

VIS - Global Practice (InfoVis)
Visualization and Interaction Systems

Carlos García

June 2017

Contents

1 Project description	3
1.1 Context	3
1.2 What do you have to do?	3
2 First decisions	4
3 Data collection	5
4 Filtering the data	8
5 Analyzing the information	12
5.1 Importing all the libraries	13
5.2 Finding the most influential users per topic	13
5.3 Plotting the degree distribution	15
5.4 Drawing the graph	16
5.5 Plot the evolution of topics over the years	18
5.6 Scatter plot comparison	21
6 Extracting some knowledge	24
6.1 Finding the most influential people per topic	24
6.2 Comparing the degree distribution in networks	33
6.3 Comparing the evolution of topics over time	34
6.4 Comparing the global relevance of topics in scatter plots	39
7 Possible improvements	41
8 Conclusions	43

1 Project description

1.1 Context

This activity consists in generating a visualization using one of the dataset available in this page: <https://snap.stanford.edu/data/>. For this part you will need to (1) get the data that you are interested from the open data portal, (2) decide a visualization that has the requirements described in the following section, and (3) implement this visualization with the use of a library.

1.2 What do you have to do?

Design a visualization using the technology that you want (we recommend the use of javascript+d3) that includes ALL the following elements:

- A visualization of a network
- A temporal visualization
- Various qualitative/quantitative description charts of the dataset: Pie charts, bar diagrams, scatter plots, etc.
- Interaction with some of the elements of the visualization

2 First decisions

When we check trending in the last years about data analysis, data processing and data visualization, one tool is in almost every post, this tool is the Jupyter Notebook. It became famous because it's powerful, dynamic, interactive (specially for programmers) and easy to use compared to other platforms. All this advantages come thanks to combining an excellent programming language like Python with the visualization tools inside the browsers.

"The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more." [1]

Another great thing about this application, is that, thanks to its modularity, it can be easily connected to other programming languages and libraries, this way we can achieve the recommendation made in the description section about using javascript+d3, for this we simply install and import the library mpld3 [2].

Now that we have selected our working environment, we needed to find good resources to learn how to use them, and fortunately there are tons of information out in the internet, but mainly I focused my attention in two resources:

1. A very complete book with code samples called "Learning IPython for Interactive Computing and Data Visualization" [3].
2. A gallery of interesting Jupyter Notebooks, this is published on Github and there we can find good information and examples [4].

This both tools (among others) were used during all the stages of this project.

3 Data collection

As we can see in the task description, the idea was selecting one dataset from the Stanford Large Network Dataset Collection [5], in this collection we have several options depending on the type of data we prefer to deal with. This data is separated in the following categories:

Social networks: online social networks, edges represent interactions between people

Networks with ground-truth communities: ground-truth network communities in social and information networks

Communication networks: email communication networks with edges representing communication

Citation networks: nodes represent papers, edges represent citations

Collaboration networks: nodes represent scientists, edges represent collaborations (co-authoring a paper)

Web graphs: nodes represent webpages and edges are hyperlinks

Amazon networks: nodes represent products and edges link commonly co-purchased products

Internet networks: nodes represent computers and edges communication

Road networks: nodes represent intersections and edges roads connecting the intersections

Autonomous systems: graphs of the internet

Signed networks: networks with positive and negative edges (friend/foe, trust/dis-trust)

Location-based online social networks: Social networks with geographic check-ins

Wikipedia networks, articles, and metadata: Talk, editing, voting, and article data from Wikipedia

Temporal networks: networks where edges have timestamps

Twitter and Memetracker: Memetracker phrases, links and 467 million Tweets

Online communities: Data from online communities such as Reddit and Flickr

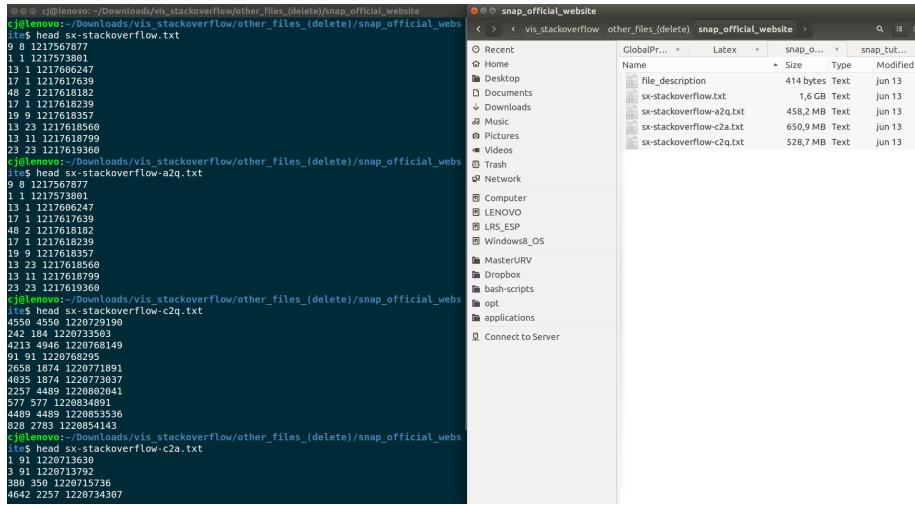
Online reviews: Data from online review systems such as BeerAdvocate and Amazon

We found “Temporal networks” as a good choice because this way we’d be able to check the evolution of the data along the time, the options for this category are the following:

⌚ Temporal networks

Name	Type	Nodes	Temporal Edges	Static Edges	Description
sx-stackoverflow	Directed, Temporal	2,601,977	63,497,050	36,233,450	Comments, questions, and answers on Stack Overflow
sx-mathoverflow	Directed, Temporal	24,818	506,550	239,978	Comments, questions, and answers on Math Overflow
sx-superuser	Directed, Temporal	194,085	1,443,339	924,886	Comments, questions, and answers on Super User
sx-askubuntu	Directed, Temporal	159,316	964,437	596,933	Comments, questions, and answers on Ask Ubuntu
wiki-talk-temporal	Directed, Temporal	1,140,149	7,833,140	3,309,592	Users editing talk pages on Wikipedia
email-Eu-core-temporal	Directed, Temporal	986	332,334	24,929	E-mails between users at a research institution
CollegeMsg	Directed, Temporal	1,899	20,296	59,835	Messages on a Facebook-like platform at UC-Irvine

As “Stack Overflow” is a very well-known collaboration network for software developers, we think we could extract some good information from this dataset, then now we proceed to download and check the data:

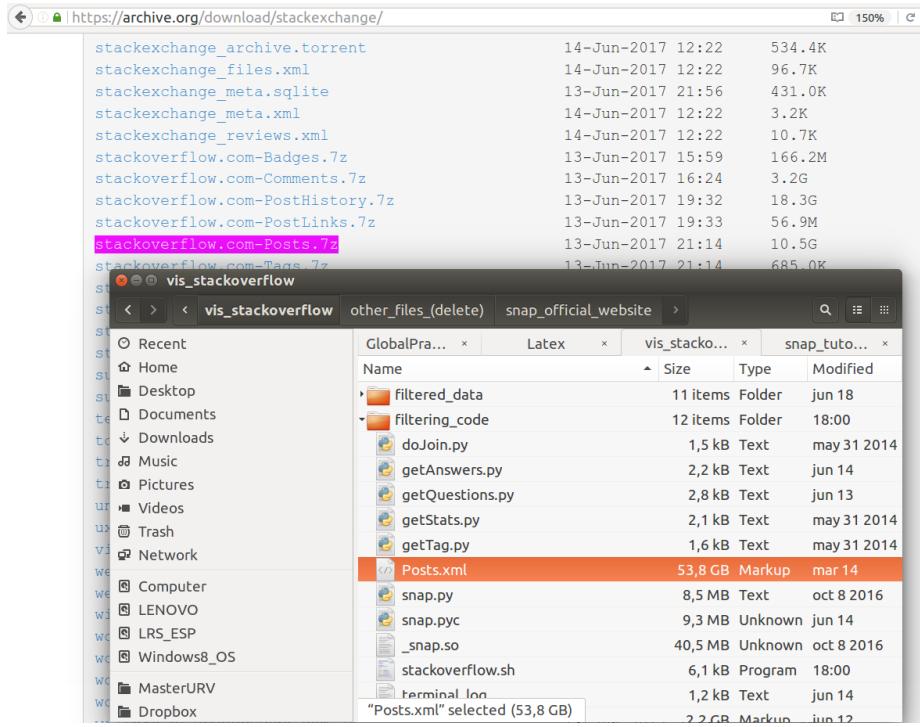


```

c:\lenovo\Downloads\vis_stackoverflow\other_files_(delete)\snap_official_webs
1s$ head sx-stackoverflow.txt
9 8 1217573801
1 1 1217573807
13 1 1217617357
17 1 1217617639
48 2 1217618182
17 1 1217618239
19 9 1217618357
13 23 1217618560
13 11 1217618799
23 23 1217619360
c:\lenovo\Downloads\vis_stackoverflow\other_files_(delete)\snap_official_webs
1s$ head sx-stackoverflow-a2q.txt
9 8 1217573877
1 1 1217606247
13 1 1217617639
48 2 1217618182
17 1 1217618239
19 9 1217618357
13 23 1217618560
13 11 1217618799
23 23 1217619360
c:\lenovo\Downloads\vis_stackoverflow\other_files_(delete)\snap_official_webs
1s$ head sx-stackoverflow-c2q.txt
490 45 1220713598
242 184 1220733593
4213 4946 1220768149
91 91 1220768295
2658 1874 123071891
4035 4489 1220802841
723 4489 1220802841
577 577 1220834891
4489 4489 1220853536
828 2783 1220854143
c:\lenovo\Downloads\vis_stackoverflow\other_files_(delete)\snap_official_webs
1s$ head sx-stackoverflow-c2a.txt
1 91 1220713630
3 91 1220713792
380 390 1220715736
4642 2257 1220734387

```

As we show in the image, after checking the downloaded packages, none of the files included a datetime column, this is probably because this data is already filtered from the original to reduce the size, but as we’d rather having the datetime stamps as well, then, we go to the main directory listing in stackexchange [6].



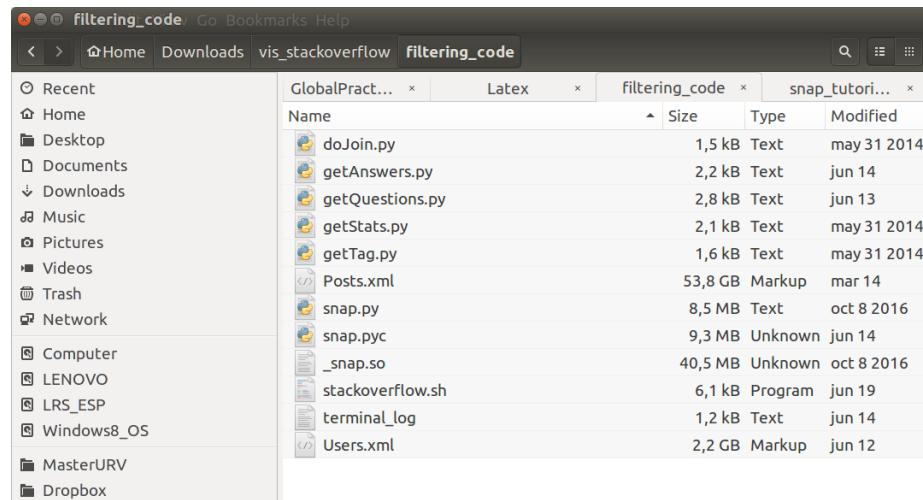
The compressed file is about 10.5 GB, and once we uncompress it increases to about 53.8 GB, this is obviously a huge amount of data to be analyzed directly from the main dataset, that's why we decided performing an initial filtering process where we locate and separate the posts according to the topics we are interested in.

The next section is about how to perform this data filtering process and what tools are used for this purpose.

4 Filtering the data

In order to help us with the process of extracting information from the datasets, the Stanford guys created some Python template scripts which goes over the big xml file and find the posts we are looking for by using a keyword (it is important to know that this libraries are developed to be used with Python 2.x, not Python 3). In our case we filtered using the tags for certain topics we think are worth learning more details about them, this topics are the following: Python, Java, Android, C++, MongoDB, Linux, Arduino, ROS (Robot Operating System), Raspberry-pi and Apache-Spark.

The Python scripts mentioned are the following:



A screenshot of a file manager window titled "filtering_code". The window shows a sidebar with "Recent" items like Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Trash, Network, Computer, LENOVO, LRS_ESP, Windows8_OS, MasterURV, and Dropbox. The main area displays a list of files in the "filtering_code" folder. The files are:

Name	Size	Type	Modified
doJoin.py	1,5 kB	Text	may 31 2014
getAnswers.py	2,2 kB	Text	jun 14
getQuestions.py	2,8 kB	Text	jun 13
getStats.py	2,1 kB	Text	may 31 2014
getTag.py	1,6 kB	Text	may 31 2014
Posts.xml	53,8 GB	Markup	mar 14
snap.py	8,5 MB	Text	oct 8 2016
snap.pyc	9,3 MB	Unknown	jun 14
_snap.so	40,5 MB	Unknown	oct 8 2016
stackoverflow.sh	6,1 kB	Program	jun 19
terminal_log	1,2 kB	Text	jun 14
Users.xml	2,2 GB	Markup	jun 12

They can be found in the tutorial exercises [7].

To use this scripts we can simply execute the file "stackoverflow.sh" and it will perform each step automatically:

```

11# requirements:
12#   python installed
13#   Snap.py installed, http://snap.stanford.edu
14#   Posts.xml, https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z, uncompress
15
16echo `date` ... "START analysis of StackOverflow"
17
18# get all the question posts and accepted answers (15min)
19echo `date` ... "extracting questions ..."
20python getQuestions.py Posts.xml > questions.txt
21
22# get all the answer posts (15min)
23echo `date` ... "extracting answers ..."
24python getAnswers.py Posts.xml > answers.txt
25
26# get all the Java question posts, id only (15min)
27echo `date` ... "identifying Java questions ..."
28python getTag.py Posts.xml java > java.txt
29
30# select questions with a Java tag (20s)
31echo `date` ... "selecting Java questions ..."
32python doJoin.py ../www15-data/java.txt ../www15-data/questions.txt 1 1 > java-posts.txt
33
34# identify users of accepted answers (40s)
35echo `date` ... "finding owners of accepted answers ..."
36python doJoin.py ../www15-data/answers.txt java-posts.txt 1 3 > qa.txt
37
38# create a graph and find top users (5s)
39echo `date` ... "building a QA graph and calculating statistics ..."
40python getStats.py qa.txt 2 6 > stats.txt
41
42echo `date` ... "results are stored in stats.txt"
43echo `date` ... "END analysis of StackOverflow"

```

In the above image, it is being executed for Java posts, but we can change it depending on our choice, in our case we used this bash script:

```

16echo `date` ... "START analysis of StackOverflow"
17# get all the question posts and accepted answers (15min)
18echo `date` ... "extracting questions ..."
19python getQuestions.py Posts.xml > questions.txt
20
21# get all the answer posts (15min)
22echo `date` ... "extracting answers ..."
23python getAnswers.py Posts.xml > answers.txt
24
25# get all the Java question posts, id only (15min)
26echo `date` ... "identifying Java questions ..."
27python getTag.py Posts.xml java > java.txt
28echo `date` ... "identifying Python questions ..."
29python getTag.py Posts.xml python > python.txt
30echo `date` ... "identifying C++ questions ..."
31python getTag.py Posts.xml c++ > c++.txt
32echo `date` ... "identifying Raspberry questions ..."
33python getTag.py Posts.xml raspberry-pi > raspberry-pi.txt
34echo `date` ... "identifying Arduino questions ..."
35python getTag.py Posts.xml arduino > arduino.txt
36echo `date` ... "identifying Ros questions ..."
37python getTag.py Posts.xml ros > ros.txt
38echo `date` ... "identifying MongoDB questions ..."
39python getTag.py Posts.xml mongodb > mongodb.txt
40echo `date` ... "identifying Android questions ..."
41python getTag.py Posts.xml android > android.txt
42echo `date` ... "identifying Linux questions ..."
43python getTag.py Posts.xml linux > linux.txt
44echo `date` ... "identifying Spark questions ..."
45python getTag.py Posts.xml apache-spark > spark.txt
46
47# select questions with a Java tag (20s)
48echo `date` ... "selecting Java questions ..."
49python doJoin.py java.txt questions.txt 1 1 > java-posts.txt
50echo `date` ... "selecting Python questions ..."
51python doJoin.py python.txt questions.txt 1 1 > python-posts.txt
52echo `date` ... "selecting C++ questions ..."

```

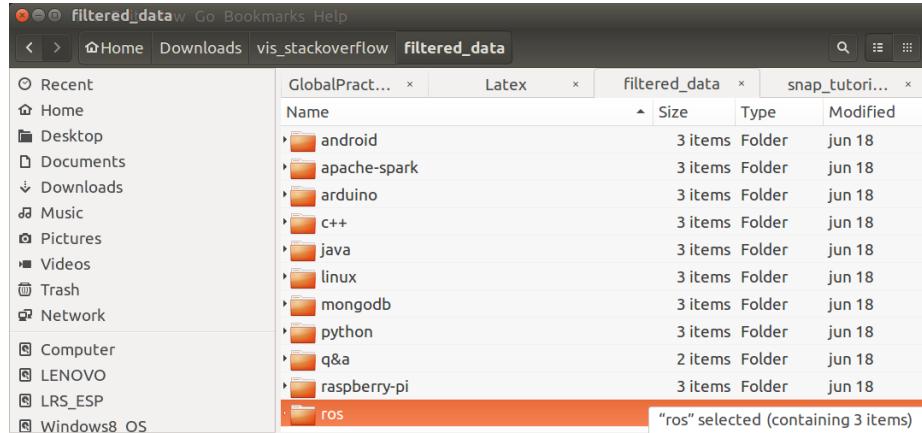
That way we perform each operation for each topic one after the other. At the end we obtain the most important file where the script combines all the information from the question along all the information from the accepted answer:

```

70# identify users of accepted answers (40s)
71 echo `date` ... "finding owners of accepted answers ..."
72 python doJoin.py answers.txt java-posts.txt 1 3 > qa-java.txt
73 echo `date` ... "finding owners of accepted answers ..."
74 python doJoin.py answers.txt python-posts.txt 1 3 > qa-python.txt
75 echo `date` ... "finding owners of accepted answers ..."
76 python doJoin.py answers.txt c++-posts.txt 1 3 > qa-c++.txt
77 echo `date` ... "finding owners of accepted answers ..."
78 python doJoin.py answers.txt raspberry-pi-posts.txt 1 3 > qa-raspberry-pi.txt
79 echo `date` ... "finding owners of accepted answers ..."
80 python doJoin.py answers.txt arduino-posts.txt 1 3 > qa-arduino.txt
81 echo `date` ... "finding owners of accepted answers ..."
82 python doJoin.py answers.txt ros-posts.txt 1 3 > qa-ros.txt
83 echo `date` ... "finding owners of accepted answers ..."
84 python doJoin.py answers.txt mongodb-posts.txt 1 3 > qa-mongodb.txt
85 echo `date` ... "finding owners of accepted answers ..."
86 python doJoin.py answers.txt android-posts.txt 1 3 > qa-android.txt
87 echo `date` ... "finding owners of accepted answers ..."
88 python doJoin.py answers.txt linux-posts.txt 1 3 > qa-linux.txt
89 echo `date` ... "finding owners of accepted answers ..."
90 python doJoin.py '../filtered_data/q&a/answers.txt' spark-posts.txt 1 3 > qa-spark.txt

```

And we organized the outputs from the script in a directory like this:



filtered_data

Name	Size	Type	Modified
android	3 items	Folder	jun 18
android.txt	8,5 MB	Text	jun 14
android-posts.txt	95,9 MB	Text	jun 14
qa-android.txt	78,2 MB	Text	jun 14
apache-spark	3 items	Folder	jun 18
apache-spark.txt	220,1 kB	Text	jun 18
apache-spark-posts.txt	2,4 MB	Text	jun 18
qa-apache-spark.txt	1,8 MB	Text	jun 18
arduino	3 items	Folder	jun 18
arduino.txt	92,0 kB	Text	jun 14
arduino-posts.txt	1,0 MB	Text	jun 14
qa-arduino.txt	678,8 kB	Text	jun 14
c++	3 items	Folder	jun 18
c++.txt	4,4 MB	Text	jun 14
c++-posts.txt	49,9 MB	Text	jun 14
qa-c++.txt	53,7 MB	Text	jun 14
java	3 items	Folder	jun 18
java.txt	10,8 MB	Text	jun 14
java-posts.txt	121,9 MB	Text	jun 14
qa-java.txt	113,5 MB	Text	jun 14
linux	3 items	Folder	jun 18
linux.txt	1,2 MB	Text	jun 14
linux-posts.txt	14,1 MB	Text	jun 14
qa-linux.txt	13,1 MB	Text	jun 14
mongodb	3 items	Folder	jun 18
mongodb.txt	702,1 kB	Text	jun 14
mongodb-posts.txt	7,9 MB	Text	jun 14
qa-mongodb.txt	7,1 MB	Text	jun 14

"android" selected (containing 3 items)

At this moment, we will focus our attention in the files "qa-topic.txt", they are placed in the folder with the same name of the topic, and this files contain all the important information related to one post, including the data about the question and the data about the accepted answer for that question.

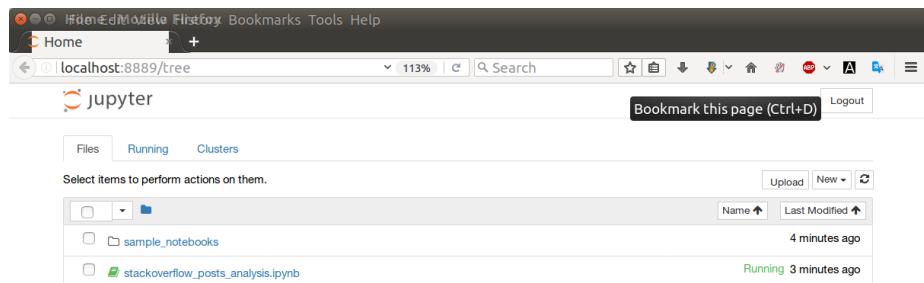
5 Analyzing the information

To analyze the data, the idea is start by running the jupyter notebook, to do this we go to the terminal, change the directory and run the corresponding command (it is recommended using a virtual environment specially the ones using the conda package manager):

```
cj@lenovo: ~/Dropbox/Personal/Stud..._Work
(venv_conda) cj@lenovo:~$ cd '/home/cj/Dropbox/Personal/Stud.../2nd_Semester/VIS/Lab_Sessions_and_Practical_Wor...
(venv_conda) cj@lenovo:~/Dropbox/Personal/Stud.../2nd_Semester/VIS/Lab_Sessions_and_Practical_Wor...
[I 10:23:37.076 NotebookApp] The port 8888 is already in use, trying another port.
[I 10:23:37.084 NotebookApp] Serving notebooks from local directory: /home/cj/Dropbox/Personal/Stud...
[I 10:23:37.084 NotebookApp] jupyter notebook
[I 10:23:37.084 NotebookApp] jupyter notebook
[I 10:23:37.084 NotebookApp] 0 active kernels
[I 10:23:37.084 NotebookApp] The Jupyter Notebook is running at: http://localhost:888...
[I 10:23:37.084 NotebookApp] Use Control-C to stop this server and shut down all kern...
els (twice to skip confirmation).
[C 10:23:37.084 NotebookApp]

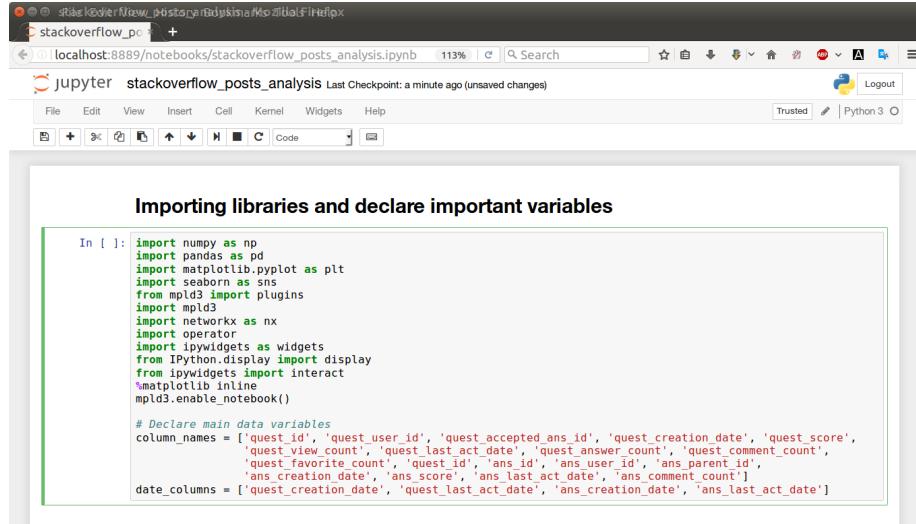
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8889/?token=4be347aefcc2e2af7046626e92dd1e53dc01bc7d87ad914
[I 10:23:37.202 NotebookApp] Accepting one-time-token-authenticated connection from 1
27.0.0.1
[I 10:23:48.070 NotebookApp] Kernel started: 17385c86-6775-4229-b7a5-5f8428e7d009
[I 10:27:48.099 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 10:29:48.066 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 10:33:48.064 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 10:35:48.060 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 11:11:48.099 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 11:13:48.061 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
[I 11:15:48.065 NotebookApp] Saving file at /stackoverflow_posts_analysis.ipynb
```

Once there, we create a new notebook and start performing the analysis in independent blocks of code:



5.1 Importing all the libraries

This block is simply to tell the notebook about the libraries will be using during the analysis (there is no output for this block):



The screenshot shows a Jupyter Notebook interface with the title "Importing libraries and declare important variables". The code cell contains Python code for importing various libraries and defining main data variables.

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpls import plugins
import mpf3
import networkx as nx
import operator
import ipywidgets as widgets
from IPython.display import display
from ipywidgets import interact
%matplotlib inline
mpld3.enable_notebook()

# Declare main data variables
column_names = ['quest_id', 'quest_user_id', 'quest_accepted_ans_id', 'quest_creation_date', 'quest_score',
                'quest_view_count', 'quest_last_act_date', 'quest_answer_count', 'quest_comment_count',
                'quest_favorite_count', 'quest_id', 'ans_id', 'ans_user_id', 'ans_parent_id',
                'ans_creation_date', 'ans_score', 'ans_last_act_date', 'ans_comment_count']
date_columns = ['quest_creation_date', 'quest_last_act_date', 'ans_creation_date', 'ans_last_act_date']
```

5.2 Finding the most influential users per topic

This block creates a graph (using networkx library) where the nodes are the users and the edges are links between the user who asks a question and the user who creates the accepted answer for that question. After having the graph we compute the degree for each node and finally we use this degrees to create a pandas DataFrame and order the rank. At the end we plot the 10 most influential users for the selected topic.

```

In [3]: @interact
def top_users(topic=['ros', 'android', 'apache-spark', 'arduino', 'c++', 'java', 'linux', 'mongodb',
                     'python', 'raspberry-pi']):
    qa = pd.read_csv('~/Downloads/vis_stackoverflow/filtered_data/' + topic + '/qa-' + topic + '.txt', '\t',
                      names=column_names, parse_dates=date_columns)
    G = nx.from_pandas_dataframe(qa[['quest_user_id', 'ans_user_id']], 'quest_user_id', 'ans_user_id')
    # Computing the degree for every node
    deg = nx.degree(G)
    df = pd.DataFrame.from_dict(deg, orient='index')
    df.columns = ['answered questions']
    # Ordering to find the most influential users
    df = df.sort_values(by='answered questions', ascending=False)
    print(topic + " results, most influential users are: \n")
    print(df.head(10))

topic ros

```

ros results, most influential users are:

	answered questions
2095383	25
5836291	13
1563315	10
943472	8
6410520	6
5146563	6
1559401	4
4050550	4
6859348	4
5811411	3

Note that this process is interactive and we can select the topic we prefer by using the dropdown list.

```

In [3]: @interact
def top_users(topic=['ros', 'android', 'apache-spark', 'arduino', 'c++', 'java', 'linux', 'mongodb',
                     'python', 'raspberry-pi']):
    qa = pd.read_csv('~/Downloads/vis_stackoverflow/filtered_data/' + topic + '/qa-' + topic + '.txt', '\t',
                      names=column_names, parse_dates=date_columns)
    G = nx.from_pandas_dataframe(qa[['quest_user_id', 'ans_user_id']], 'quest_user_id', 'ans_user_id')
    # Computing the degree for every node
    deg = nx.degree(G)
    df = pd.DataFrame.from_dict(deg, orient='index')
    df.columns = ['answered questions']
    # Ordering to find the most influential users
    df = df.sort_values(by='answered questions', ascending=False)
    print(topic + " results, most influential users are: \n")
    print(df.head(10))

topic ros

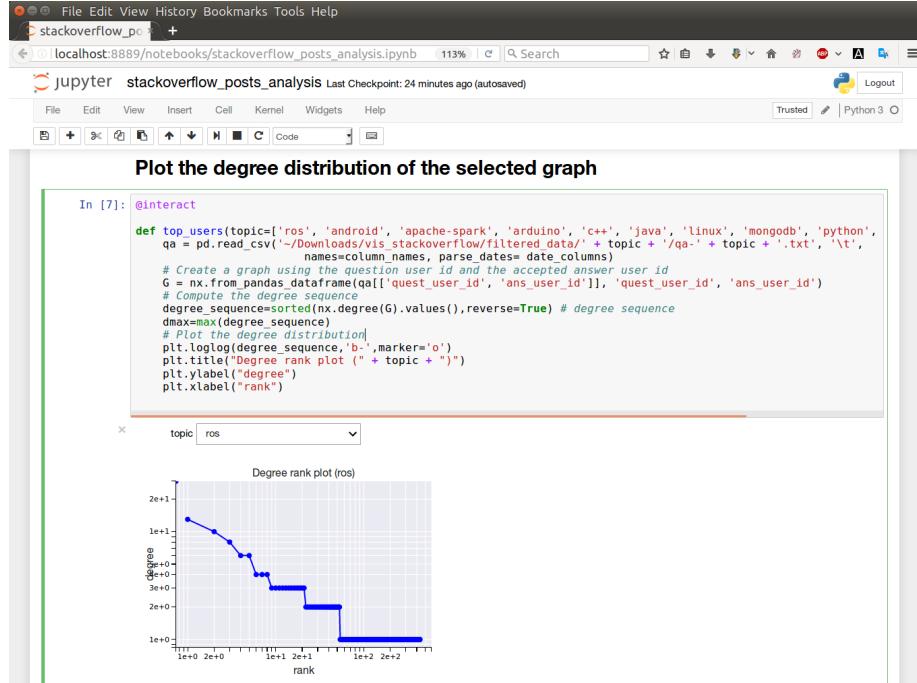
```

ros results, most influential users are:

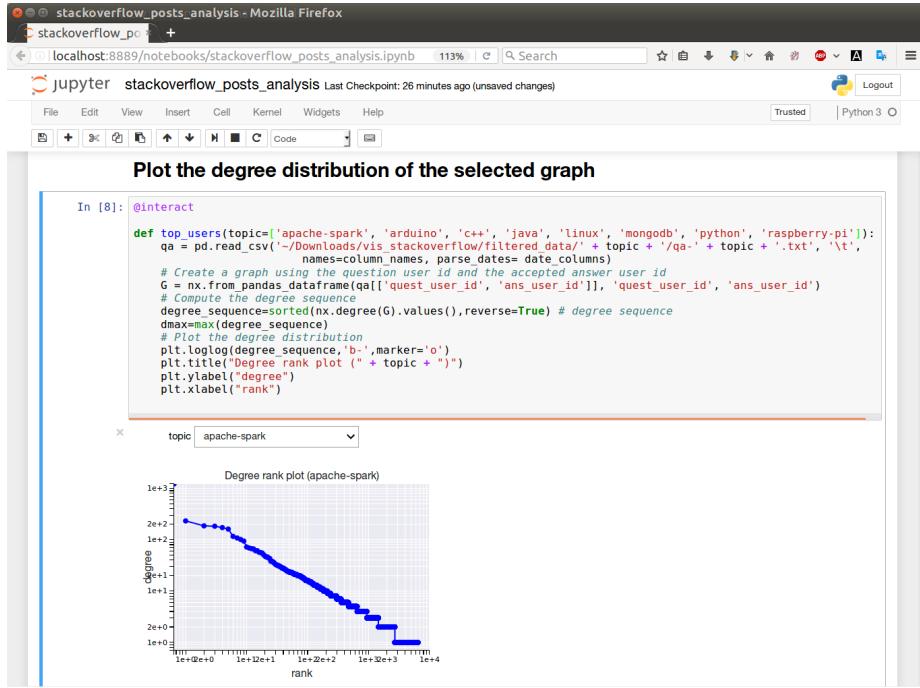
	answered questions
2095383	25
5836291	13
1563315	10
943472	8
6410520	6
5146563	6
1559401	4
4050550	4
6859348	4
5811411	3

5.3 Plotting the degree distribution

In this block we plot the degree distribution of the graph for an specific topic, “In the study of graphs and networks, the degree of a node in a network is the number of connections it has to other nodes and the degree distribution is the probability distribution of these degrees over the whole network” [8].

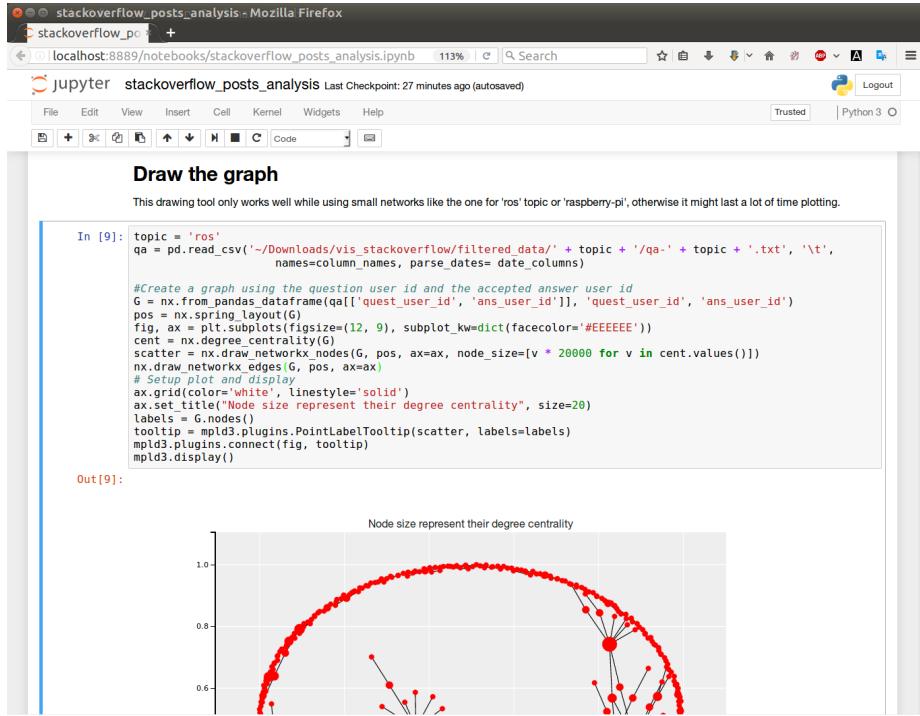


This block is also interactive and we can change the topic simply by clicking over the dropdown list or by changing the list in the code, however we prefer:

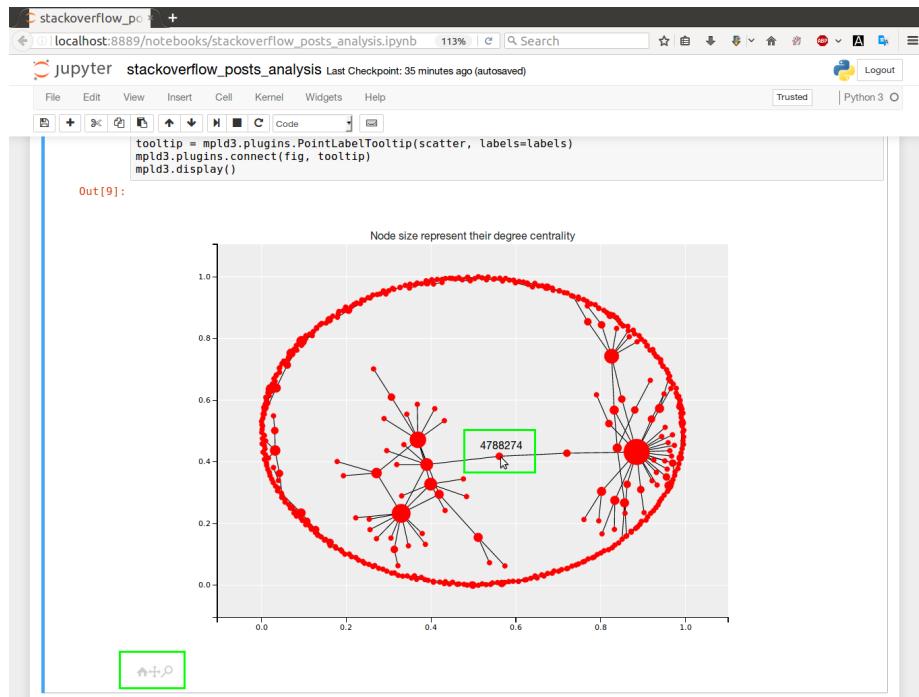


5.4 Drawing the graph

Here we make use of the networkx library and the mplot3 utilities [2] to draw the graph showing the links between users, additionally we compute the degree centrality [9] of the nodes and make another rank to check the most influential users per topic. By drawing the graph we have a more didactic and interactive view of the network, this way we can visually identify the most important nodes.



Thanks to the mpld3 library [2], we can easily interact with the plot by hovering the mouse cursor over the nodes, or by using the tool buttons (home, pan and zoom) placed in the left bottom corner of the image:



Note that this block should only be tested with small networks, otherwise it last a very long time compiling and drawing the network.

5.5 Plot the evolution of topics over the years

In this block we group the data per month and then we count the posts per topic and make the plot according to this number, additionally, we sum (also per month) the score of the questions per topic and represent this value as the width of the line, this width uses a transparency property to avoid mixing with the other information.

File Edit View History Bookmarks Tools Help
 stackoverflow_pc +
 localhost:8889/notebooks/stackoverflow_posts_analysis.ipynb 113% C Search Logout
 jupyter stackoverflow_posts_analysis Last Checkpoint: an hour ago (autosaved)
 File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

Plot the evolution of the topics over the years

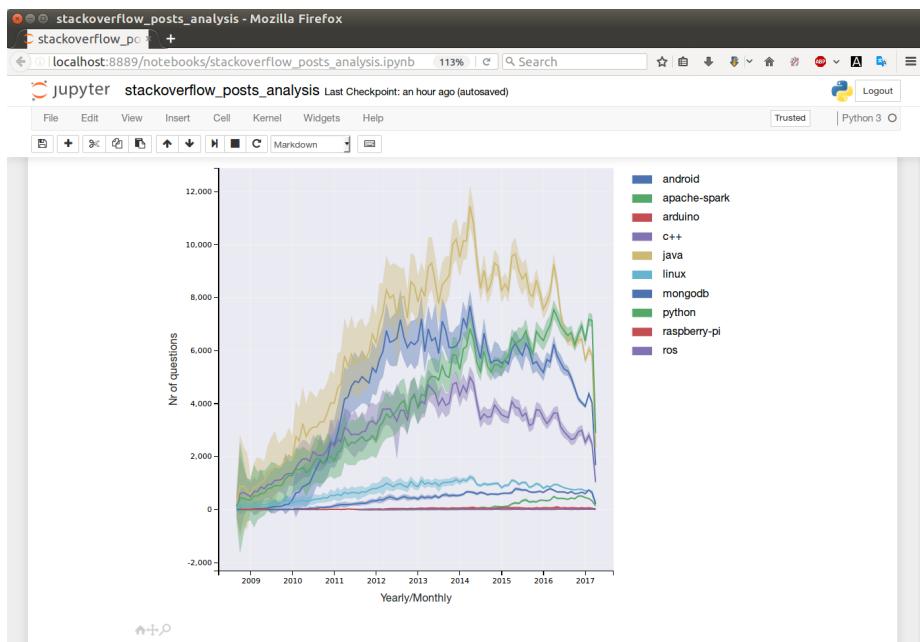
```
In [10]: # Create new figure
fig, ax = plt.subplots(figsize=(9, 7), dpi=100)
fig.subplots_adjust(right=0.7)
ax.grid(True, alpha=0.3)

# Loop over array of topics
# topics = ['ros', 'arduino', 'apache-spark'] # smallest sets
topics = ['android', 'apache-spark', 'arduino', 'c++', 'java', 'linux', 'mongodb', 'python', 'raspberry-pi', 'ros']

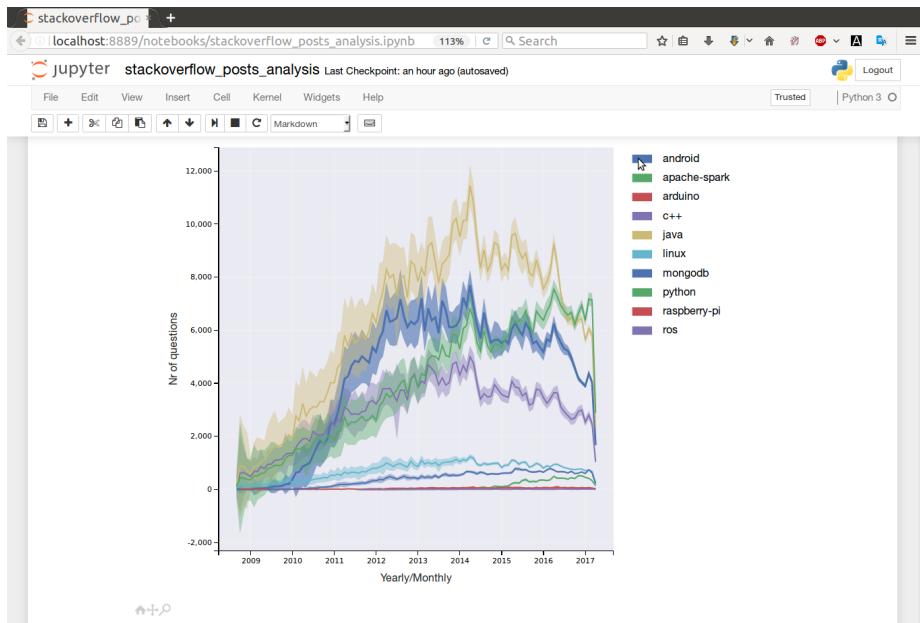
for topic in topics:
    # Load data from files
    qa = pd.read_csv('~/Downloads/vis_stackoverflow/filtered_data/' + topic + '/qa-' + topic + '.txt', '\t',
                      names=column_names, parse_dates=date_columns)
    monthly_count = qa.resample('M', on='quest_creation_date').count()
    monthly_sum = qa[['quest_creation_date', 'quest_score']].resample('M', on='quest_creation_date').sum().fillna(0)
    # Plot data using the same figure
    l, = ax.plot(monthly_count.index, monthly_count['quest_id'], label=topic)
    ax.fill_between(monthly_count.index, monthly_count['quest_id'] - (monthly_sum['quest_score'] / 20),
                    monthly_count['quest_id'] + (monthly_sum['quest_score'] / 20),
                    color=l.get_color(), alpha=.4)

# Define interactive legend
handles, labels = ax.get_legend_handles_labels() # return lines and labels
interactive_legend = plugins.InteractiveLegendPlugin(zip(handles, ax.collections), labels, alpha_unsel=0.5,
                                                    alpha_over=1.5, start_visible=True)
plugins.connect(fig, interactive_legend)
# Setup the axis and display the plot
ax.set_xlabel('Yearly/Monthly')
ax.set_ylabel('Nr of questions')
mpld3.display()

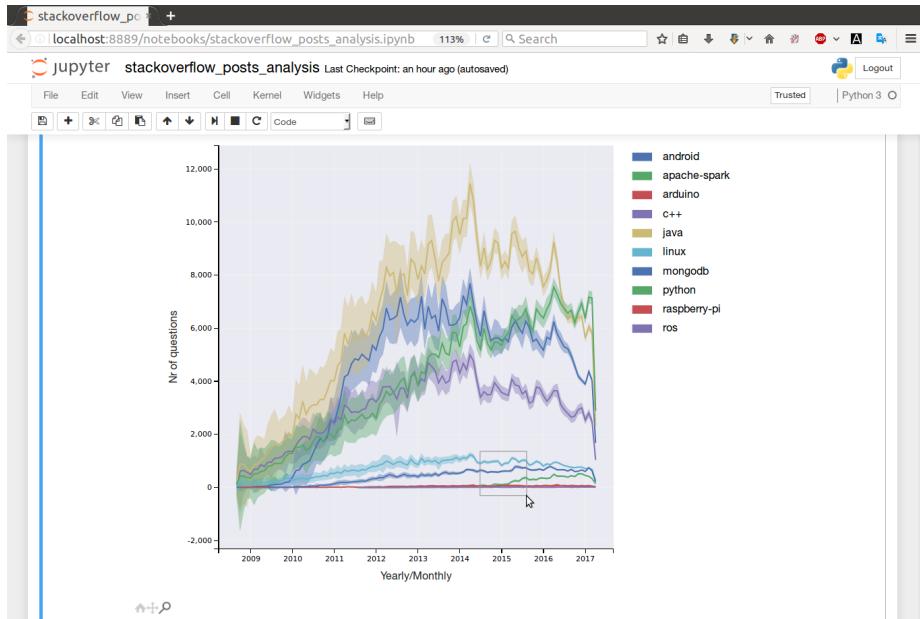
Out[10]:
```



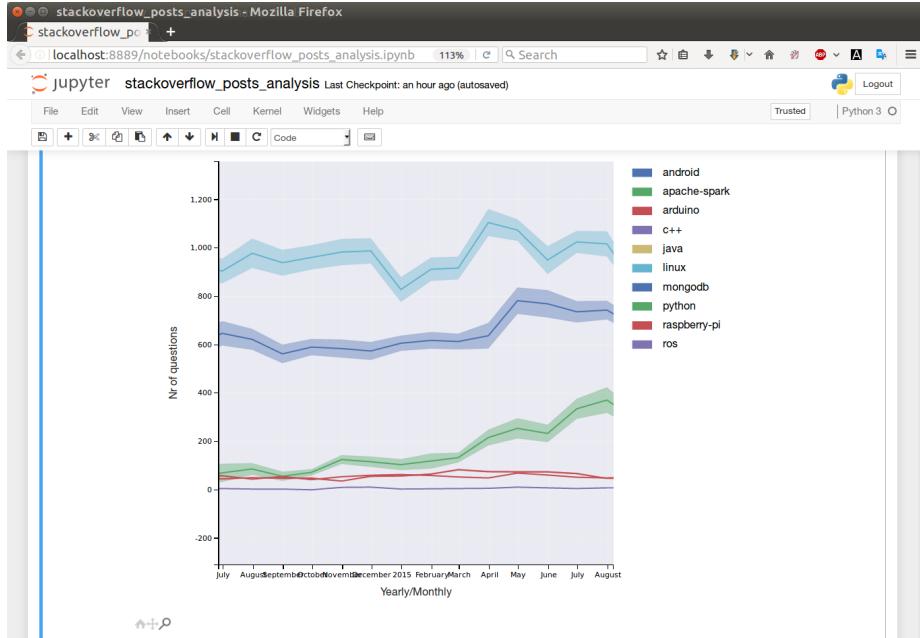
In this interactive plot we can highlight one of the topics simply by placing the mouse over its name in the legend:



For this plot is almost mandatory using the zooming and panning tools to check the details in the plot (note that the axis also changes according to the zoom, to indicate the specific month we are observing), for example to check the ROS evolution we zoom over the lowest part in the plot:



Now the plot is showing a lot more details for the lowest curves in the plot (including also the axis):



5.6 Scatter plot comparison

In this block we create a scatter plot trying to compare certain global properties of the topics, this properties are:

- Global question score (represented in the horizontal axis)
- Global answer score (represented in the vertical axis)
- Global question count (represented in the size of the points)

This is the code in the jupyter notebook:

```

In [11]: # Create new figure
fig, ax = plt.subplots(figsize=(9, 7), dpi=100, subplot_kw=dict(facecolor="#EEEEEE"))
labels = []; xs = []; ys = []; sizes = []
# fig.subplots_adjust(right=0.7)
ax.grid(True, alpha=0.3)

# Loop over array of topics
topics = ['android', 'apache-spark', 'arduino', 'c++', 'java', 'linux', 'mongodb', 'python', 'raspberry-pi', 'r']
for topic in topics:
    # Load data from file
    qa = pd.read_csv(f'Downloads/vis/stackoverflow/finalized_data/{topic}/qa-{topic}.txt', '\t',
                      names=['question_id', 'score'], parse_dates=['date'])
    # Append new values to arrays used in scatter plot
    total_count = qa['question_id'].count()
    total_sum = qa['score'].sum()
    xs.append(total_sum['quest_score'])
    ys.append(total_sum['ans_score'])
    sizes.append(total_count['quest_id'] / 100)
    labels.append(topic + f' (total questions = {total_count["quest_id"]})')

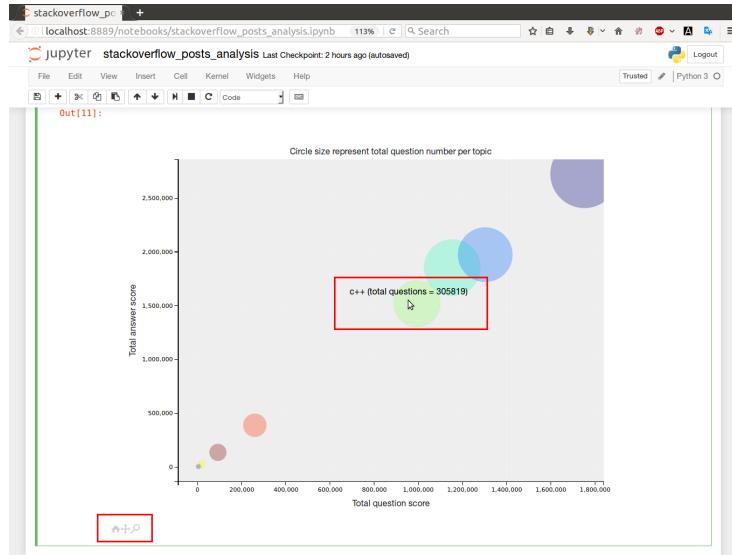
# Create scatter plot
scatter = ax.scatter(xs,
                     ys,
                     c=np.random.random(size=len(topics)),
                     s=sizes,
                     alpha=0.3,
                     cmap=plt.cm.get_cmap('jet'))

# Setup plot and display
ax.set_color_cycle(['lightblue', 'lightgreen', 'lightred', 'lightpurple'])
ax.set_xlabel('Total question score')
ax.set_ylabel('Total answer score')
ax.set_title('Circle size represent total question number per topic')
ax.set_xlim(0, 1800000)
ax.set_ylim(0, 2500000)
tooltip = mpd3.plugins.PointLabelTooltip(scatter, labels=labels)
mpd3.plugins.connect(fig, tooltip)
mpd3.display()

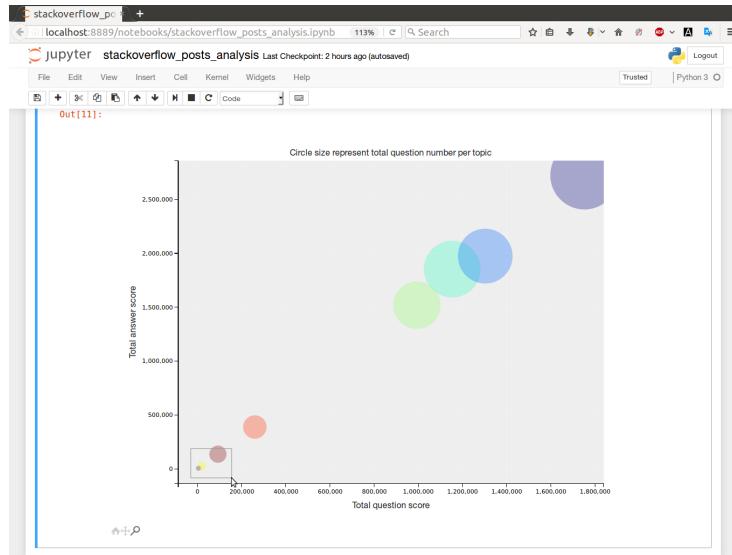
```

Out[11]:

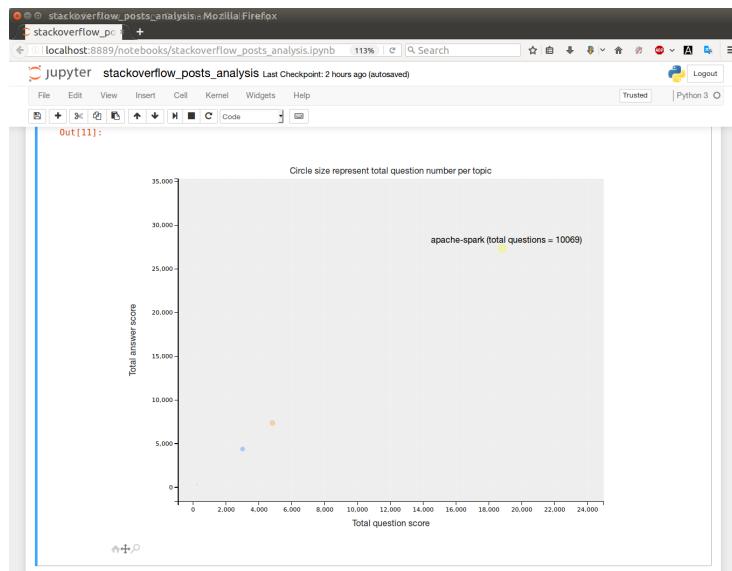
And this is the resulting output, note that this plot is also interactive, we can hover the cursor over the points to check the total number of questions related to this topic:



Additionally we should make use of the zooming and panning tool to check the differences in the lowest part of the graph:



Once there we can check the post counting values for the least influential or newest topics like ROS, Raspberry-pi, Arduino and Apache-Spark:



6 Extracting some knowledge

Once we have filtered and visualized the data, it is impossible not getting some extra information and conclusions, we think this are some of the most interesting:

6.1 Finding the most influential people per topic

If we execute the second block from our jupyter notebook and go over each topic the results are the following:

Python: Most influential user: 100297 with 9213 answered questions in this topic.

python results, most influential users are:	
	answered questions
100297	9213
_none	3898
190597	3330
771848	2959
104349	2594
908494	2213
2225682	2199
2141635	2022
20862	1870
95810	1656

We searched about this user and this is what we found:

The screenshot shows the Stack Overflow user profile for Martijn Pieters. At the top, there's a search bar with 'user:100297'. Below the search bar, there are tabs for 'Profile' (which is selected), 'Activity', and 'Developer Story'. The profile page features a large image of a black ninja-like character, Martijn's reputation of 562,603, and three badge counts: 87 (yellow), 1678 (grey), and 1727 (orange). To the right, Martijn's name is displayed in bold with a diamond moderator badge and a 'top 0.01% this quarter' badge. Below his name, it says 'Software Engineer at Facebook' and 'Invisible framework coding ninja'. A note mentions an 'Amazon Wishlist'. A bulleted list includes links to his 'Personal website', an 'Interview at Talk Python to Me', and 'Keybase.io'. At the bottom, it says 'SO badge firsts:' followed by a small, partially visible badge row.

Java: Most influential user: 22656 with 5044 answered questions in this topic.

topic ▾

java results, most influential users are:

	answered questions
none	6276
22656	5044
992484	3809
571407	3738
57695	3503
139985	3313
522444	3073
157882	2824
131872	2764
207421	2278

Questions Developer Jobs Documentation BETA Tags Users

Profile Activity



954,369 REPUTATION

561 7001 7804

Jon Skeet top 0.01% overall

Senior Software Engineer at Google

Author of [C# in Depth](#).
Currently a software engineer at Google, London.
Usually a Microsoft MVP (C#, 2003-2010, 2011-)

Sites:

- [C# in Depth](#)
- [Coding blog](#)
- [C# articles](#)
- [Twitter updates \(@jonskeet\)](#)
- [Google+ profile](#)

Android: Most influential user: 115145 with 8354 answered questions in this topic.

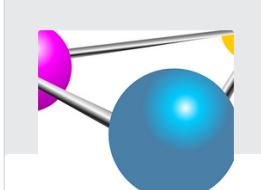
topic android

android results, most influential users are:

	answered questions
115145	8354
__none__	4563
501696	1505
1202025	1374
653856	1332
101361	1170
1631193	969
493939	948
1676363	844
1267661	834

Questions Developer Jobs Documentation BETA Tags Users

Profile Activity Developer Story



CommonsWare top 0.01% overall

Mark Murphy, founder of CommonsWare, is the author of [The Busy Coder's Guide to Android Development](#), the first, largest, and most up-to-date book on Android app programming, covering Android Studio and the latest Android SDKs. This book gets updated every ~8 weeks, with subscribers getting access to those updates. The latest update, June 2017's Version 8.6, added more coverage of Android O, as part of its 4,542 pages.

He also offers [office hours chats](#) and [training](#) on Android app development, in case you are interested in help beyond what fits the Stack Overflow model.

...

C++: Most influential user: 204847 with 1882 with 8354 answered questions in this topic.

topic c++

c++ results, most influential users are:

	answered questions
__none__	4281
204847	1882
596781	1746
440558	1573
179910	1558
85371	1557
661519	1489
560648	1371
673730	1283
2069064	1272

Profile **Activity**

Mike Seymour top 0.04% overall

I'm an experienced programmer, specialising in creating high performance software for Linux using C++. In the past, I've built audio processing engines for digital music players, and I'm currently working on real-time financial trading.

I live in London, UK.

197,092 REPUTATION

13 278 492

MongoDB: Most influential user: 1259510 with 1134 answered questions in this topic.

topic ▾

mongodb results, most influential users are:

	answered questions
1259510	1134
2313887	984
122005	935
5031275	555
383478	401
3100115	369
1388319	355
none	346
431012	324
1620671	253

Profile **Activity**

JohnnyHK top 0.05% this year

Full-stack software development consultant and contractor.

Tech I enjoy using in this pursuit:

- ASP.NET MVC
- Node.js
- SQL Server
- MongoDB
- Knockout
- AngularJS
- Kendo UI
- Bootstrap

Linux: Most influential user: 548225 with 589 answered questions in this topic.

topic ▼

linux results, most influential users are:

	answered questions
none	1098
548225	589
841108	546
20862	472
15168	455
1983854	328
1491895	305
134633	300
14122	298
50617	287

The screenshot shows a user profile for 'anubhava' on Stack Overflow. At the top, there's a navigation bar with links for Questions, Developer Jobs, Documentation (BETA), Tags, and Users. A search bar on the right contains the query 'user:548225'. Below the navigation, there are two tabs: 'Profile' (which is selected) and 'Activity'. The main profile area features a large thumbnail photo of a man, his name 'anubhava' in bold black text, and a badge indicating 'top 0.01% this year'. It also lists his title 'Lead Engineer at AOL Inc'. Below this, there's a summary section with '427,400 REPUTATION' and three colored buttons showing counts: yellow (35), grey (208), and orange (273). To the right of the profile summary, there's a bulleted list of links: 'My Linked-In Profile', 'On Twitter: Follow @anubhava', 'Works for AOL', 'My Blog', and '#SOreadytohelp'.

Arduino: Most influential user: 3368201 with 59 answered questions in this topic.

topic		arduino
arduino results, most influential users are:		
	answered questions	
3368201	59	
none	51	
1899801	47	
5271927	37	
1927972	37	
1290438	34	
20862	34	
4023	26	
840992	26	
4100891	26	

A screenshot of a user profile page. At the top, there's a navigation bar with links for Questions, Developer Jobs, Documentation (BETA), Tags, and Users. A search bar shows the query "user:3368201". Below the navigation, there are tabs for Profile (which is selected), Activity, and Developer Story. The main content area shows a large gray placeholder image, the user's name "frarugi87" with a "top 10% this year" badge, and a status message: "Apparently, this user prefers to keep an air of mystery about them." Below this, the user's reputation is listed as "1,983 REPUTATION" with three colored squares indicating activity levels: yellow (1), gray (8), and brown (27).

ROS (Robot Operating System): Most influential user: 2095383 with 25 answered questions in this topic.

A screenshot of a search results page. The search bar at the top has "topic" and "ros" entered. Below the search bar, it says "ros results, most influential users are:" followed by a table titled "answered questions".

	answered questions
2095383	25
5836291	13
1563315	10
943472	8
6410520	6
5146563	6
1559401	4
4050550	4
6859348	4
5811411	3

luator top 22% overall

Apparently, this user prefers to keep an air of mystery about them.

1,663 REPUTATION

1 10 26

Raspberry-pi: Most influential user: 11654 with 13 answered questions in this topic.

answered questions	
none	31
11654	13
541038	13
100297	13
504554	9
2587646	9
1461050	9
3207652	8
7432	8
89766	8

The screenshot shows a user profile page. At the top, there is a navigation bar with links for Questions, Developer Jobs, Documentation (BETA), Tags, and Users. A search bar contains the query "user:11654". Below the navigation bar, there are tabs for Profile (selected), Activity, and Developer Story. The main content area features a large empty placeholder box for a profile picture. To its right, the user's name "CL." is displayed in bold, with a badge indicating "top 0.09% this year". Below this, the user's reputation is shown as "109,012 REPUTATION" in bold. Underneath the reputation, three colored boxes show the user's participation counts: 11 gold, 76 silver, and 113 bronze.

Apache-Spark: Most influential user: 1560062 with 870 answered questions in this topic.

topic		apache-spark
apache-spark results, most influential users are:		
answered questions		
1560062		870
6910411		232
3415409		186
764040		183
none		172
5344058		161
779513		116
3318517		108
1870803		101
572083		94

Profile Activity

zero323 top 0.03% this year

I would love to tell you, but then, of course, I'd have to erase your memory.

[codementor](#) [CONTACT ME ▶](#)

105,495 REPUTATION

23 219 306

6.2 Comparing the degree distribution in networks

By comparing the degree distribution in a couple of networks:

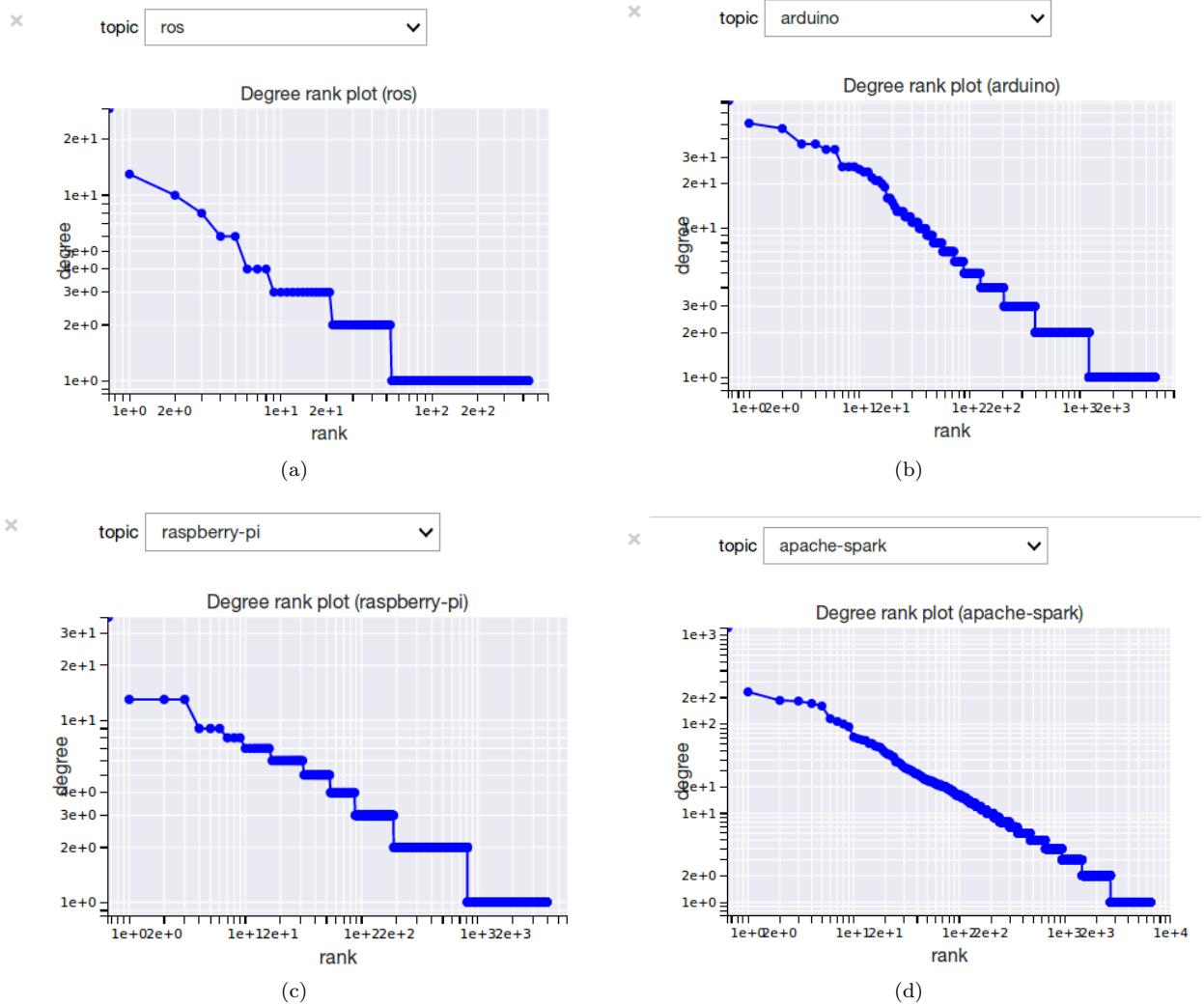
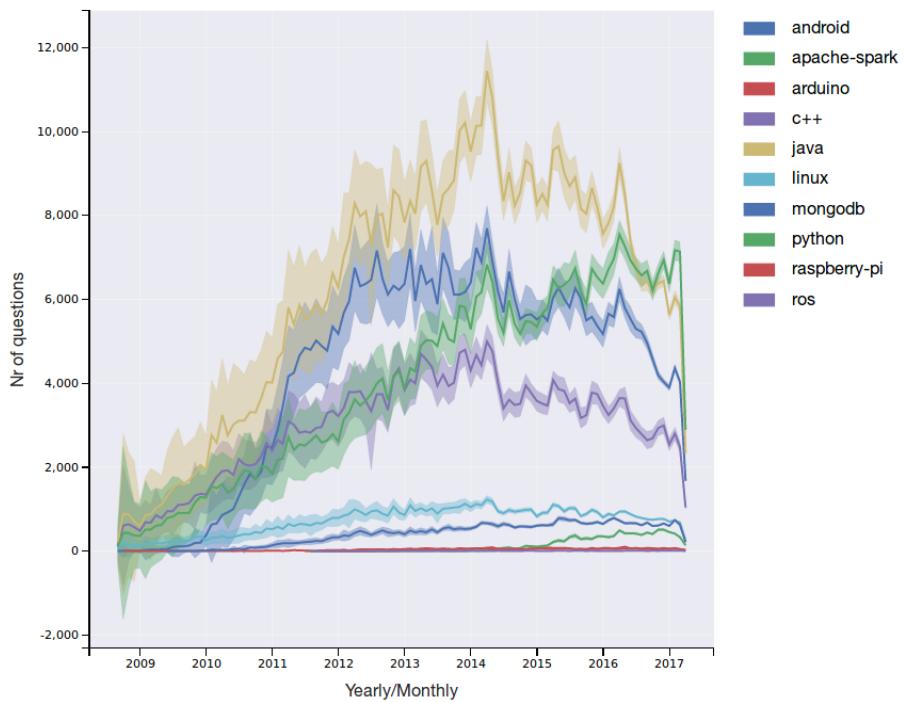


Figure 1: Degree distribution compilation

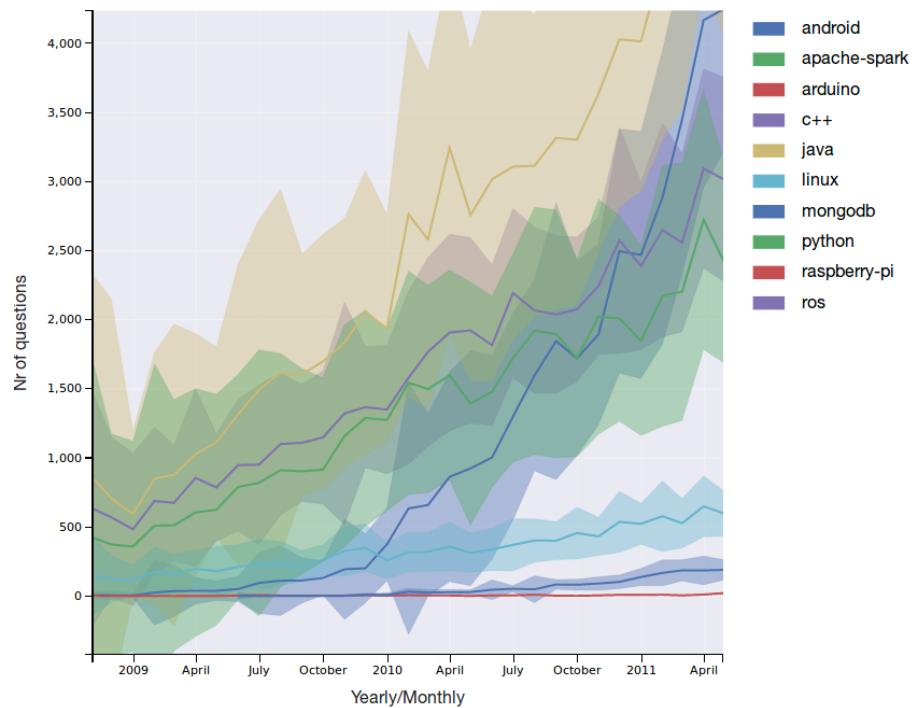
We can confirm that for real networks a power-law distribution is a very good approximation, this means that usually for real networks a lot of nodes only have a few links while only a few nodes have a big number of links.

6.3 Comparing the evolution of topics over time

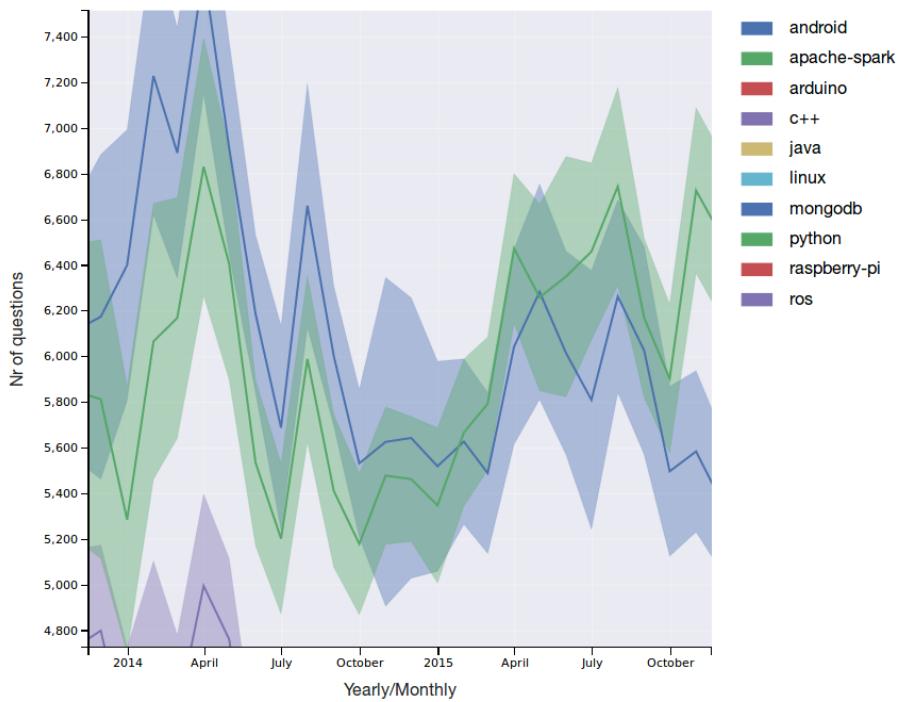
By watching the time evolution of the topics:



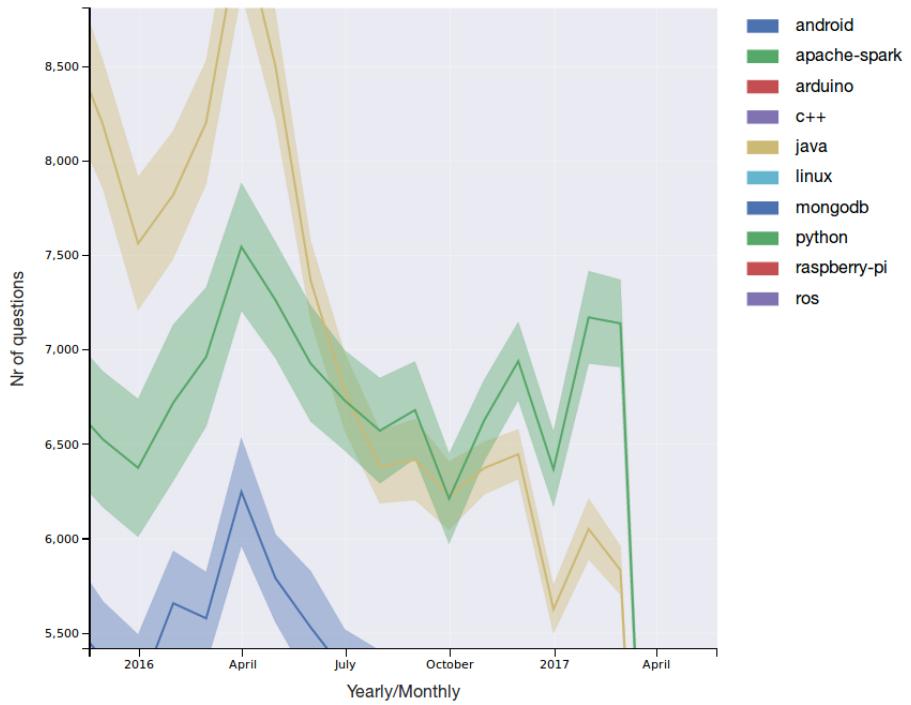
It is interesting to see how important the Java language has been from the beginning of stackoverflow, and with the appearance of Android, it became even stronger:



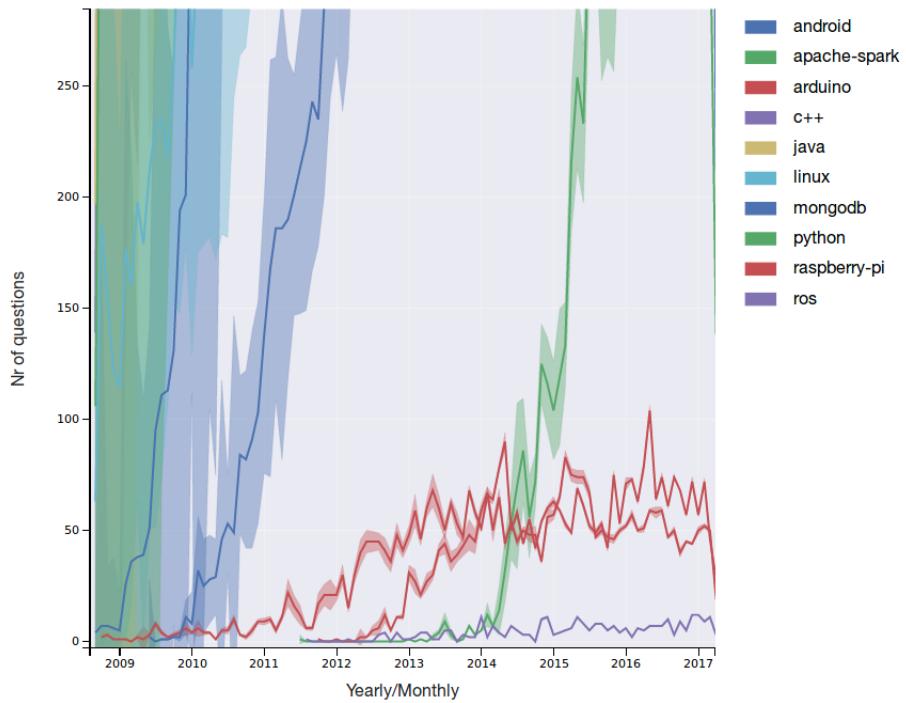
Besides, let us check how Python has been struggling for the last couple of years and it only overtakes Android by February-March 2015,



and overtakes Java by July-2016:

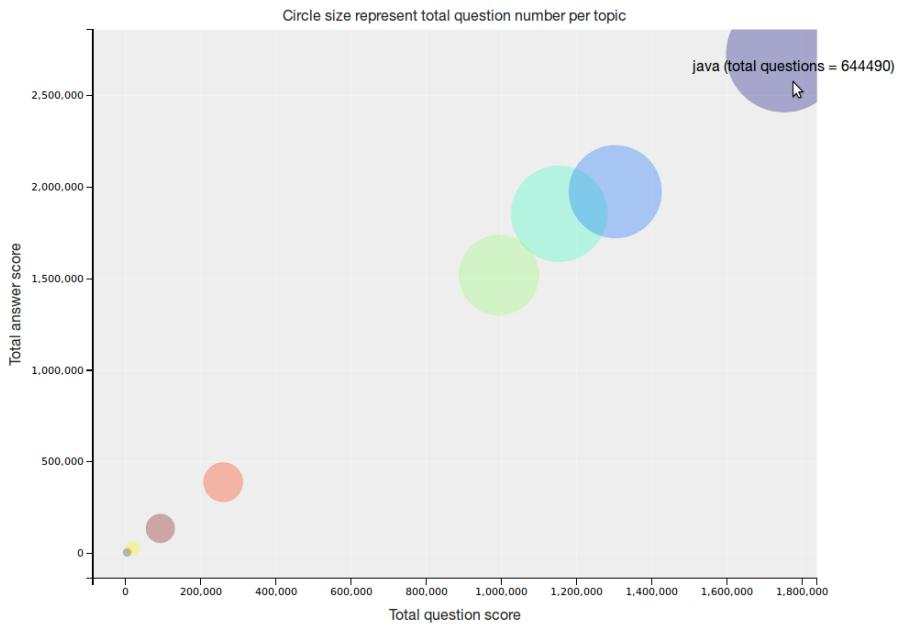


Finally, if we take a look to the lowest part of the plot, we can discover an approximate time when each technology has appeared (at least for those who appeared after the beginning of stackoverflow):



6.4 Comparing the global relevance of topics in scatter plots

This plot is quite self-explanatory, here we can check the big difference in the relevance and influence from the old well-known languages (like Java and Python) and the new experimental frameworks or technologies (like ROS, Raspberry-pi and Apache-Spark):



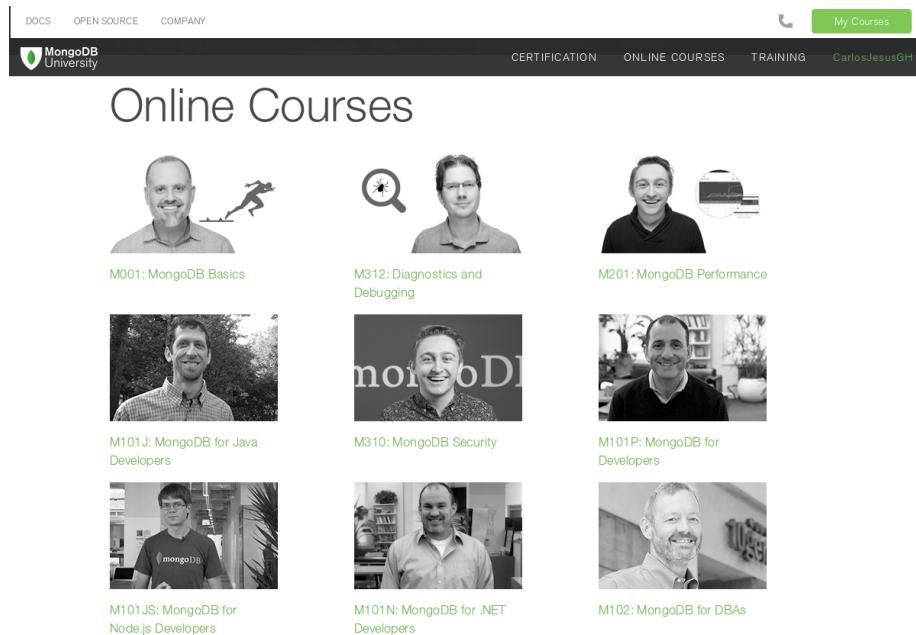
7 Possible improvements

There are several possible improvements for this data analysis, like including some other topics like C, PHP, Scala, Groovy, IOS, etc. Another easy improvement is selecting some extra information from the posts and use it in the plots, for instance using the number of comments in the questions or the number of comments in the accepted answer, or the number of answer apart from the accepted one, etc.

Besides, there are some other improvements, not so easy, but with an important impact in the data analysis, this improvements are:

Using a database: databases are good to organize, index and search into big amounts of data, much of the time the performance is much better querying into databases compared to simple text files. A great option for this purpose is the NoSql Json-like database MongoDB [10].

MongoDB is free, open source and also include some free courses in the official website:



The screenshot shows the MongoDB University website's 'Online Courses' section. At the top, there's a navigation bar with links for 'DOCS', 'OPEN SOURCE', 'COMPANY', 'CERTIFICATION', 'ONLINE COURSES', 'TRAINING', and a user account 'CarlosJesusGH'. Below the navigation, the page title 'Online Courses' is centered. The main content area displays nine course cards arranged in a 3x3 grid. Each card includes a thumbnail image, the course name, and a brief description. The courses are:

- M001: MongoDB Basics
- M312: Diagnostics and Debugging
- M201: MongoDB Performance
- M101J: MongoDB for Java Developers
- M310: MongoDB Security
- M101P: MongoDB for Developers
- M101JS: MongoDB for Node.js Developers
- M101N: MongoDB for .NET Developers
- M102: MongoDB for DBAs

Parallel computing: this is probably the best improvement and also the most difficult to carry out, there are some free tools out there to accomplish this task, like Hadoop [11], but mastering this tools require significant time, effort and hardware to make tests. A really good option for this task, considering that we are using Jupyter Notebooks (and even more if we decide to use MongoDB), is to give a try on Apache-Spark. “Apache Spark™ is a fast and general engine for large-scale data processing” [12].

We think it's a better option because it can be combined with different languages and tools, for instance there are free courses and references to combine Apache-Spark with MongoDB and Jupyter Notebooks in their own official websites.



A screenshot of the MongoDB University course page for M233. The page has a header with "Courses > M233". Below the header, there is a section titled "Next Session:" with the start date "16 May 2017 at 17:00 UTC" and end date "15 May 2018 at 17:00 UTC". A green "Register" button is present. To the right, there is a "What You'll Learn" section containing text about the course's purpose and prerequisites. The text states: "This course provides an introduction to Spark and helps students get started using the MongoDB Spark connector to build data analytics applications. We provide an overview of the Spark Scala and Java APIs with plenty of sample code and demonstrations." Prerequisites are described as requiring an intermediate level of expertise with MongoDB, with a passing grade in M101J or 3-6 months experience developing MongoDB applications in Java or Scala.

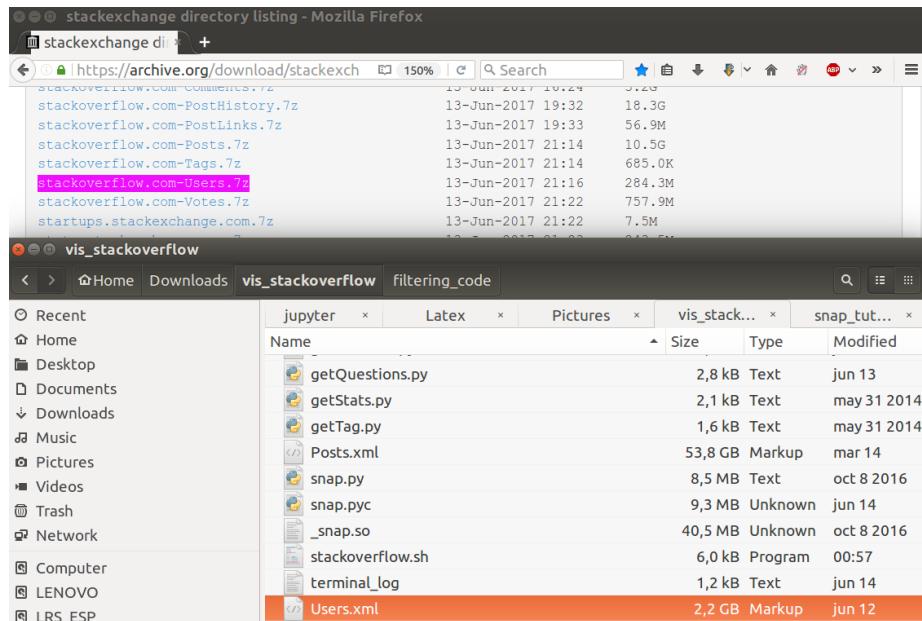
A screenshot of the Jupyter website. The URL in the address bar is https://jupyter.org/index.html. The page features the Jupyter logo (an orange circle with a white dot) and the word "jupyter". Below the logo is the Apache Spark logo, which consists of the word "Spark" in a stylized font with a star. A section titled "Big data integration" is shown, with the text: "Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.".

8 Conclusions

The idea with this project was essentially create some visualizations by using the selected data source from the Stanford Large Network Dataset Collection. As I have been working with Python in the last months, I decided to make use of this project to learn some of the most useful libraries and tools for data analysis in this language, this tools are Pandas, Seaborn, Mpld3 and of course the Jupyter Notebook.

Even when learning new tools can take you a lot of time, I am happy that I did it this way, now I have a broader idea about how useful the data analysis can be, and specially how valuable an interactive data visualization can be when analyzing a dataset.

I must say I wish I had a few more time to include also in this project the analysis of the users dataset from the same stackoverflow collaboration network, I even downloaded and extracted the dataset but filtering and analyzing the data takes a considerable amount of time.



The screenshot shows a dual-monitor setup. The left monitor displays a Mozilla Firefox browser window titled "stackexchange directory listing - Mozilla Firefox". The URL bar shows "https://archive.org/download/stackexchange/di...". The main content area lists several files from the Stack Exchange archive, including "stackoverflow.com-Comments.7z", "stackoverflow.com-PostHistory.7z", "stackoverflow.com-PostLinks.7z", "stackoverflow.com-Posts.7z", "stackoverflow.com-Tags.7z", "stackoverflow.com-Users.7z", "stackoverflow.com-Votes.7z", and "startups.stackexchange.com.7z". The right monitor displays a terminal window titled "vis_stackoverflow" with a file browser interface. The sidebar on the left shows a file tree with categories like Recent, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Trash, Network, Computer, LENOVO, and LRS ESP. The main pane shows a list of files in the current directory, including "getQuestions.py", "getStats.py", "getTag.py", "Posts.xml", "snap.py", "snap.pyc", "_snap.so", "stackoverflow.sh", "terminal_log", and "Users.xml". The "Users.xml" file is highlighted with a red border.

It would be interesting mixing the data from both datasets and get some noteworthy extra information.

Finally I would like to mention another interesting feature about Jupyter Notebooks and it is the capacity of sharing the notebooks online by using the Jupyter Notebook Viewer, “the nbviewer is a simple way to share Jupyter Notebooks” [13].

References

- [1] jupyter.org. <http://jupyter.org/>. [Online; accessed 19-Juny-2017].
- [2] mpld3. <http://mpld3.github.io/>. [Online; accessed 19-Juny-2017].
- [3] Cyrille Rossant. *Learning IPython for Interactive Computing and Data Visualization*. Second edition, 2013.
- [4] A gallery of interesting Jupyter Notebooks. <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks>. [Online; accessed 19-Juny-2017].
- [5] snap.stanford.edudata. <https://snap.stanford.edu/data/>. [Online; accessed 19-Juny-2017].
- [6] stackexchange. <https://archive.org/download/stackexchange/>. [Online; accessed 19-Juny-2017].
- [7] snap.stanford.eduprojnsnap-icwsmicwsm14-T4-code.zip. <http://snap.stanford.edu/proj/snap-icwsm/icwsm14-T4-code.zip>. [Online; accessed 19-Juny-2017].
- [8] Wikipedia. Degree distribution — wikipedia, the free encyclopedia, 2017. [Online; accessed 20-June-2017].
- [9] Degree centrality. https://en.wikipedia.org/wiki/Centrality#Degree_centrality. [Online; accessed 19-Juny-2017].
- [10] mongodb.com. <https://www.mongodb.com/>. [Online; accessed 19-Juny-2017].
- [11] hadoop.apache.org. <http://hadoop.apache.org/>. [Online; accessed 19-Juny-2017].
- [12] spark.apache.org. <https://spark.apache.org/>. [Online; accessed 19-Juny-2017].
- [13] nbviewer.jupyter.org. <http://nbviewer.jupyter.org/>. [Online; accessed 19-Juny-2017].