
SocialHammer Application32        1
Tweetings for Android             1
SocialSignIn Application          1
```

```
sdfdgsddfgsa                    1
gameskids                       1
Vine for Android                1
Storify                         1
Name: source_name, dtype: int64
```

In [48]:

```
sources.size
```

Out[48]:

86

Using this data, we can plot a graph to more clearly see what percentage of users are using the various different applications. But first we have to group all the sources which are very uncommon (<10% of the most common source) into an 'Other' field.

In [49]:

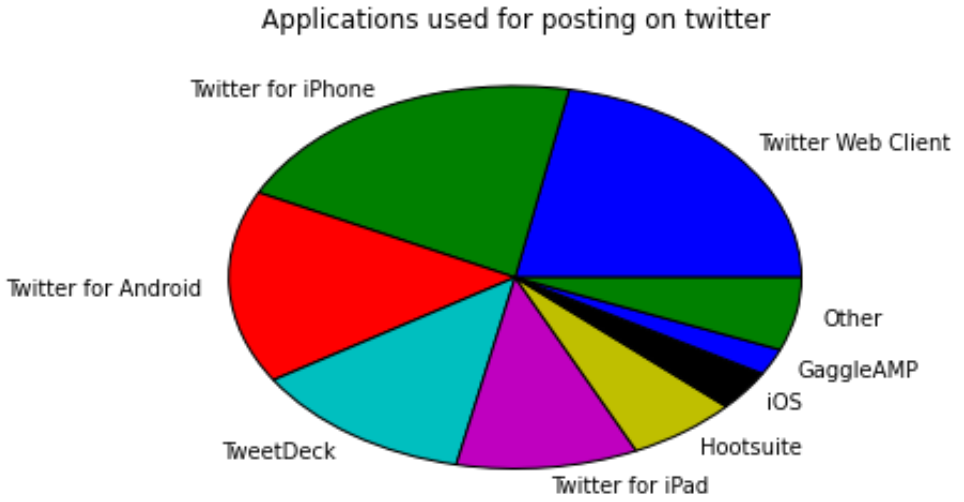
```
other_total = sources.select(lambda label: sources[label] < sources.max()/10).
sum()
sources = sources.select(lambda label: sources[label] >= sources.max()/10)
sources['Other'] = other_total
```

In [50]:

```
source_plot = sources.plot(kind='pie', title = 'Applications used for posting
on twitter')
source_plot.set_ylabel('')
```

Out[50]:

<matplotlib.text.Text at 0x7f4cea802518>



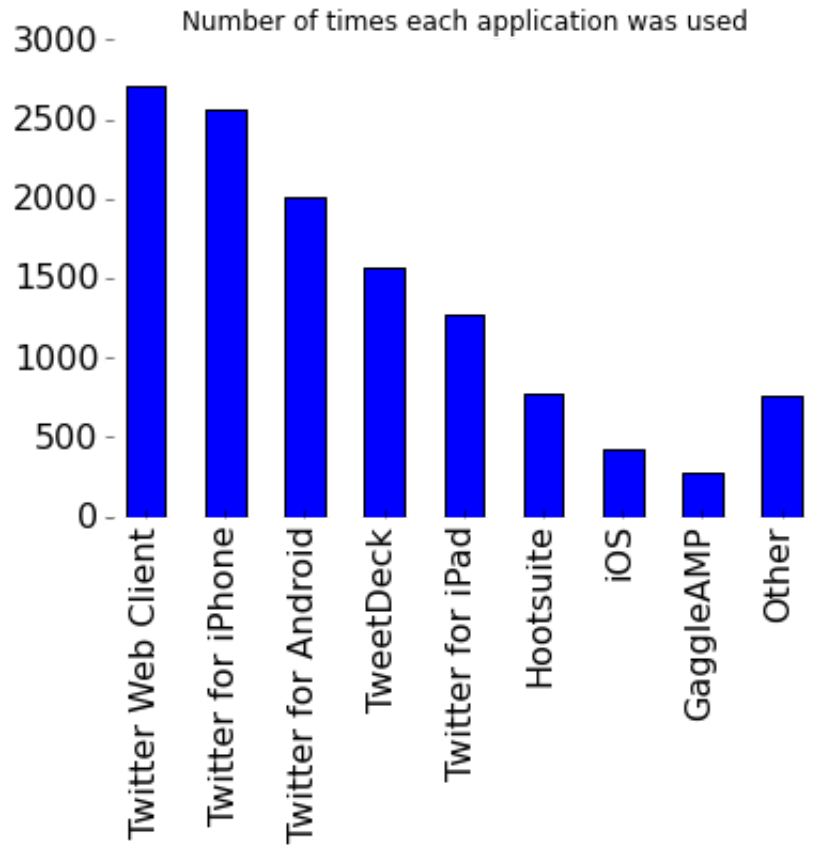
This pie chart shows the relative frequencies of use of each of the applications used to upload tweets with applications with a less than 10% share being grouped together for readability. It shows the twitter clients for iPhone and the web client had a lion's share of the tweets, with android being placed third, there were a further 5 applications which had over 10% of the tweets. This was to be expected as at a symposium there would be less access to web clients than normal, but the access which was afforded would be to people like journalists and company representatives who are likely to have been the source of a large volume of tweets per user. The high phone usage is to be expected as lots of tweets will have been from attendees with access to mobile technologies during the event.

Now let's look at the same data in a bar chart for a more numerical representation.

In [51]:

```
ax = sources.plot(kind='bar', title='Number of times each application was use
d', fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.spines['right'].set_visible(False)

ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()
```



This bar graph shows the application usage in numerical form with 'other' being created in the same way as previously. It shows that the web client had slightly more tweets originating from it (~200) than iPhone; all of the other clients had greater differences between them. It also shows that nearly 1000 tweets were sent from other service which is surprising as one would expect that far fewer would be sent from smaller services overall.

7. Networkx and Graphviz

Using the networkx (https://networkx.github.io/) and graphviz (http://www.graphviz.org/) libraries, we are able to create graph networks to show the interactions between users, easily seeing which users are mentioned, replied and retweeted the most. The graphs created are quite large and take some time to load because they contain many nodes.

In [52]:

```
import networkx as nx
```

We want to create a DiGraph because we want our edges to be directional so we can see who is the user being mentioned and who is the user mentioning someone else.

In [53]:

```
g = nx.DiGraph()
```

Because of the number of nodes we have, we want to use graphviz as it as options to set no overlap between our nodes.

In [54]:

```
A = nx.drawing.nx_agraph.to_agraph(g)
```

Now let's populate our graph with some nodes! First we will need some data!

We cannot create a graph with every single row in our dataset as there would be too many nodes and the graph would take a long time to generate and may be too clustered for us to see anything useful. So let's take a sample of 1000 entries in our dataset first and have a look. Here we'll look at retweets and try to see which users have been retweeted the most.

In [55]:

```
amount = 1000
datalist = df.retweeted_user.dropna().sample(amount)
```

In [56]:

```
xs = [x for x in datalist]
count = Counter(xs)
```

In [57]:

```
#creates tuples of (row[t1], row[t2]) and append to tuple_list
def populate_tuple(row, tuple_list, t1, t2):
    tuple_list.append((row[t1], row[t2]))
```

In [58]:

```
tuple_list = []
df.dropna(subset=['retweeted_user']).sample(amount).apply(
    lambda row: populate_tuple (row, tuple_list, 'from_user', 'retweeted_use
r'), axis=1);
```

With our data in the tuple_list, we can now populate the graph with nodes and edges.

In [59]:

```
for x in count:
    n = count[x]
    length = len(count)
    A.add_node(x, label=x, fontsize=12+2*n,
               height=1+n/(length/2), width=1+n/(length/3),
               color='firebrick1')
```

We also need to connect the nodes up with edges using our tuple list from earlier. This will show directed lines which let us see the user interaction for retweets.

In [60]:

```
A.add_edges_from(tuple_list)
```

Finally we just have to change a few of the graph settings so there is no overlap between nodes and lines.

In [61]:

```
A.graph_attr['overlap'] = 'false'
A.graph_attr['splines'] = 'true'
A.node_attr['style'] = 'filled'
A.node_attr['color'] = 'firebrick1'
A.node_attr['fontname'] = 'Helvetica'
```

Now we can create an image of visualisation and save it as “retweeted_network.png”

In [62]:

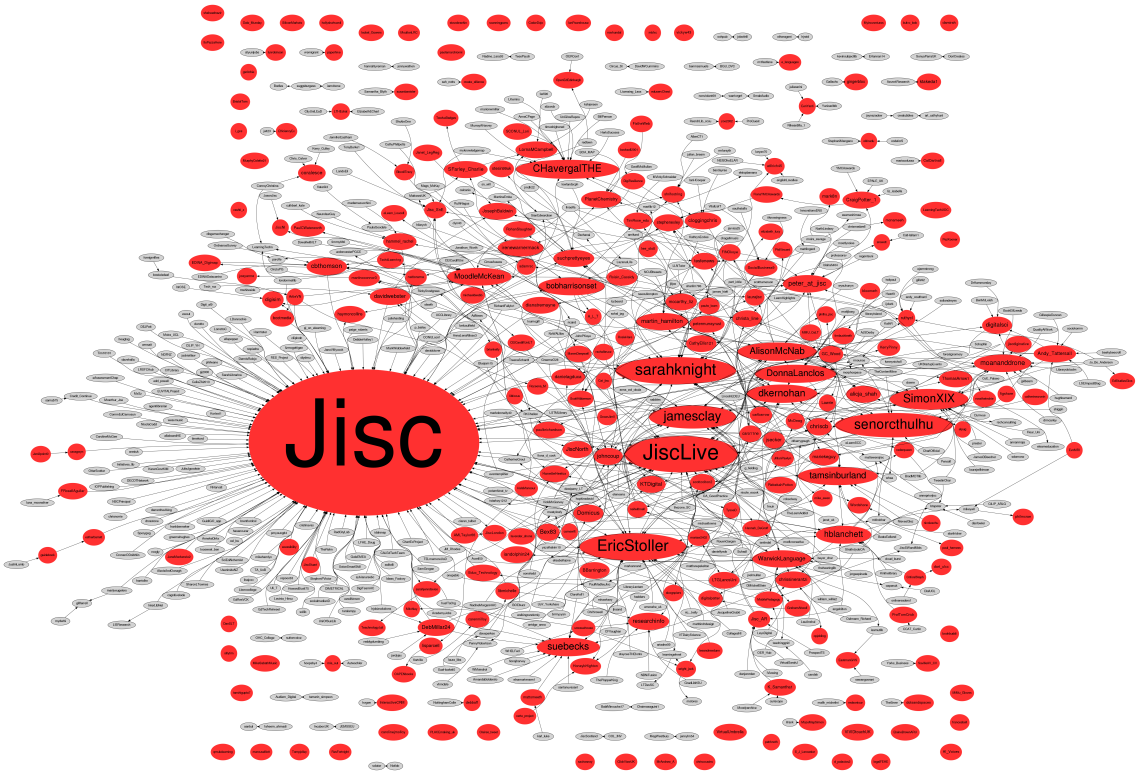
```
A.draw('retweeted_network.png', prog='sfdp')
```

Let's have a look at the image we created.

In [63]:

```
from IPython.display import Image
Image(filename='retweeted_network.png')
```

Out[63]:



The red nodes represent users in our sample that have been retweeted and grey nodes are users that retweeted others but are not retweeted themselves.

Similar to the the popular users we saw earlier using user mentions, Jisc is the main user being retweeted (this is shown by the arrows pointing to it). There are also other users that have less retweets but still stand above the rest, these users (SimonXIX, sarahknight, EricStoller...) also appeared in the user mentions section above. It is also interesting to note that Jisc rarely (if at all) re-tweeted anyone else but understandably large numbers of people re-tweeted jisc whereas the other larger nodes more commonly re-tweeted other users.

We can do the same for replies instead of retweets, hopefully this time the more popular users are different since large company twitters like Jisc are unlikely to reply all the time.

In [64]:

```
datalist = df.in_reply_to_screen_name.dropna()
g2 = nx.DiGraph()
A2 = nx.drawing.nx_agraph.to_agraph(g2)
```

No need to take a sample here, since there are only a very small number of replies in the dataset.

In [65]:

```
xs = [x for x in datalist]
count = Counter(xs)
```

In [66]:

```
#creates tuples of (row[t1], row[t2]) and append to tuple_list
def populate_tuple(row, tuple_list, t1, t2):
    tuple_list.append((row[t1], row[t2]))
```

In [67]:

```
tuple_list = []
df.dropna(subset=['in_reply_to_screen_name']).apply(
    lambda row: populate_tuple (row, tuple_list, 'from_user', 'in_reply_to_scr
een_name'), axis=1);
```

With our data in the tuple_list, we can now populate the graph with nodes and edges.

In [68]:

```
for x in count:
    n = count[x]
    length = len(count)
    A2.add_node(x, label=x, fontsize=12+2*n,
              height=1+n/(length/2), width=1+n/(length/3),
              color='firebrick1')
```

We also need to connect the nodes up with edges using our tuple list from earlier. This will show directed lines which let us see the user interaction for retweets.

In [69]:

```
A2.add_edges_from(tuple_list)
```

Finally we just have to change a few of the graph settings so there is no overlap between nodes and lines.

In [70]:

```
A2.graph_attr['overlap'] = 'false'
A2.graph_attr['splines'] = 'true'
A2.node_attr['style'] = 'filled'
A2.node_attr['color'] = 'firebrick1'
A2.node_attr['fontname'] = 'Helvetica'
```

Now create the visualisationand save it as “retweeted_network.png”

In [71]:

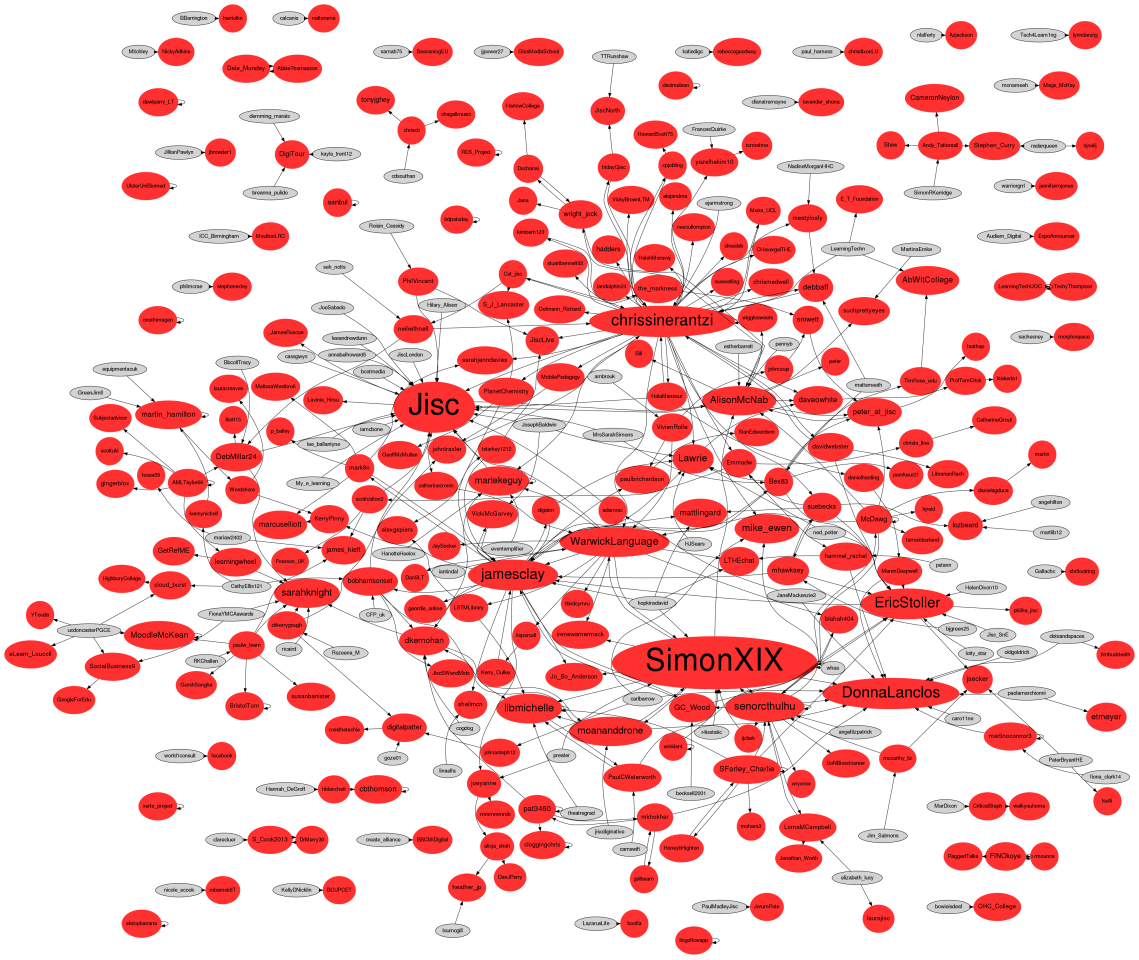
```
A2.draw('replies_network.png', prog='sfdp')
```

Let's have a look at the replies network.

In [72]:

```
from IPython.display import Image
Image(filename='replies_network.png')
```

Out[72]:



Unremarkably, the data shows very similar trends to the previous graph, only that jisc is smaller as a node due to them not receiving as many replies as perhaps their tweets were less conversational and more like announcements than other users, furthermore the other users which were prominent in the previous graphic are also more likely to reply to other users than jisc, this may be due to the way the accounts are run as in the previous graphic there was a user named ‘JiscLive’ this may be the manned part of the jisc twitter team.

We can also use networkx to create a different visualisation for the most common hashtags.

In [73]:

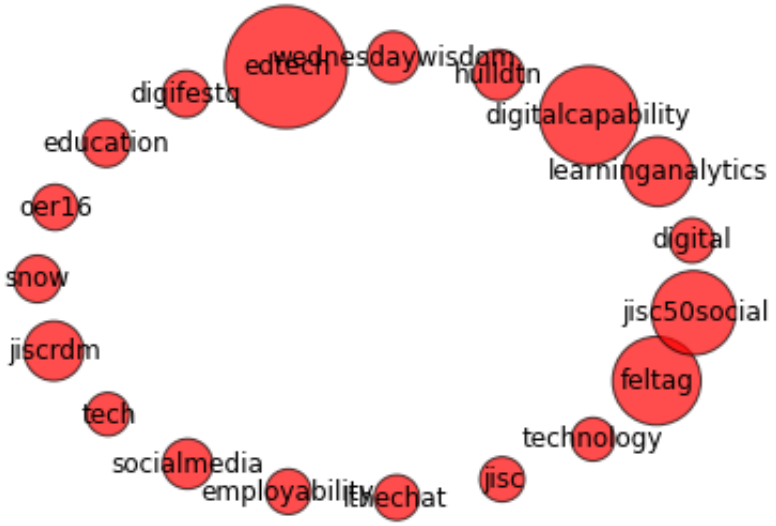
```
ht_bubbles = nx.Graph()
ht_bubbles.add_nodes_from([row[0] for row in ht_count.most_common(20) if row
[0] != "digifest16"], count=0)
```

Let's define the add_count method which populates the count attribute of each node. This adds bubbles to the visualization. The hashtag 'digifest16' is ignored as the bubble size be too large would render the graphic useless.

In [74]:

```
def add_count(g, rows):
    for row in rows:
        if row[0] != 'digifest16':
            g.node[row[0]]['count'] = row[1]

add_count(ht_bubbles, ht_count.most_common(20))
cloud_sizes = [10 + 10*ht_bubbles.node[s]['count'] for s in ht_bubbles.nodes
()]
nx.draw(ht_bubbles, node_size=cloud_sizes, alpha=0.7, with_labels=True)
```



This graphic shows the most popular hashtags by bubble size, showing the prominence of the 20 most common hashtags, it is unsurprising that jisc features in 3 of them, and digifest is in one, a large proportion which also make the list are technology related such as ‘technology’ and ‘digital’. Overall I would say this graph shows that a large number of the popular hashtags are all in the same areas to allow the best coverage in the groups which are interested.

8. Followers, friends and retweets

Let's define a function that calculates the number of times a user has been retweeted. Any row which had more than 5000 or less than 1 followers or friends are removed because jisc would skew the data too much and lead to a poor visualisation and 0 of either followers or friends would result in a large clump of data points at the origin.

In [75]:

```
def getRetweets(s):
    num=0
    q=df[(df['user_friends_count']>0)&(df['user_friends_count']<5000)&(df['use
r_followers_count']>0)&(df['user_followers_count']<5000)]['retweeted_user'].dr
opna().tolist()
    for i in q:
        if s in i:
             num=num+1
    return num


y = pd.Series([getRetweets(s) for s in df[(df['user_friends_count']>0)&(df['us
er_friends_count']<5000)&(df['user_followers_count']>0)&(df['user_followers_co
unt']<5000)]['from_user']])
```

In [85]:

```
from mpl_toolkits.mplot3d import Axes3D
#use tk to show the 3D plot in a window where it is interactive.
%matplotlib tk
```

We create lists from the pandas series and check that the number of retweets is not 0, and the number of friends and followers is between 0 and 5000 , this is because jisc would skew the data too much and lead to a poor visualisation and 0 of either followers or friends would result in a large clump of data points at the origin. Then we add the data points to the visualisation and set the colour to #aa3863, add the labels to the axis to explain what each is, create a 3d projection on which to plot followers friends and retweet, and finally, set the viewing angle of the visualisation and elevation and show the visualisation from the desired perspective.

In [84]:

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

xs = y[(y>0)&(y<100)] [y].tolist()
ys = df[(df['user_friends_count']>0)&(df['user_friends_count']<5000)&(df['user
_followers_count']>0)&(df['user_followers_count']<5000)]['user_friends_coun
t'].tolist()
zs = df[(df['user_friends_count']>0)&(df['user_friends_count']<5000)&(df['user
_followers_count']>0)&(df['user_followers_count']<5000)]['user_followers_coun
t'].tolist()
depthshade=True
ax.scatter(xs, ys, zs, c='#aa3863')

ax.set_xlabel('retweets')
ax.set_ylabel('friends')
ax.set_zlabel('followers')


ax.azim = 55
ax.elev = 35


plt.show()
```

This shows that the vast majority of users that recieved less than 10 retweets, I removed and users which recieved over 100 retweets as this is likely just to of been jiscs account. It also shows the accounts with more friends and followers have more retweets as all of the accounts with more than 25 retweets also had a good number of friends and followers.