VIS - Global Practice (InfoVis)

Visualization and Interaction Systems

Carlos García

June 2017

Contents

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/distrust)

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

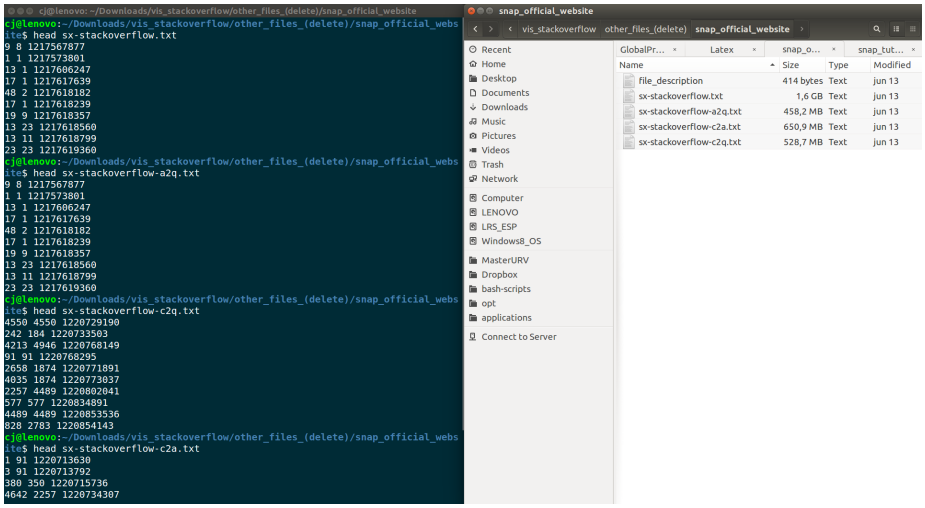
Online reviews: Data from online review systems such as BeerAdvocate and Amazon

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:

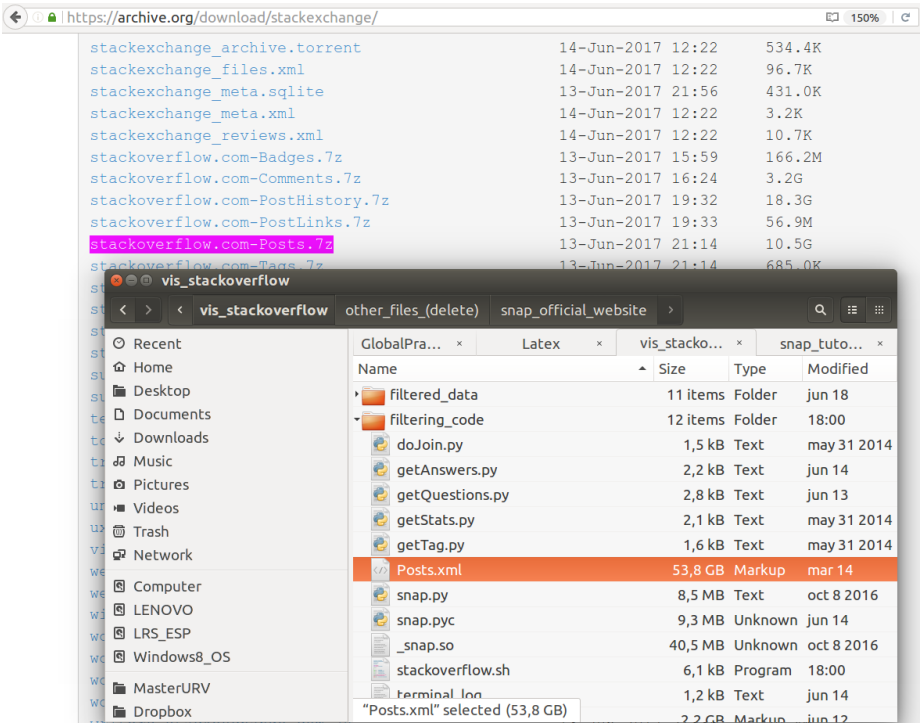
Temporal networks

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:



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].



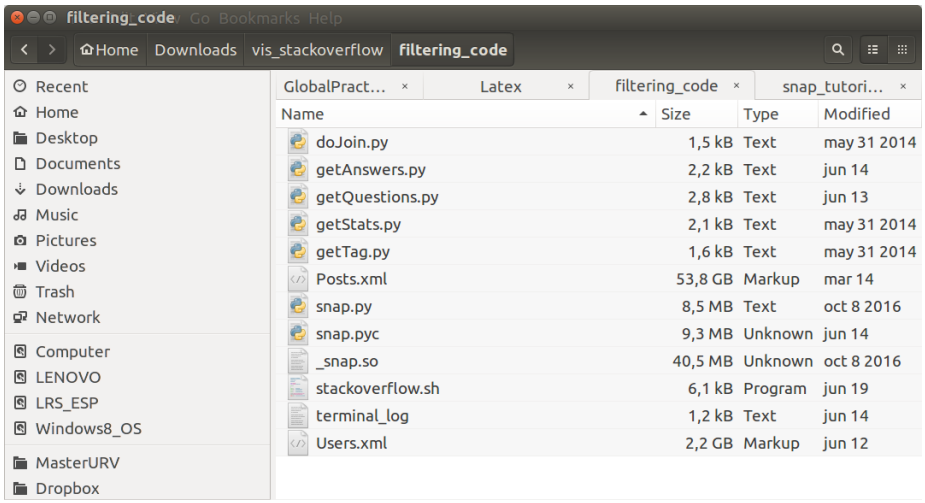
The compressed file is about 10.5 GB, and once we uncompress it increases to about 53.8 GB, this is obviously a huge amount of data to be analyzed directly from the main dataset, that's why we decided performing an initial filtering process where we locate and separate the posts according to the topics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



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:

```
#   requirements:
#     python installed
#     Snap.py installed, http://snap.stanford.edu
#     Posts.xml, https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z, uncompress

echo `date` ... "START analysis of StackOverflow"

# get all the question posts and accepted answers (15min)
echo `date` ... "extracting questions ..."
python getQuestions.py Posts.xml > questions.txt

# get all the answer posts (15min)
echo `date` ... "extracting answers ..."
python getAnswers.py Posts.xml > answers.txt

# get all the Java question posts, id only (15min)
echo `date` ... "identifying Java questions ..."
python getTag.py Posts.xml java > java.txt

# select questions with a Java tag (20s)
echo `date` ... "selecting Java questions ..."
python doJoin.py ../www15-data/java.txt ../www15-data/questions.txt 1 1 > java-posts.txt

# identify users of accepted answers (40s)
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py ../www15-data/answers.txt java-posts.txt 1 3 > qa.txt

# create a graph and find top users (5s)
echo `date` ... "building a QA graph and calculating statistics ..."
python getStats.py qa.txt 2 6 > stats.txt

echo `date` ... "results are stored in stats.txt"
echo `date` ... "END   analysis of StackOverflow"
```

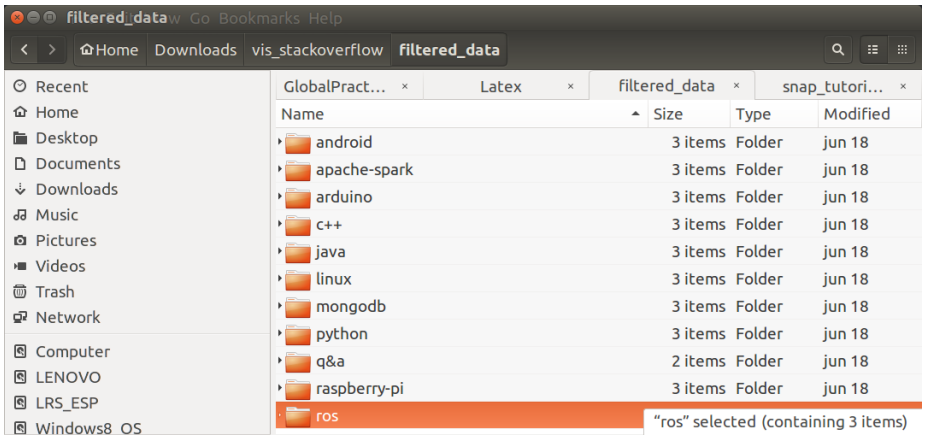
In the above image, it is being executed for Java posts, but we can change it depending on our choice, in our case we used this bash script:

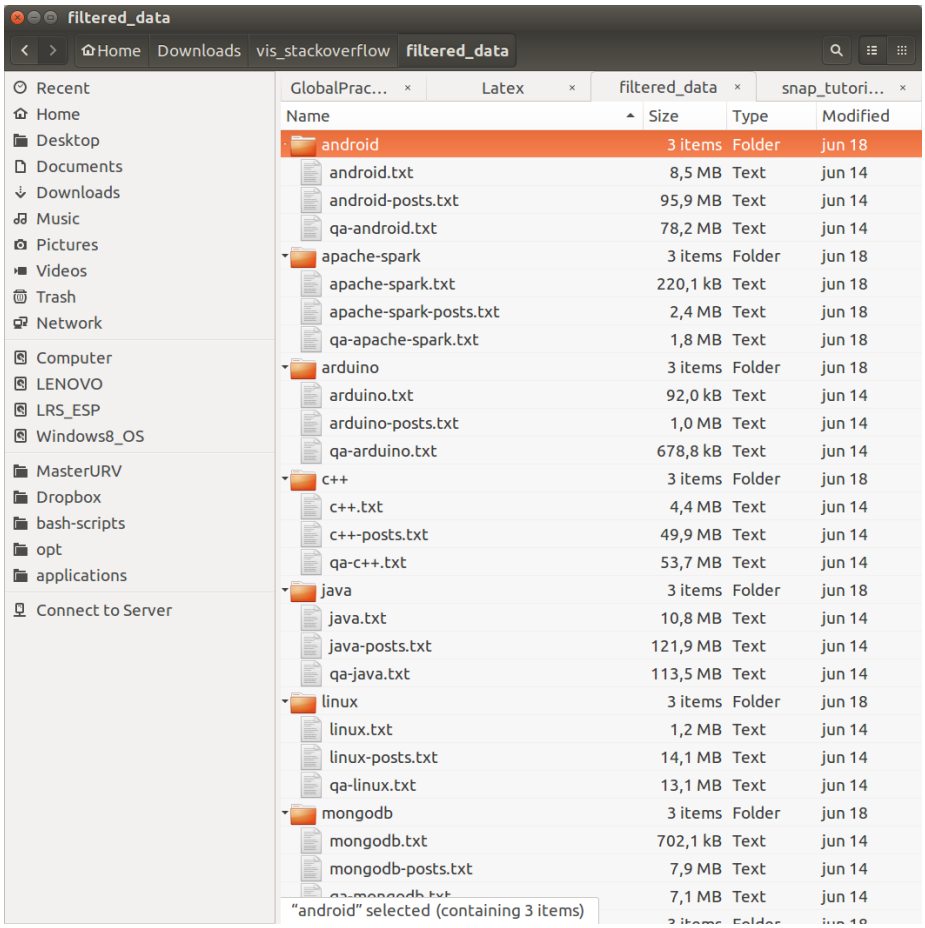
```
echo `date` ... "START analysis of StackOverflow"
# get all the question posts and accepted answers (15min)
echo `date` ... "extracting questions ..."
python getQuestions.py Posts.xml > questions.txt

# get all the answer posts (15min)
echo `date` ... "extracting answers ..."
python getAnswers.py Posts.xml > answers.txt

# get all the Java question posts, id only (15min)
 echo `date` ... "identifying Java questions ..."
 python getTag.py Posts.xml java > java.txt
 echo `date` ... "identifying Python questions ..."
 python getTag.py Posts.xml python > python.txt
 echo `date` ... "identifying C++ questions ..."
 python getTag.py Posts.xml c++ > c++.txt
 echo `date` ... "identifying Raspberry questions ..."
 python getTag.py Posts.xml raspberry-pi > raspberry-pi.txt
 echo `date` ... "identifying Arduino questions ..."
 python getTag.py Posts.xml arduino > arduino.txt
 echo `date` ... "identifying Ros questions ..."
 python getTag.py Posts.xml ros > ros.txt
 echo `date` ... "identifying MongoDB questions ..."
 python getTag.py Posts.xml mongodb > mongodb.txt
 echo `date` ... "identifying Android questions ..."
 python getTag.py Posts.xml android > android.txt
 echo `date` ... "identifying Linux questions ..."
 python getTag.py Posts.xml linux > linux.txt
 echo `date` ... "identifying Spark questions ..."
 python getTag.py Posts.xml apache-spark > spark.txt

# select questions with a Java tag (20s)
 echo `date` ... "selecting Java questions ..."
 python doJoin.py java.txt questions.txt 1 1 > java-posts.txt
 echo `date` ... "selecting Python questions ..."
 python doJoin.py python.txt questions.txt 1 1 > python-posts.txt
 echo `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:

```
# identify users of accepted answers (40s)
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt java-posts.txt 1 3 > qa-java.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt python-posts.txt 1 3 > qa-python.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt c++-posts.txt 1 3 > qa-c++.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt raspberry-pi-posts.txt 1 3 > qa-raspberry-pi.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt arduino-posts.txt 1 3 > qa-arduino.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt ros-posts.txt 1 3 > qa-ros.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt mongodb-posts.txt 1 3 > qa-mongodb.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt android-posts.txt 1 3 > qa-android.txt
echo `date` ... "finding owners of accepted answers ..."
python doJoin.py answers.txt linux-posts.txt 1 3 > qa-linux.txt
echo `date` ... "finding owners of accepted answers ..."
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:

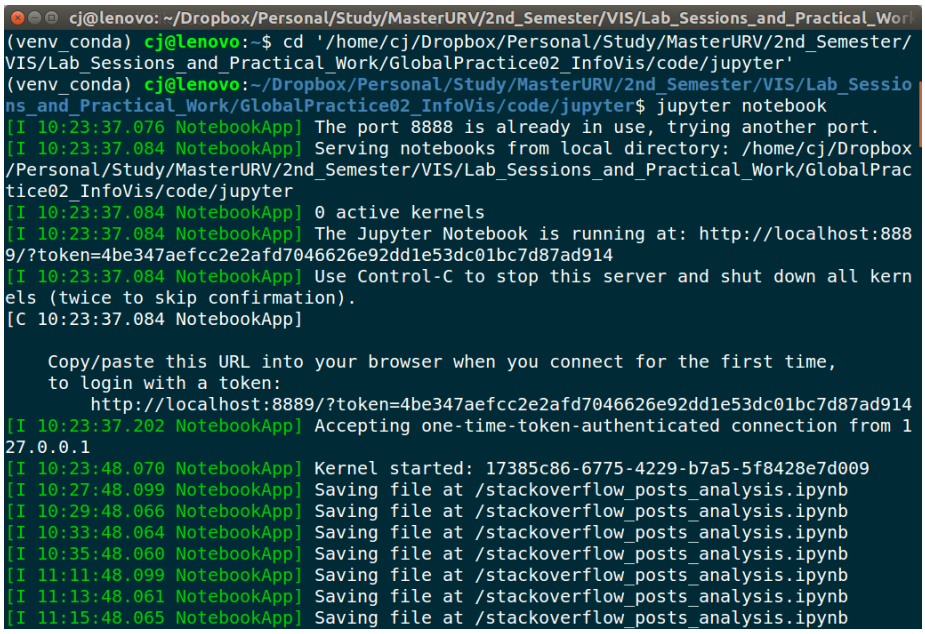




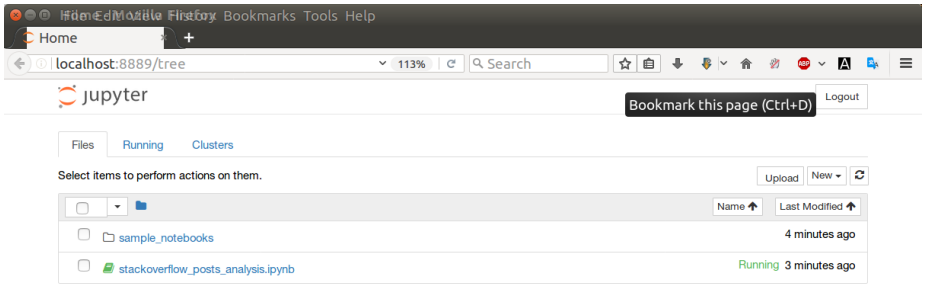
At this moment, we will focus our attention in the files "qa-topic.txt", they are placed in the folder with the same name of the topic, and this files contain all the important information related to one post, including the data about the question and the data about the accepted answer for that question.

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):

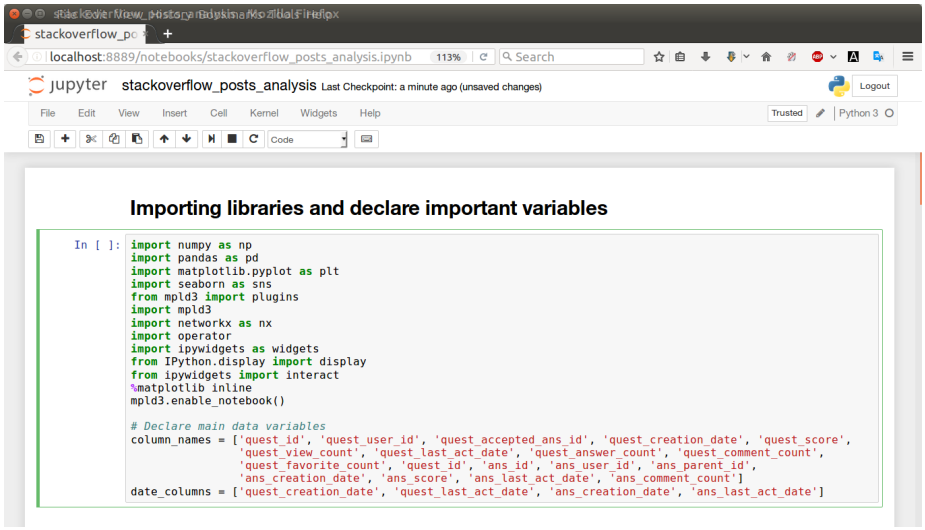


Once there, we create a new notebook and start performing the analysis in independent blocks of code:



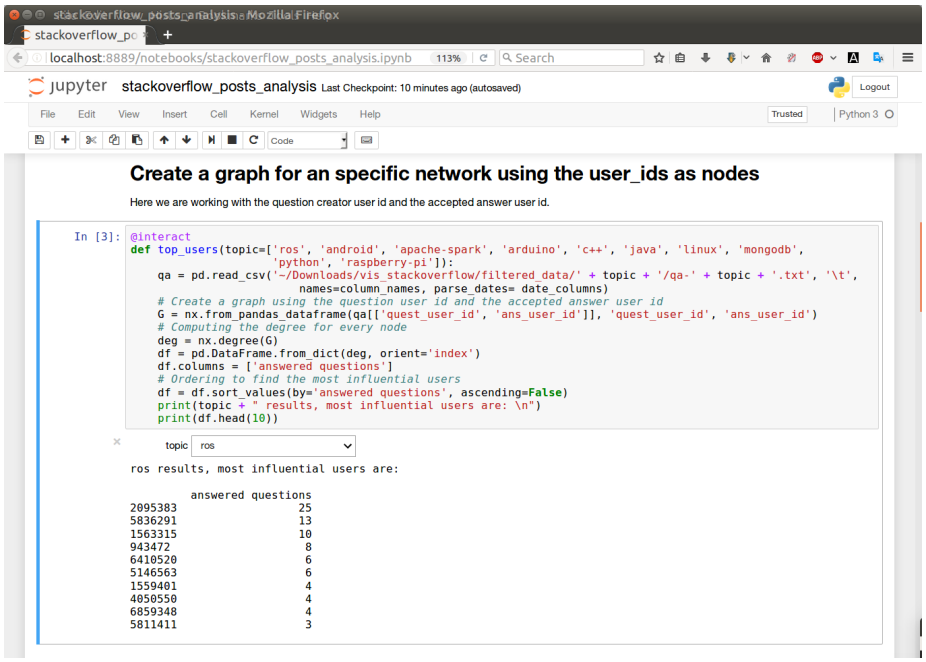
5.1 Importing all the libraries

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

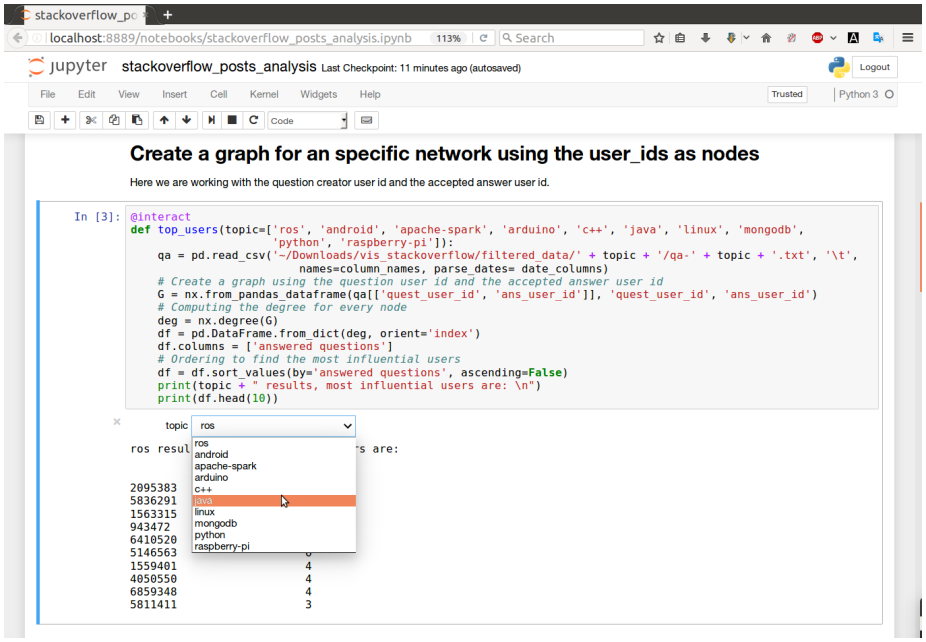


5.2 Finding the most influential users per topic

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.

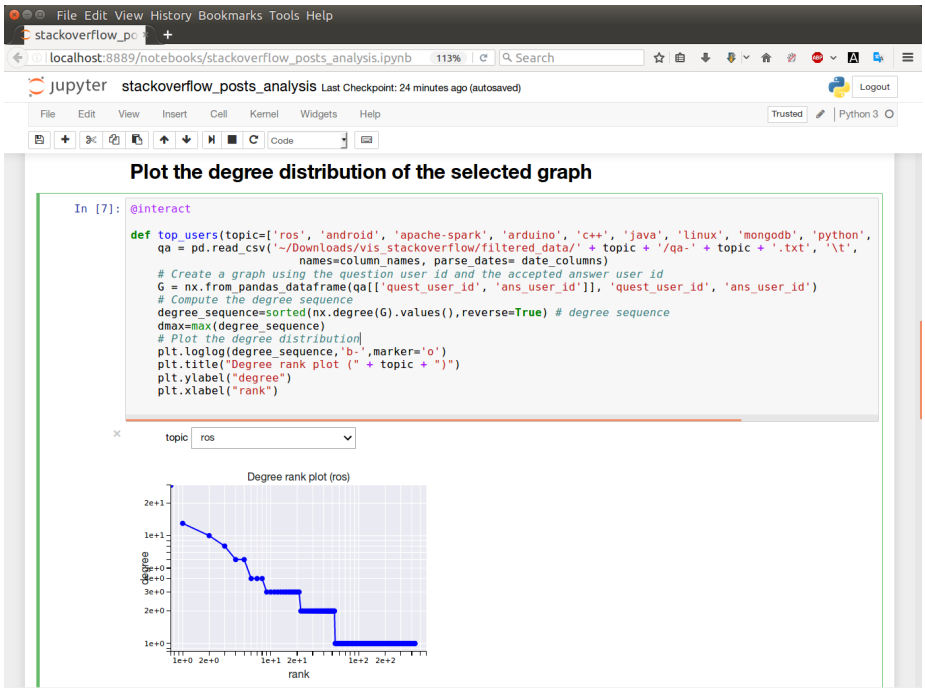


Note that this process is interactive and we can select the topic we prefer by using the dropdown list.

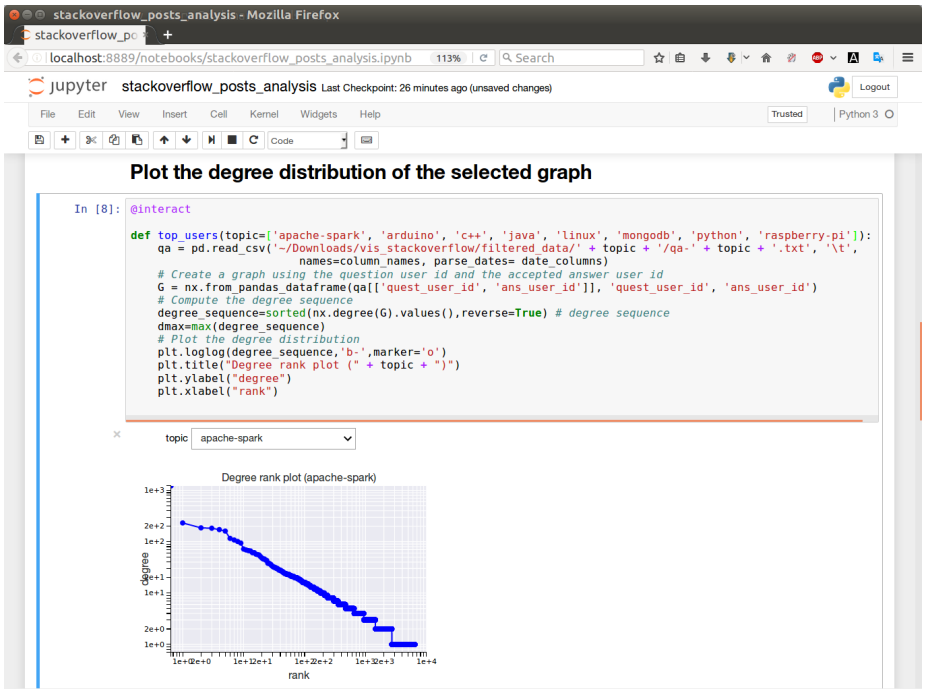


5.3 Plotting the degree distribution

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].

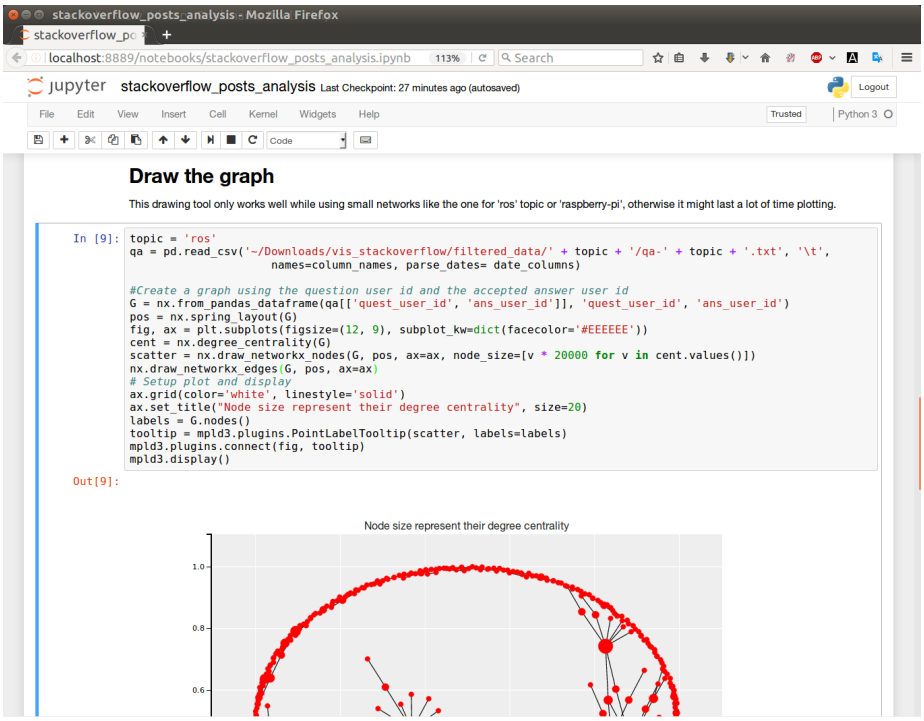


This block is also interactive and we can change the topic simply by clicking over the dropdown list or by changing the list in the code, however we prefer:

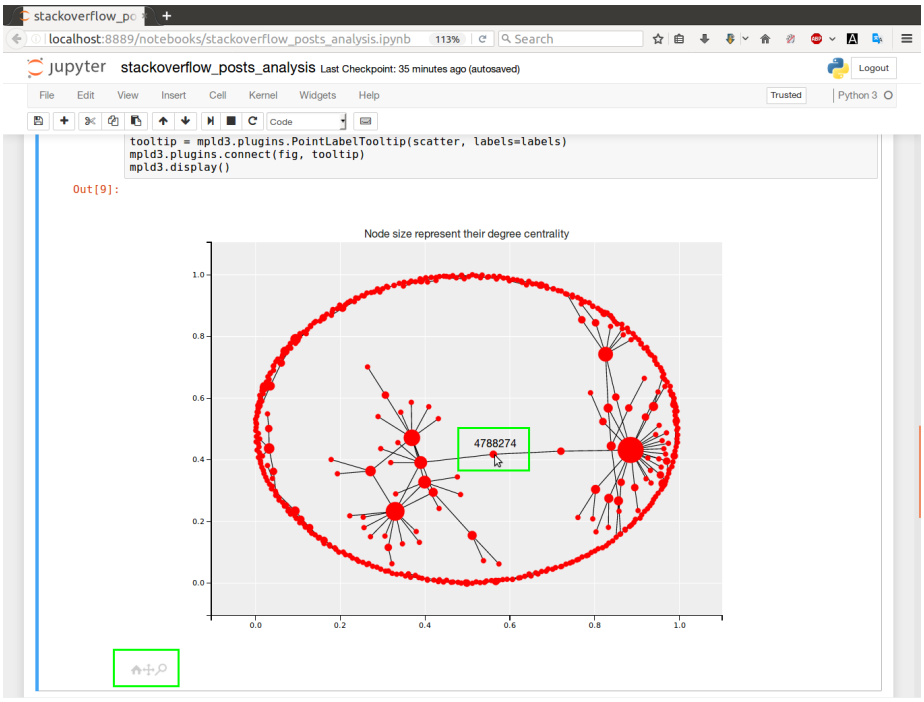


5.4 Drawing the graph

Here we make use of the networkx library and the mpld3 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.



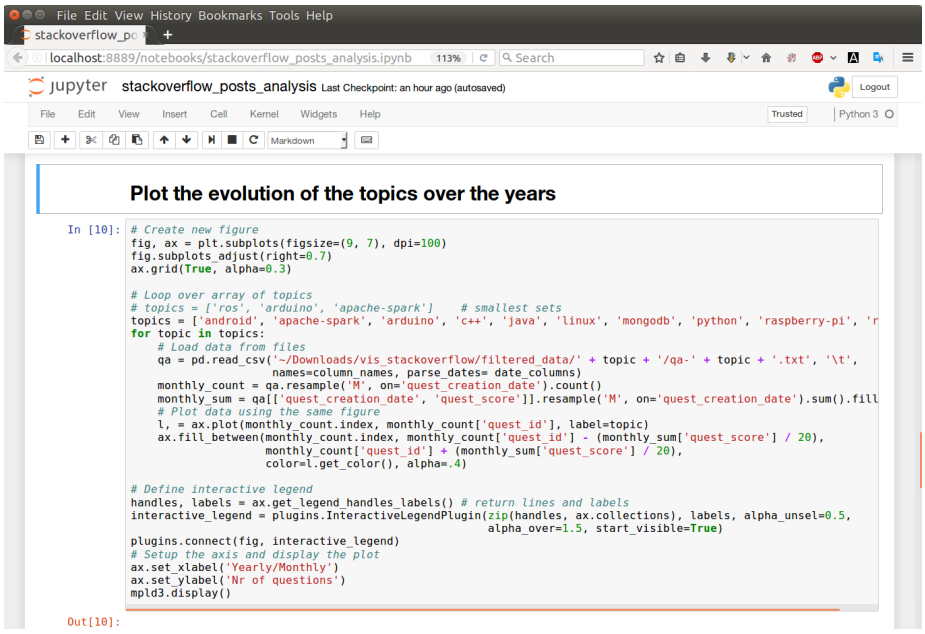
Thanks to the mpld3 library [2], we can easily interact with the plot by hovering the mouse cursor over the nodes, or by using the tool buttons (home, pan and zoom) placed in the left bottom corner of the image:

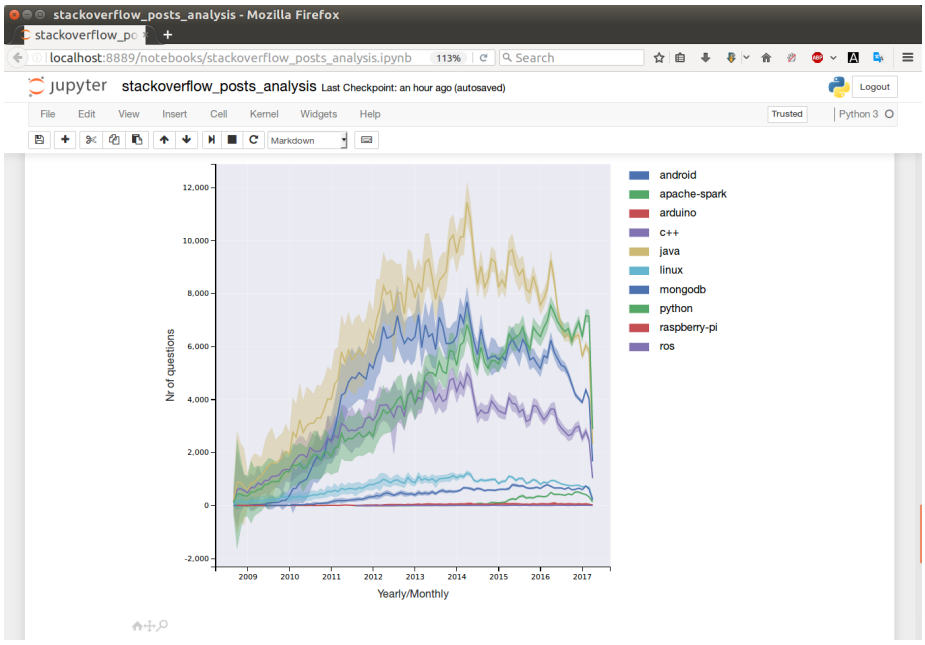


Note that this block should only by tested with small networks, otherwise it last a very long time compiling and drawing the network.

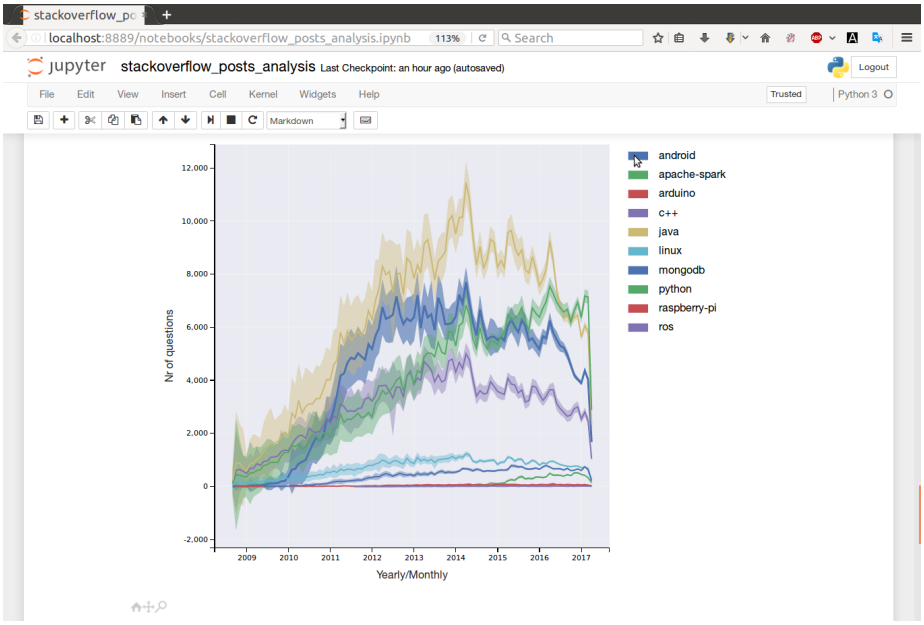
5.5 Plot the evolution of topics over the years

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.

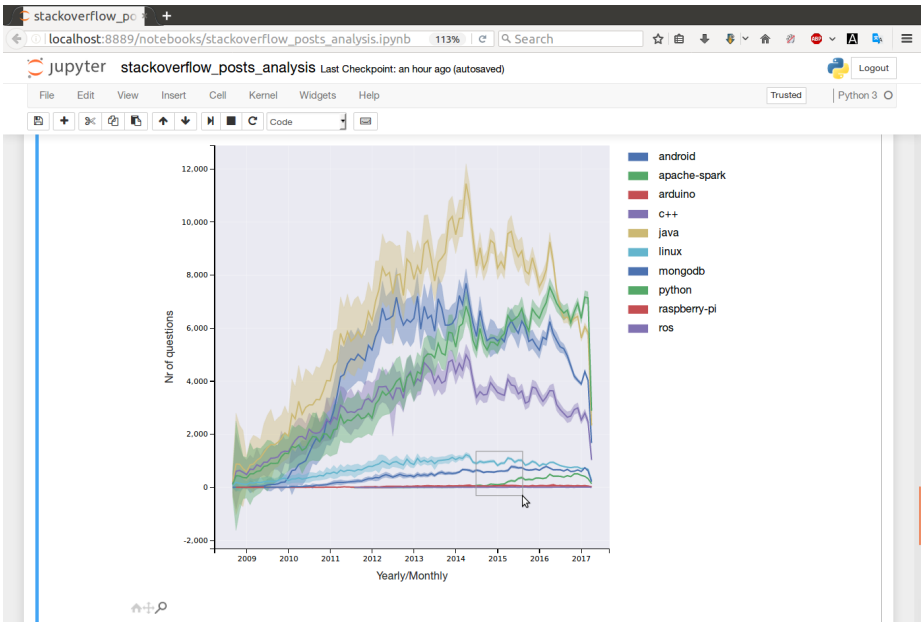




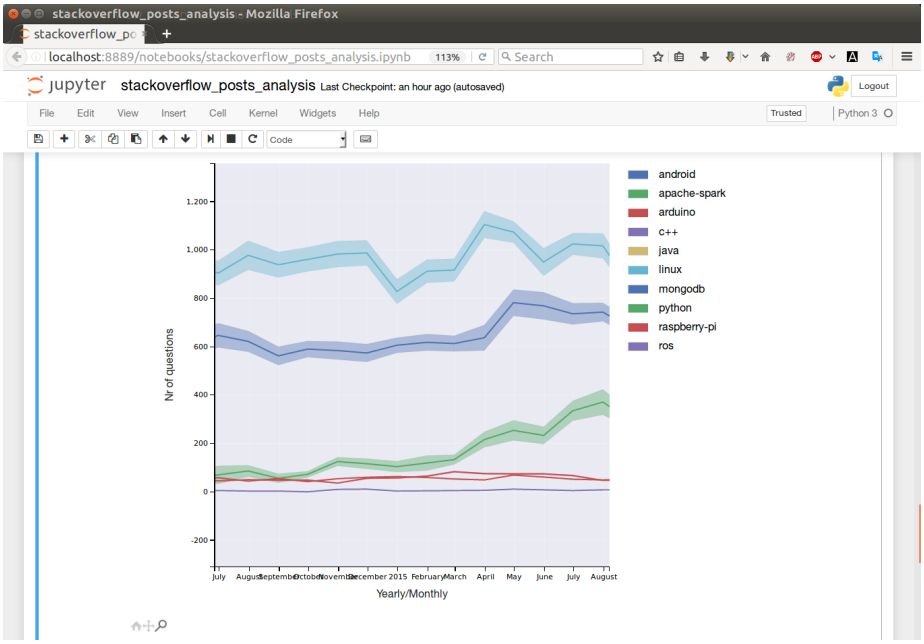
In this interactive plot we can highlight one of the topics simply by placing the mouse over its name in the legend:



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):

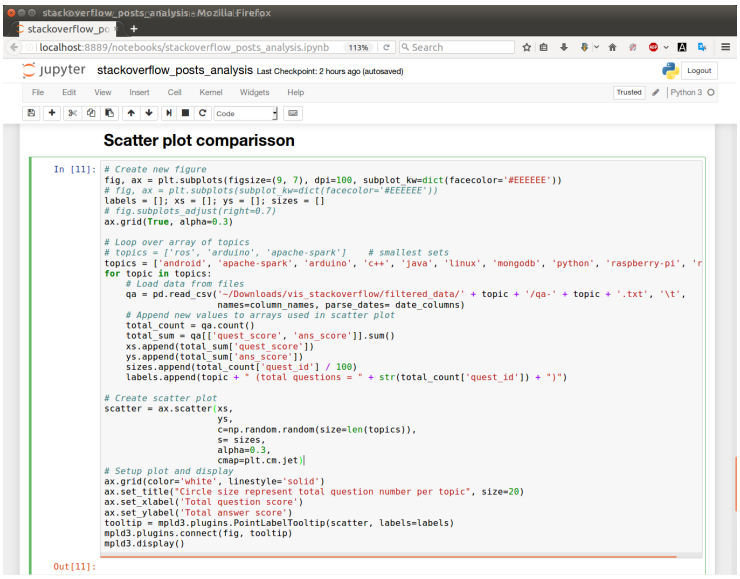


5.6 Scatter plot comparison

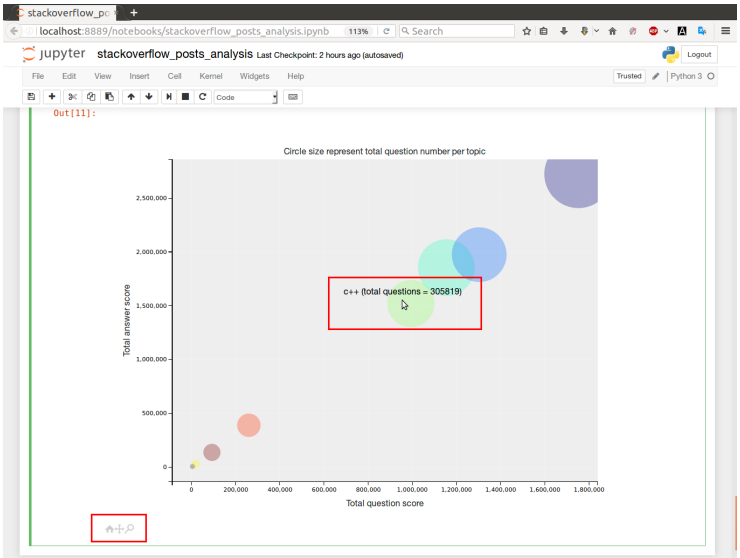
In this block we create a scatter plot trying to compare certain global properties of the topics, this properties are:

- 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)

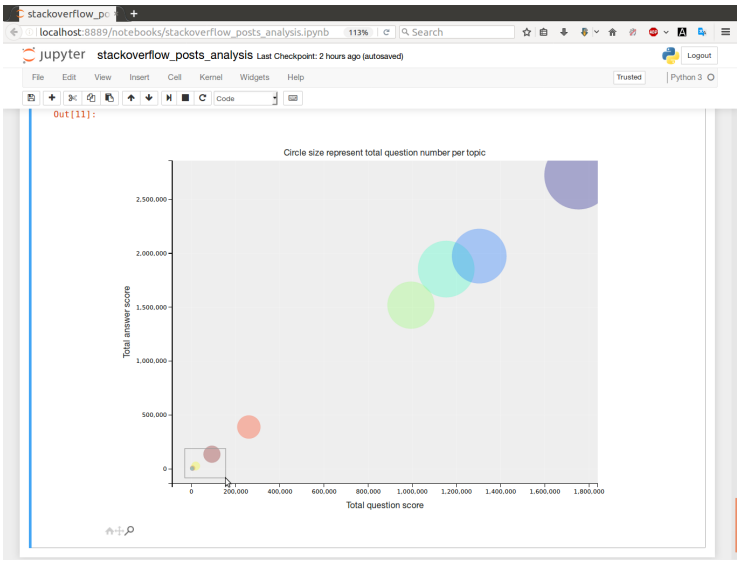
This is the code in the jupyter notebook:



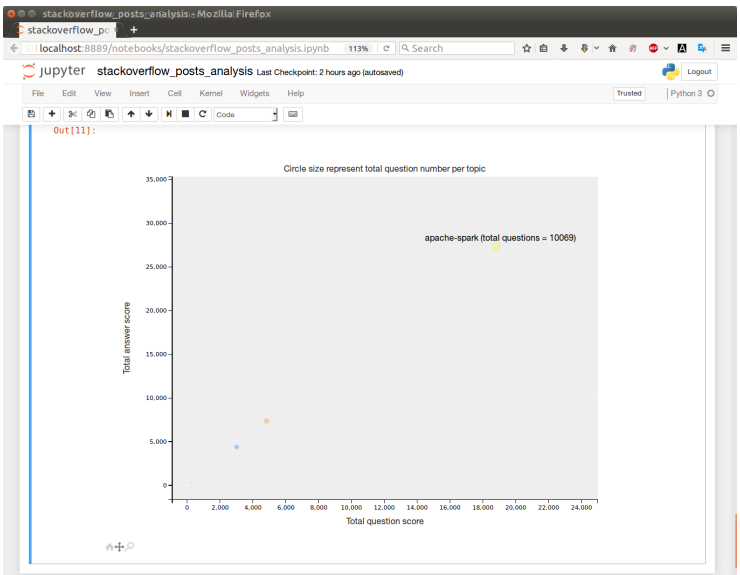
And this is the resulting output, note that this plot is also interactive, we can hover the cursor over the points to check the total number of questions related to this topic:



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

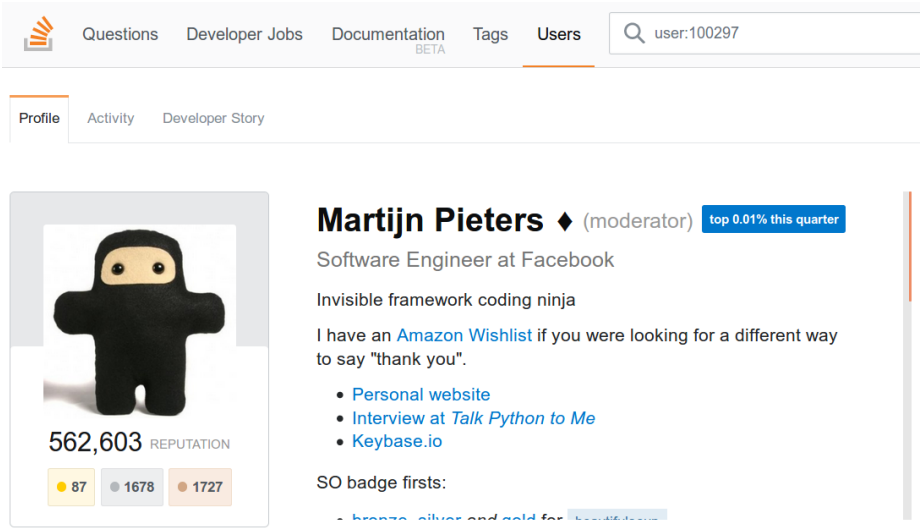
If we execute the second block from our jupyter notebook and go over each topic the results are the following:

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

topic python

```
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
```

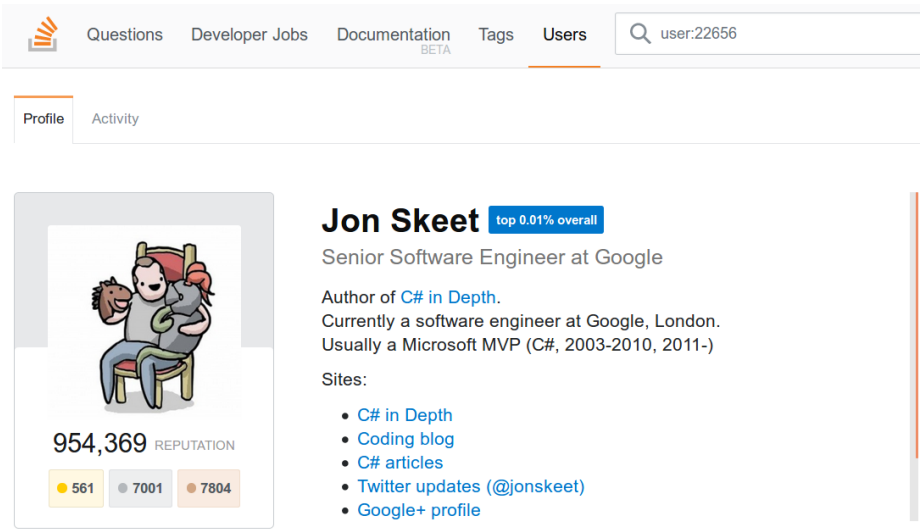
We searched about this user and this is what we found:



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

topic java

```
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
```

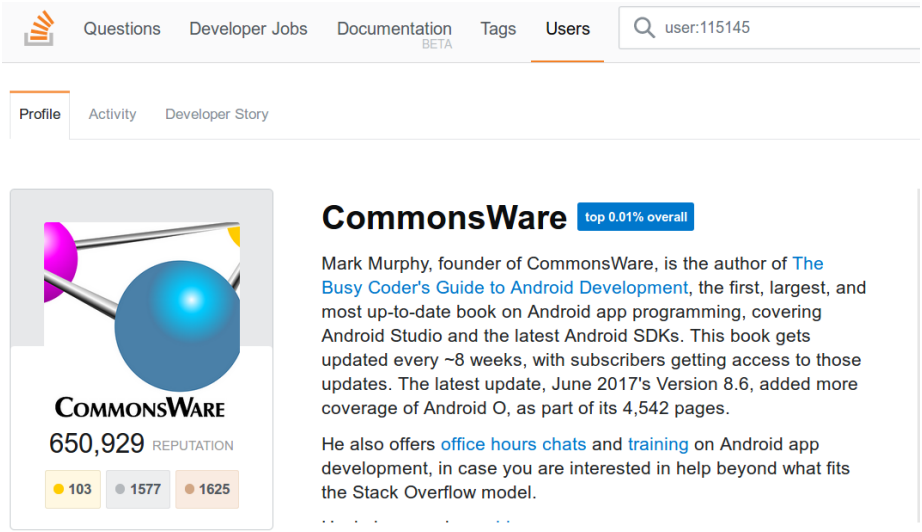


Android: Most influential user: 115145 with 8354 answered questions in this topic.

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

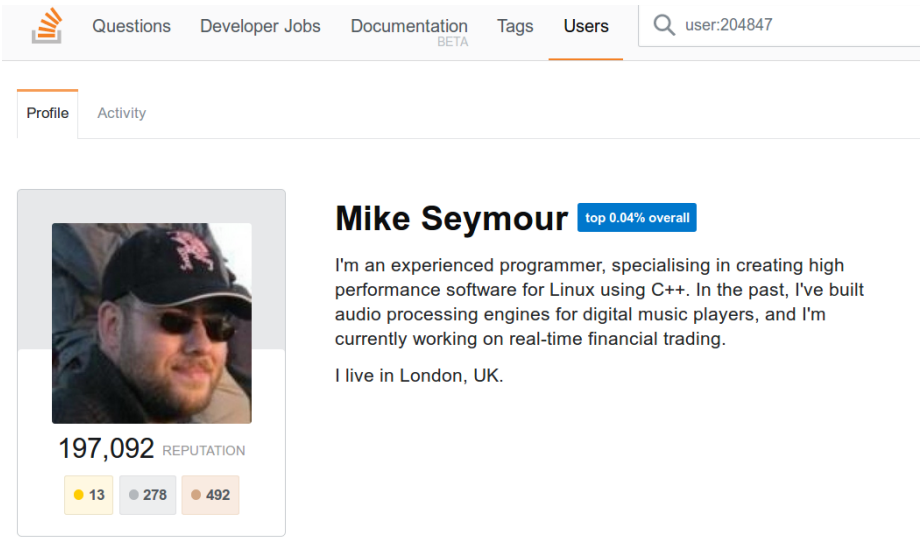


C++: Most influential user: 204847 with 1882 with 8354 answered questions in this topic.

topic c++

c++ results, most influential users are:

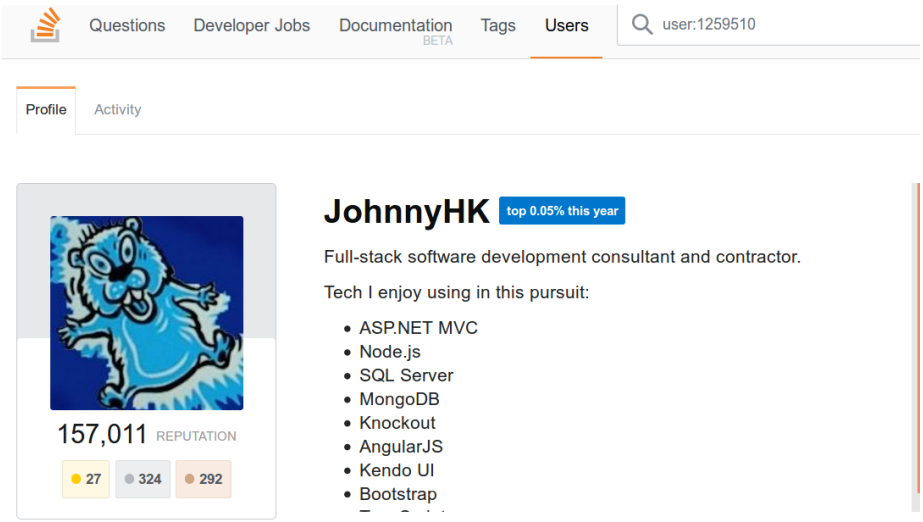
	answered questions
__none__	4281
204847	1882
596781	1746
440558	1573
179910	1558
85371	1557
661519	1489
560648	1371
673730	1283
2069064	1272



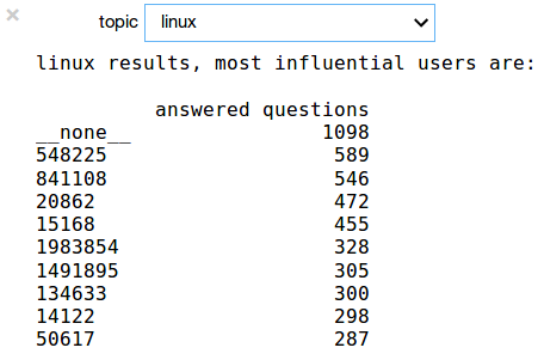
MongoDB: Most influential user: 1259510 with 1134 answered questions in this topic.

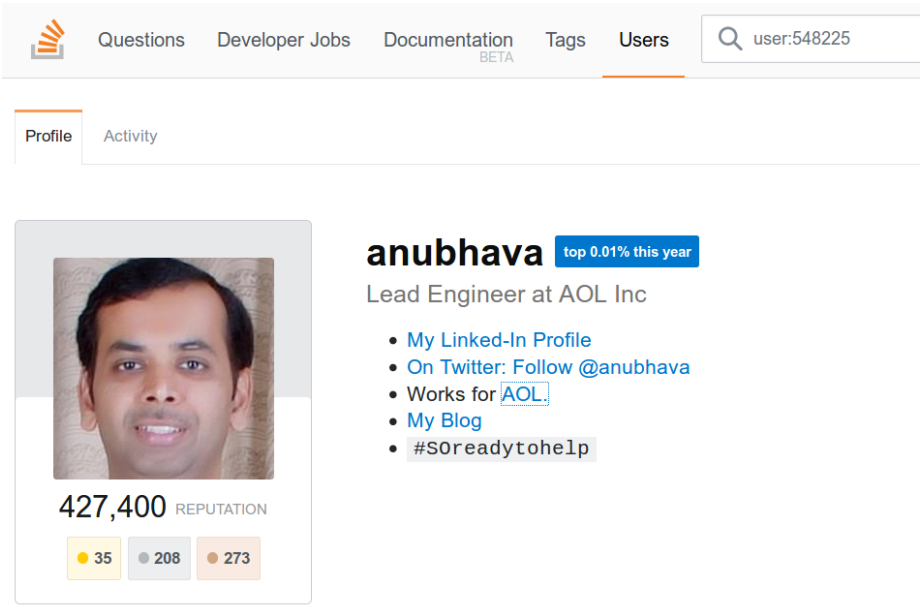
topic mongodb

```
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
```



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



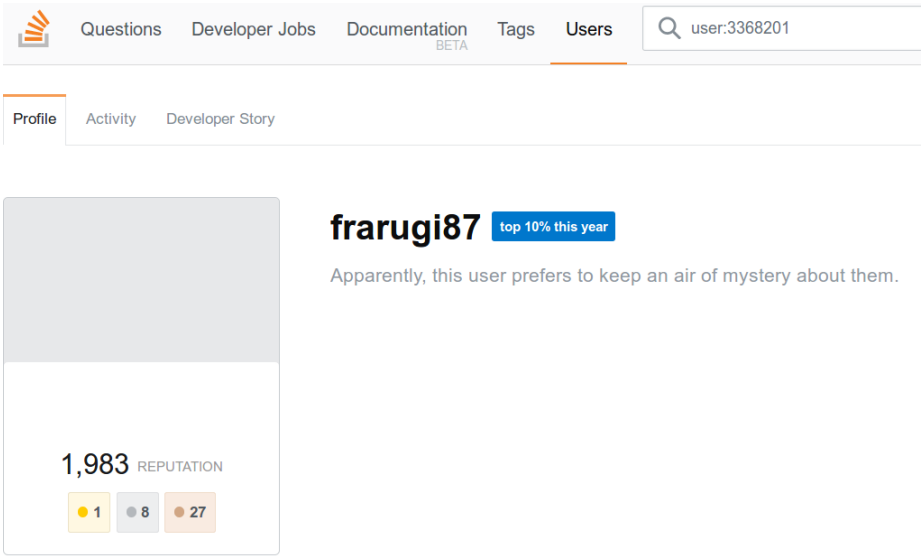


Arduino: Most influential user: 3368201 with 59 answered questions in this topic.

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

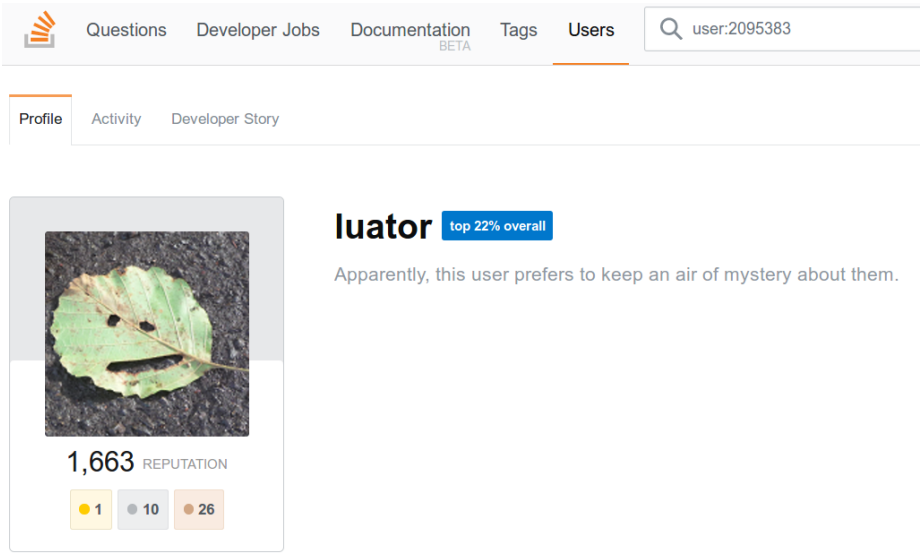


ROS (Robot Operating System): Most influential user: 2095383 with 25 answered questions in this topic.

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

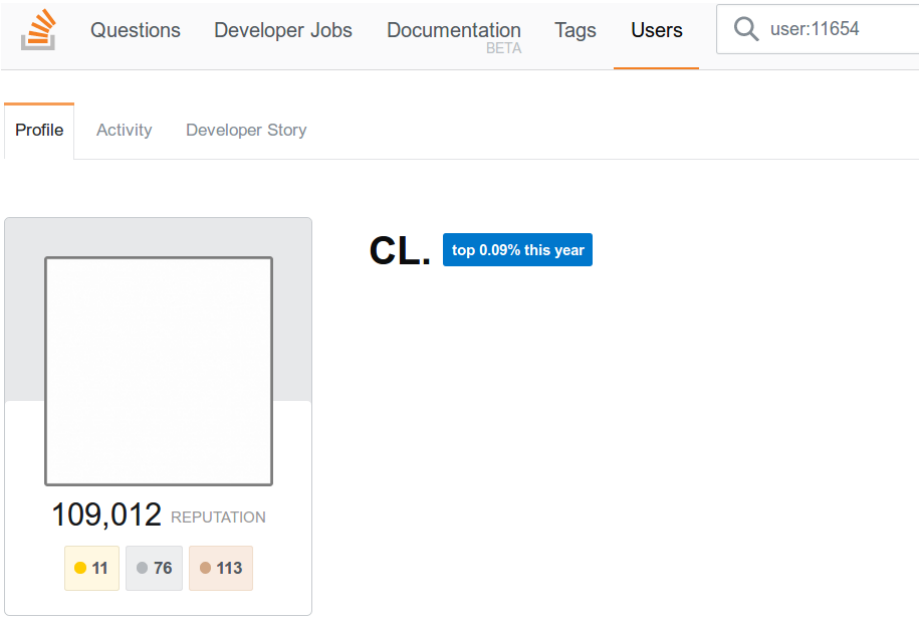


Raspberry-pi: Most influential user: 11654 with 13 answered questions in this topic.

topic raspberry-pi

raspberry-pi results, most influential users are:

	answered questions
__none__	31
11654	13
541038	13
100297	13
504554	9
2587646	9
1461050	9
3207652	8
7432	8
89766	8

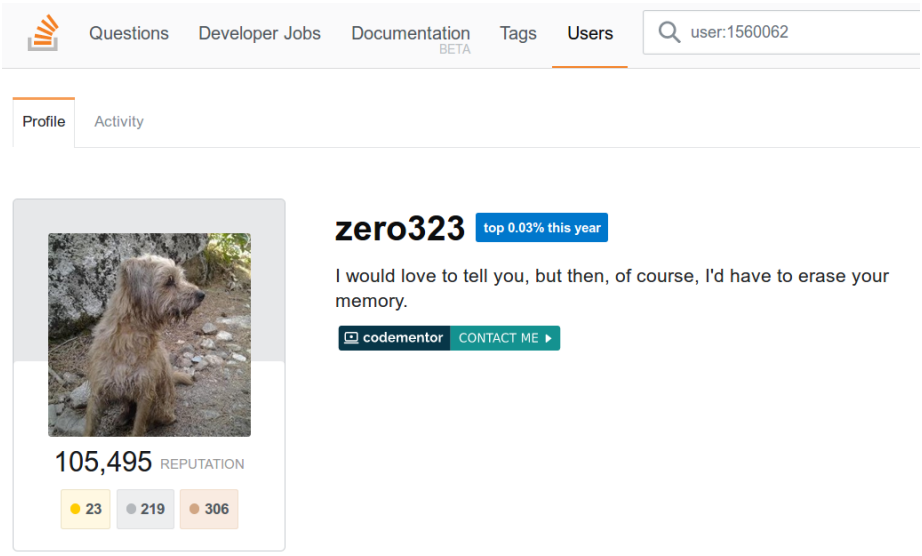


Apache-Spark: Most influential user: 1560062 with 870 answered questions in this topic.

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



6.2 Comparing the degree distribution in networks

By comparing the degree distribution in a couple of networks:

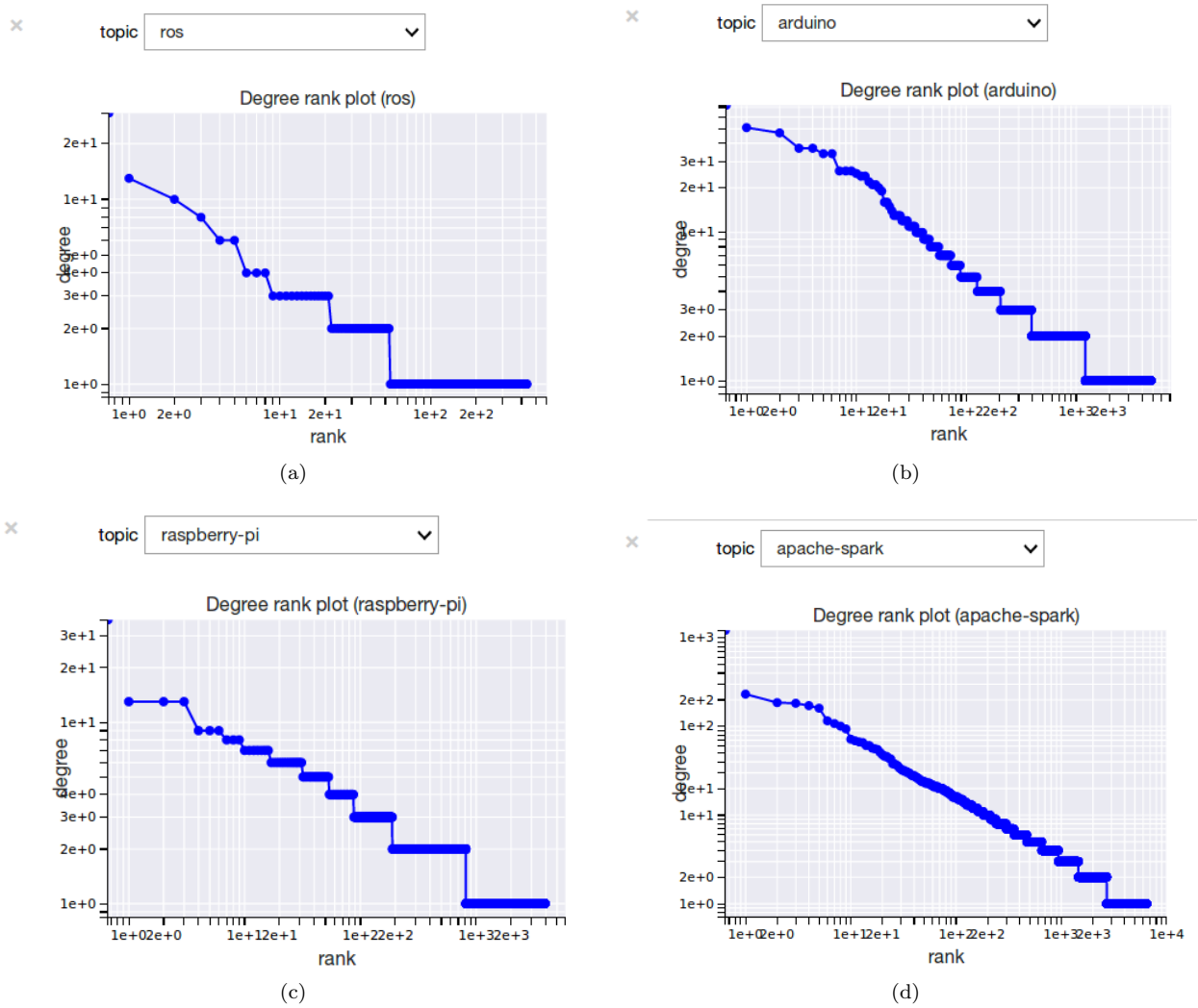
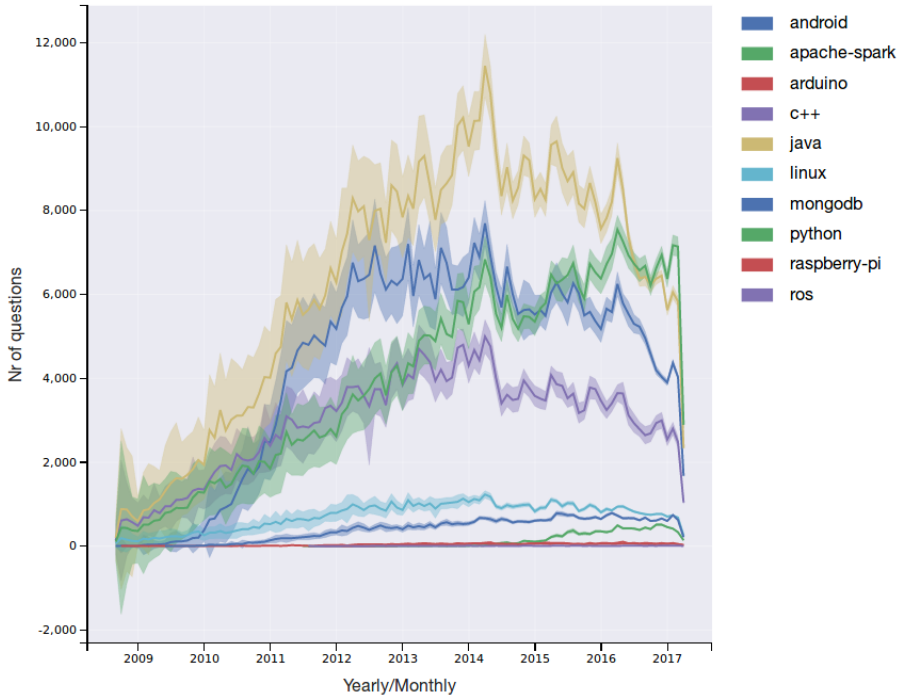


Figure 1: Degree distribution compilation

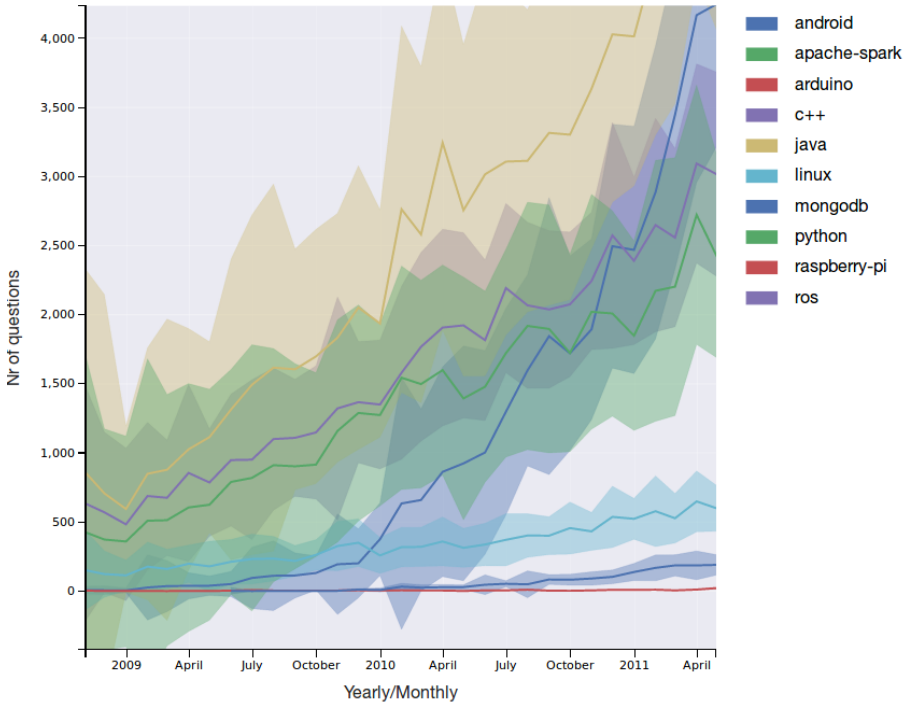
We can confirm that for real networks a power-law distribution is a very good approximation, this means that usually for real networks a lot of nodes only have a few links while only a few nodes have a big number of links.

6.3 Comparing the evolution of topics over time

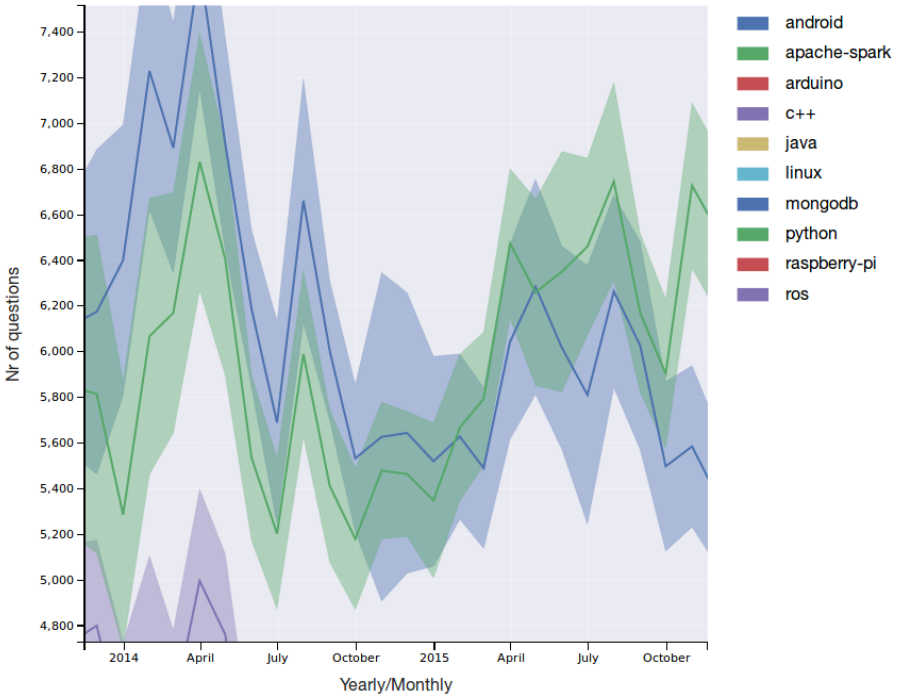
By watching the time evolution of the topics:



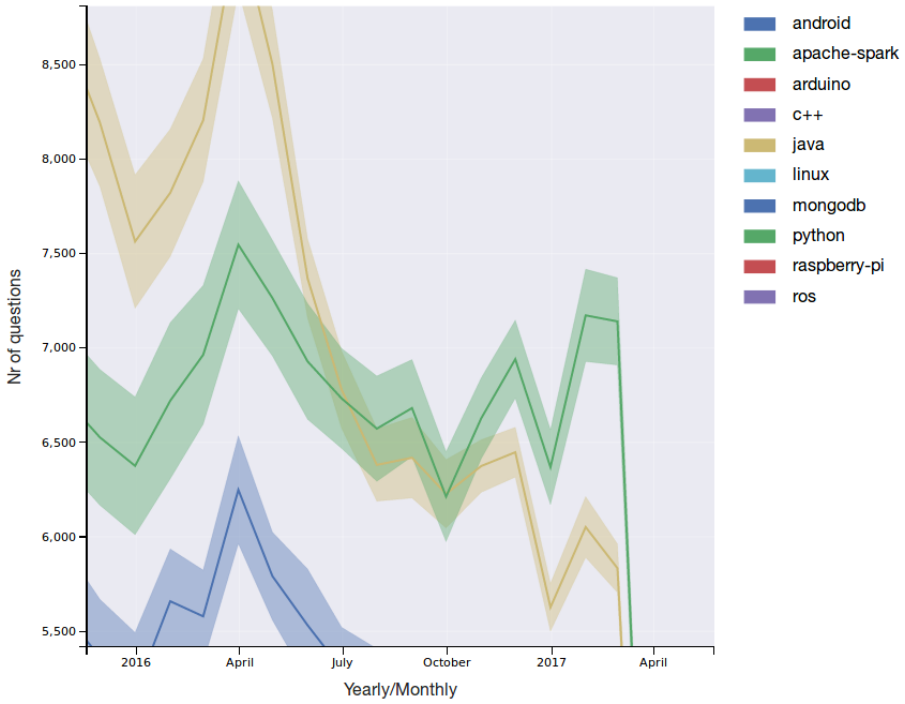
It is interesting to see how important the Java language has been from the beginning of stackoverflow, and with the appearance of Android, it became even stronger:



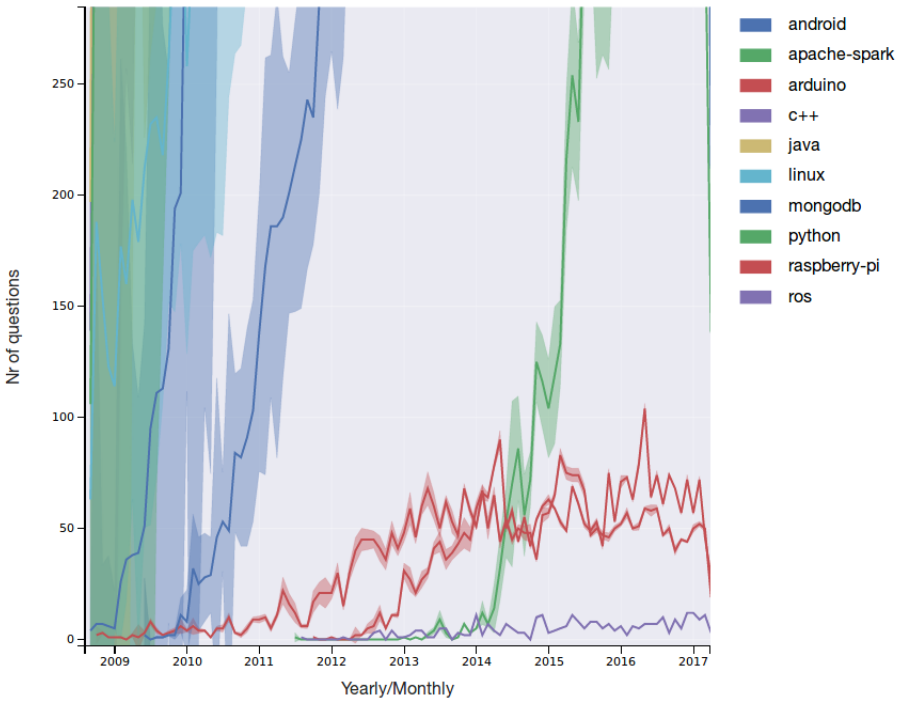
Besides, let us check how Python has been struggling for the last couple of years and it only overtakes Android by February-March 2015,



and overtakes Java by July-2016:

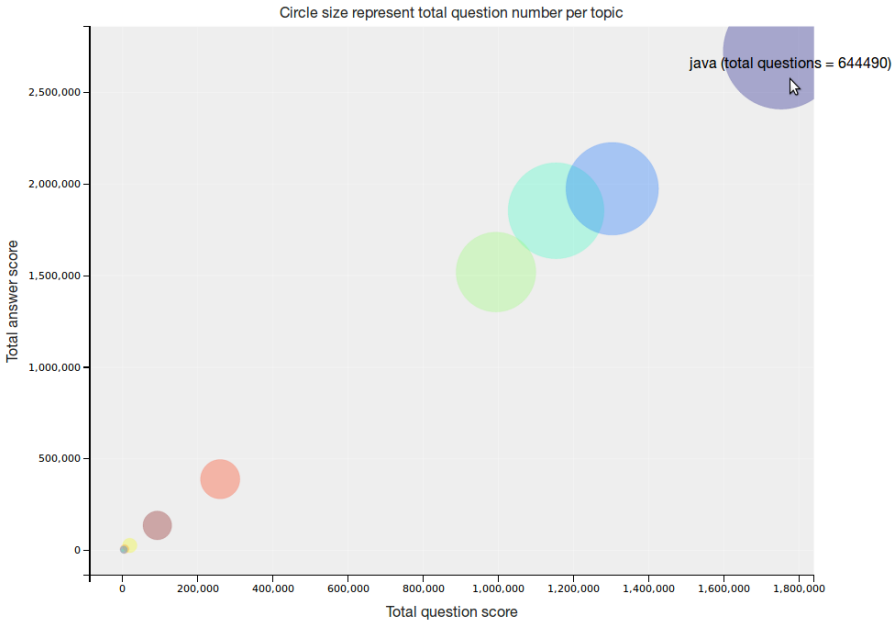


Finally, if we take a look to the lowest part of the plot, we can discover an approximate time when each technology has appeared (at least for those who appeared after the beginning of stackoverflow):



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):


Circle size represent total question number per topic
java (total questions = 644490)
Total answer score
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
0
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000
1,600,000
1,800,000
Total question score

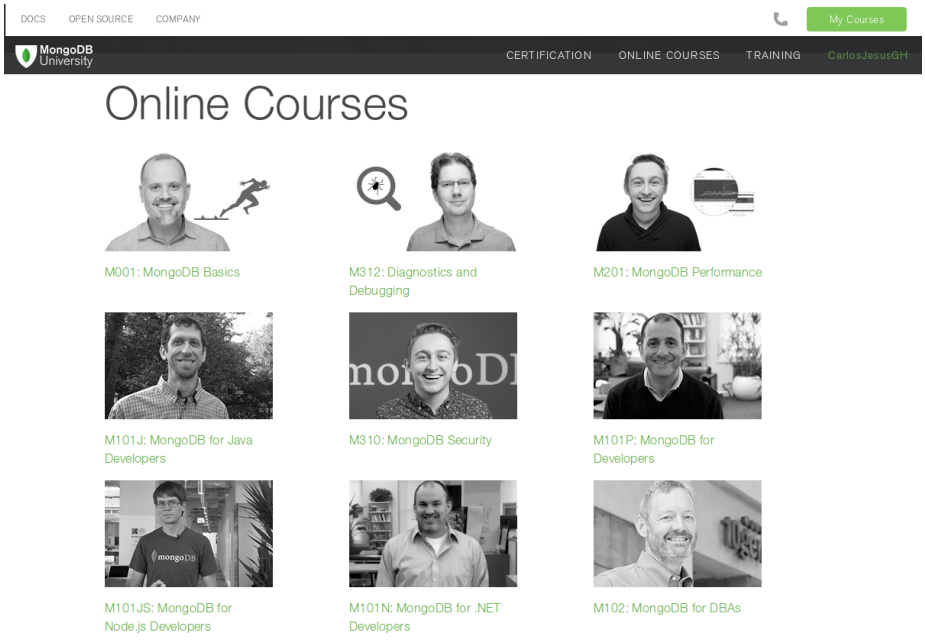
7 Possible improvements

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:

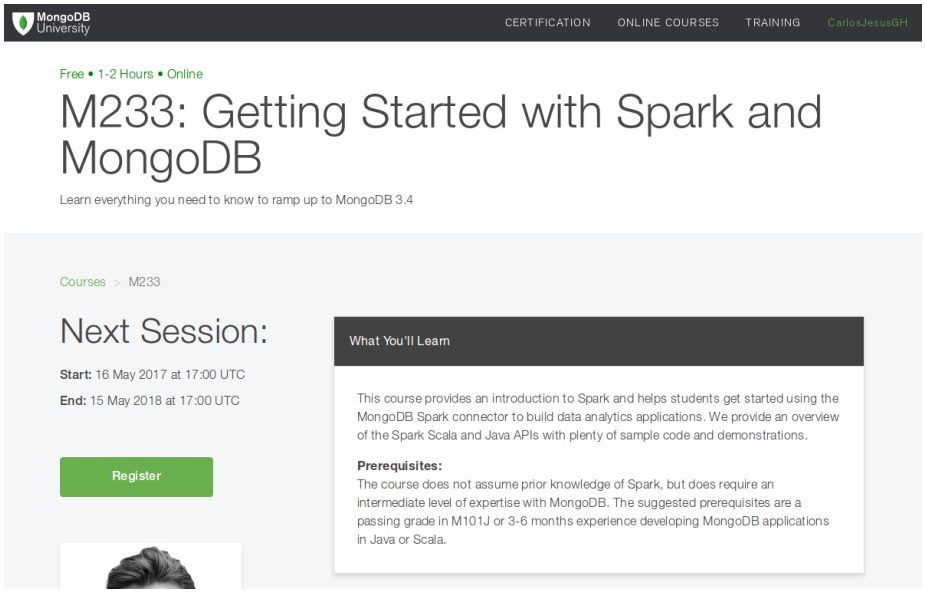
Using a database: databases are good to organize, index and search into big amounts of data, much of the time the performance is much better querying into databases compared to simple text files. A great option for this purpose is the NoSql Json-like database MongoDB [10].

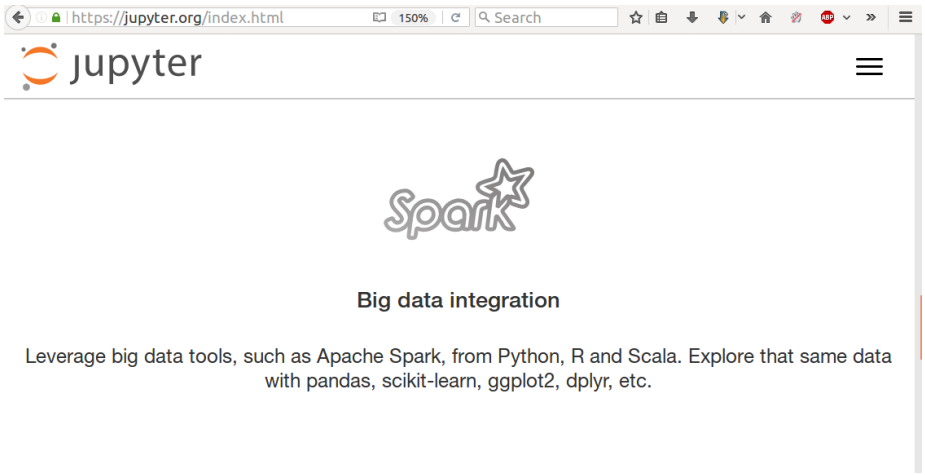
MongoDB is free, open source and also include some free courses in the official website:



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^{TM} 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.



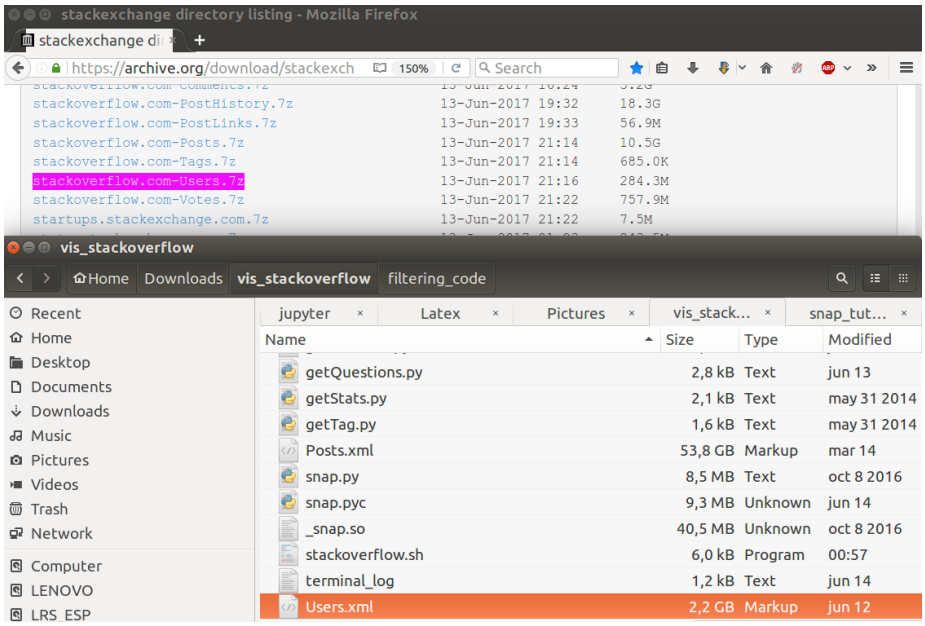


8 Conclusions

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.



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.eduprojsnap-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].