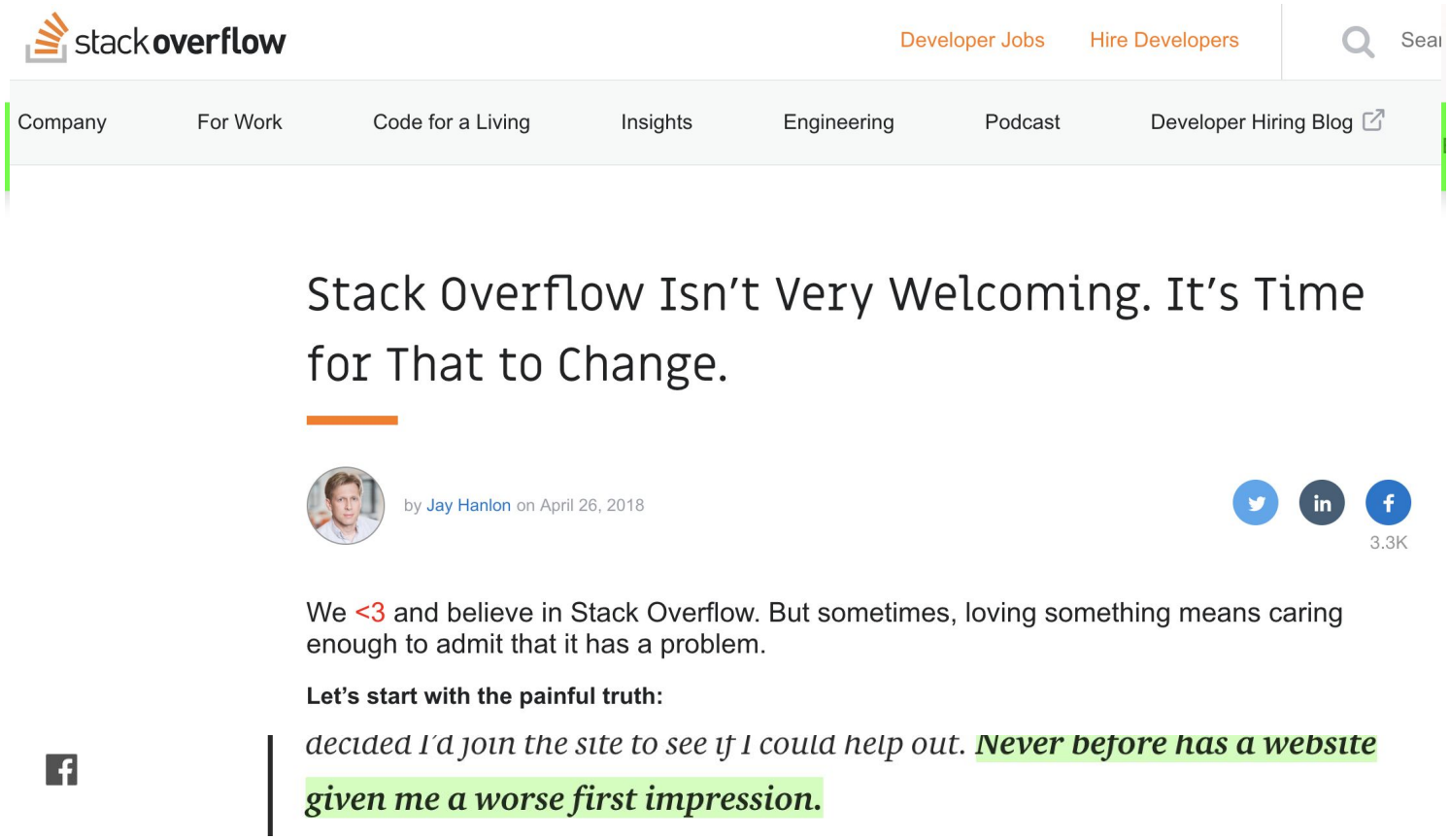
```
#include <stdio.h>
#include “CAPP 30123 @ UChicago”
#include “Group name: HackyStacks”
#include “Group members: Adam Shelton,  Dhruval Bhatt,  Li Liu,  Sanittawan Tan”

int main(int argc, char** argv) {

    printf(“Hello, world!”);
    printf(“Exploring the Sentiment of the Developer
                Community’s Responses ”);

}
```

stackoverflow

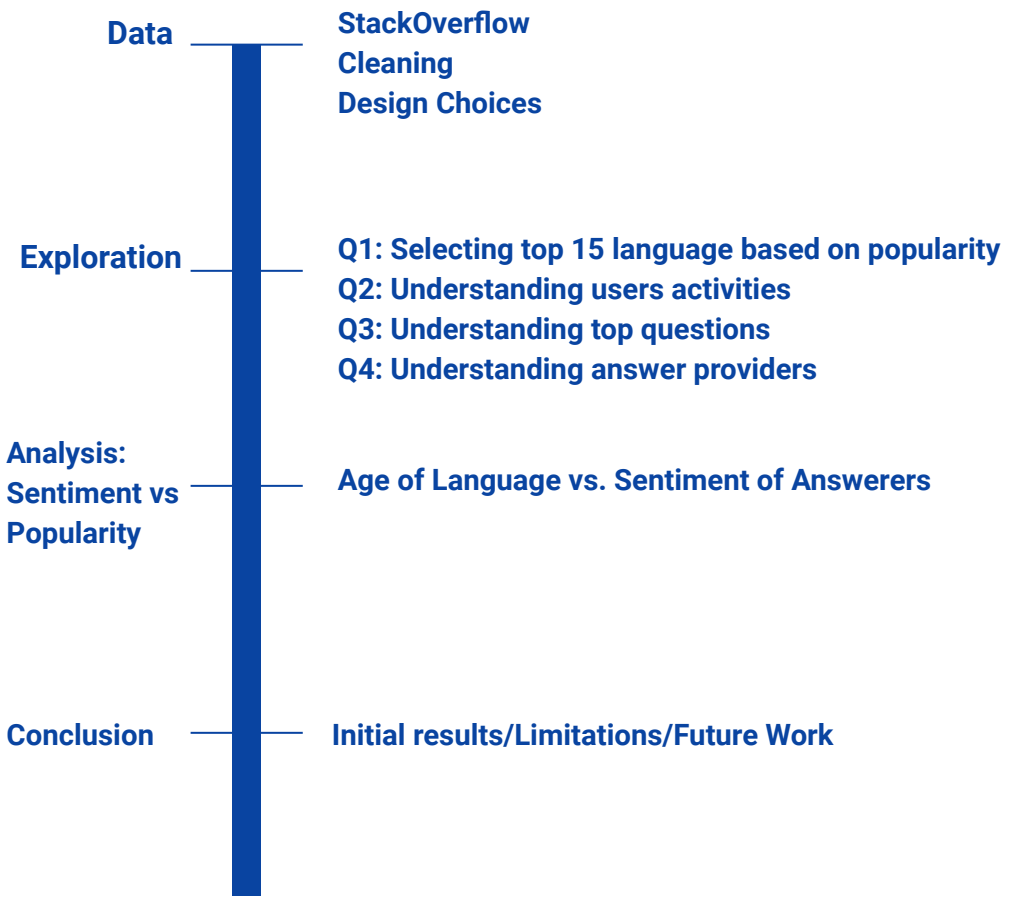
Research Question and Hypothesis

- **Question**: As programming languages (and their users) mature and become more popular, do StackOverflow answer providers become more *impatient and meaner* to answer seekers of those languages?
- **For measurability, our refined question**: As programming languages mature, have sentiments of the answers become *more negative*?
- **Hypothesis:** We expect StackOverflow answer providers to become more negative (~meaner, less patient) when answering questions related to those languages as programming languages become more popular.

Research Inspiration



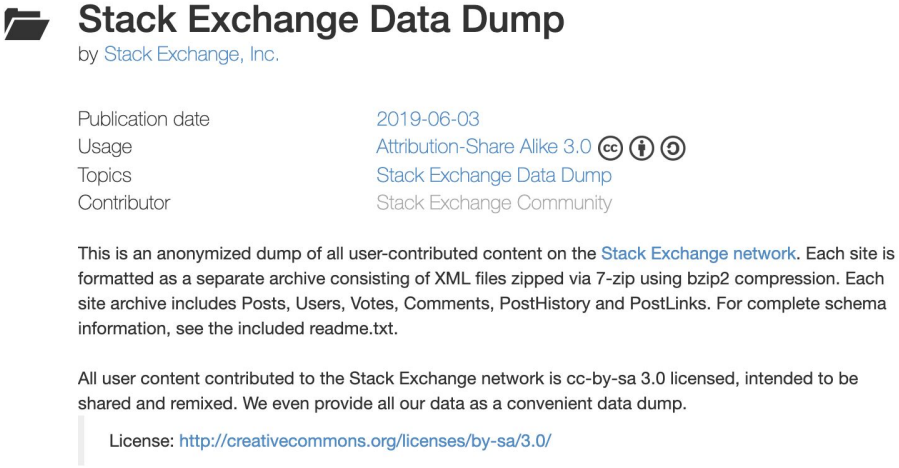
Roadmap


Data
StackOverflow
Cleaning
Design Choices
Exploration
Q1: Selecting top 15 language based on popularity
Q2: Understanding users activities
Q3: Understanding top questions
Q4: Understanding answer providers
Analysis:
Sentiment vs
Popularity
Age of Language vs. Sentiment of Answerers
Conclusion
Initial results/Limitations/Future Work

Data

Source: Stack Exchange Data Dump

https://archive.org/details/stackexchange



Stack Exchange Data Dump

by Stack Exchange, Inc.

Publication date	2019-06-03
Usage	Attribution-Share Alike 3.0
Topics	Stack Exchange Data Dump
Contributor	Stack Exchange Community

This is an anonymized dump of all user-contributed content on the Stack Exchange network. Each site is formatted as a separate archive consisting of XML files zipped via 7-zip using bzip2 compression. Each site archive includes Posts, Users, Votes, Comments, PostHistory and PostLinks. For complete schema information, see the included readme.txt.

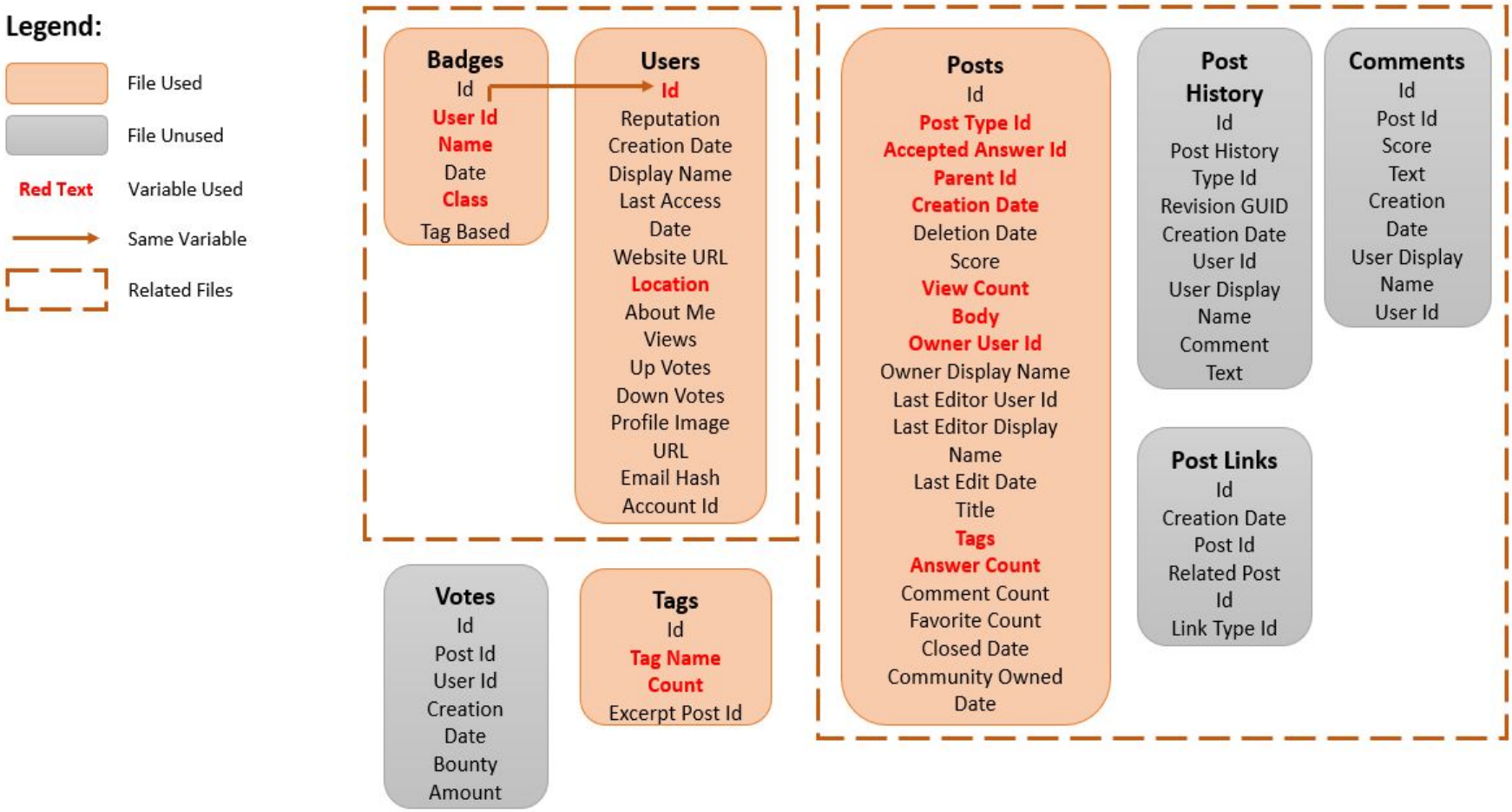
All user content contributed to the Stack Exchange network is cc-by-sa 3.0 licensed, intended to be shared and remixed. We even provide all our data as a convenient data dump.

> License: http://creativecommons.org/licenses/by-sa/3.0/

Data

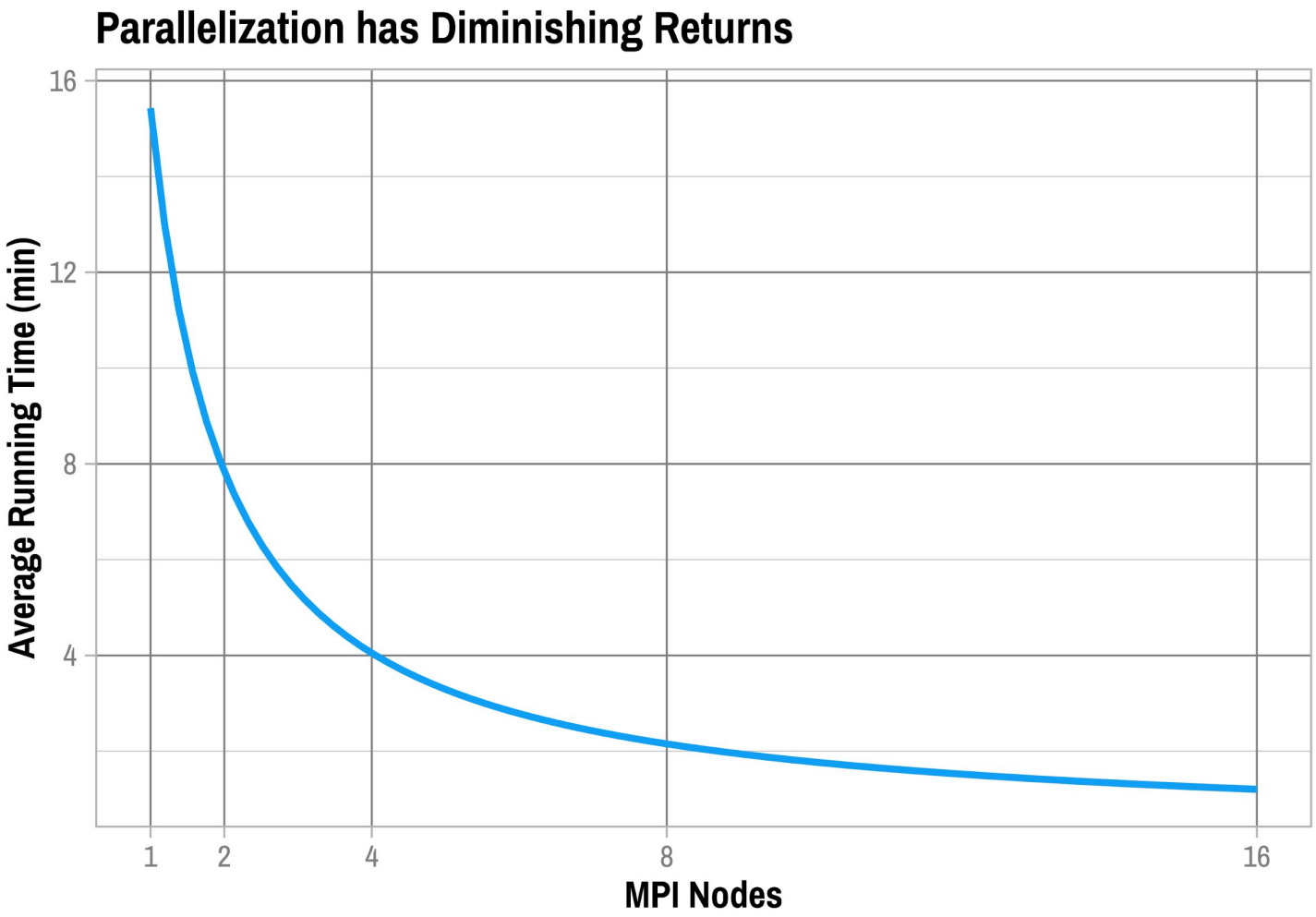
No.	File name	Type	Size	Lines	CSV File Size
1.	Tags	XML	4.7 MB	54,467	5.6 MB
2.	Post Links	XML	667 MB	5.7 mil.	NA
2.	Users	XML	3.1 GB	10 mil.	1.79 GB
3.	Badges	XML	3.4 GB	30.3 mil.	3.28 GB
4.	Votes	XML	16 GB	167 mil.	NA
5.	Comments	XML	18 GB	72 mil.	NA
6.	Posts	XML	66 GB	43.9 mil.	22.5 GB
7.	Post History	XML	113 GB	< inf	NA

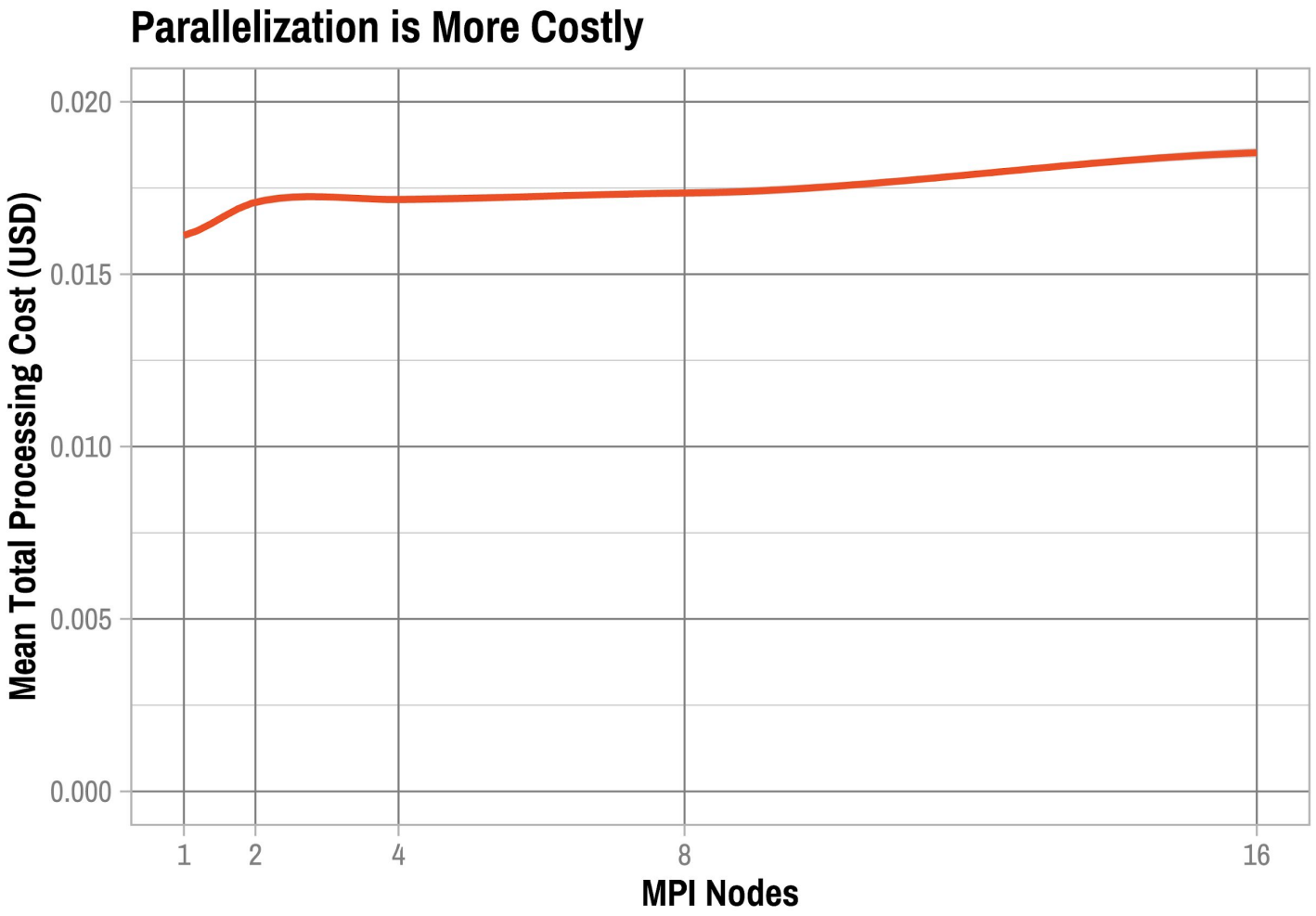
Data Scheme

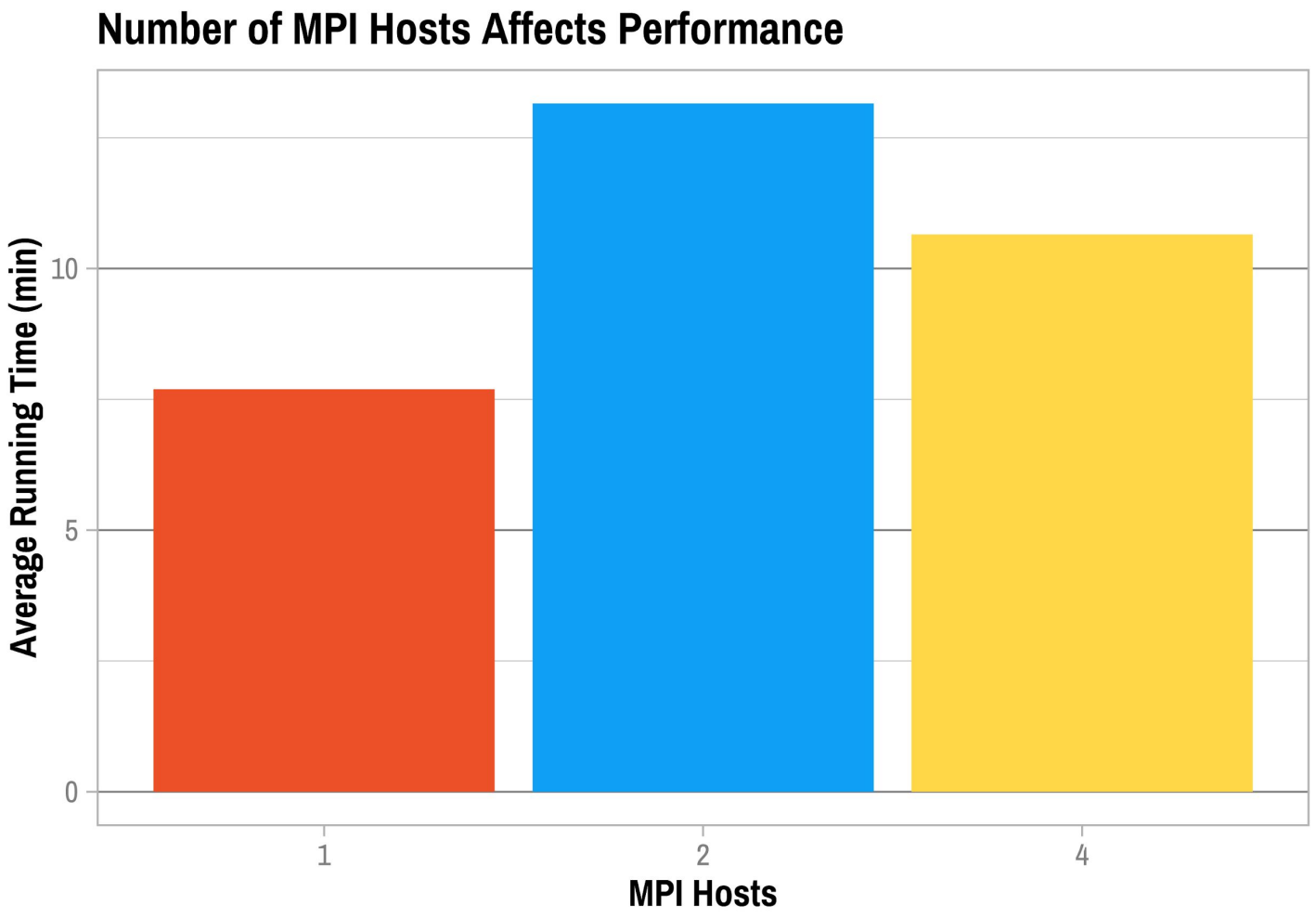

Legend:
File Used
File Unused
Red Text
Variable Used
Same Variable
Related Files
Badges
Id
User Id
Name
Date
Class
Tag Based
Users
Id
Reputation
Creation Date
Display Name
Last Access Date
Website URL
Location
About Me
Views
Up Votes
Down Votes
Profile Image URL
Email Hash
Account Id
Posts
Id
Post Type Id
Accepted Answer Id
Parent Id
Creation Date
Deletion Date
Score
View Count
Body
Owner User Id
Owner Display Name
Last Editor User Id
Last Editor Display Name
Last Edit Date
Title
Tags
Answer Count
Comment Count
Favorite Count
Closed Date
Community Owned Date
Post History
Id
Post History Type Id
Revision GUID
Creation Date
User Id
User Display Name
Comment Text
Comments
Id
Post Id
Score
Text
Creation Date
User Display Name
User Id
Post Links
Id
Creation Date
Post Id
Related Post Id
Link Type Id
Votes
Id
Post Id
User Id
Creation Date
Bounty Amount
Tags
Id
Tag Name
Count
Excerpt Post Id

Data Cleaning Challenges and Design Choices

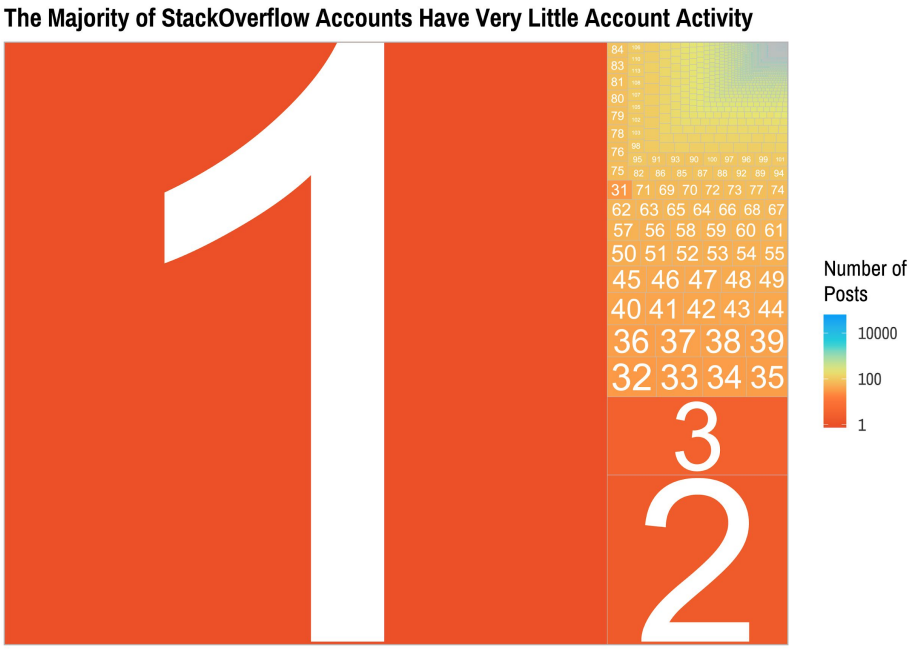
- It was necessary to **convert XML files to CSV** files for analysis
- Large data makes performing even simple tasks difficult
  - Parsing an XML file needed to be performed line by line to avoid memory restrictions and allow for easy parallelization
  - XML is a markup language similar to HTML which encloses data in starting and ending tags - this makes splitting the data difficult
- Once code was developed and tested to perform this process, it could be altered to work with MPI to accelerate the process
  - However this led to more questions...


Parallelization has Diminishing Returns
16
12
8
4
Average Running Time (min)
1
2
4
8
16
MPI Nodes


Parallelization is More Costly
Mean Total Processing Cost (USD)
0.020
0.015
0.010
0.005
0.000
1
2
4
8
16
MPI Nodes


Number of MPI Hosts Affects Performance
Average Running Time (min)
0
5
10
1
2
4
MPI Hosts

Exploratory Analysis: Active Users



Most users have very minimal activities

Data Used:

- Posts.csv (22.5 GB)

Techniques/Tools Used:

- Apache Spark's Dataframe
- Count the number of times each unique user ID appears in the data set

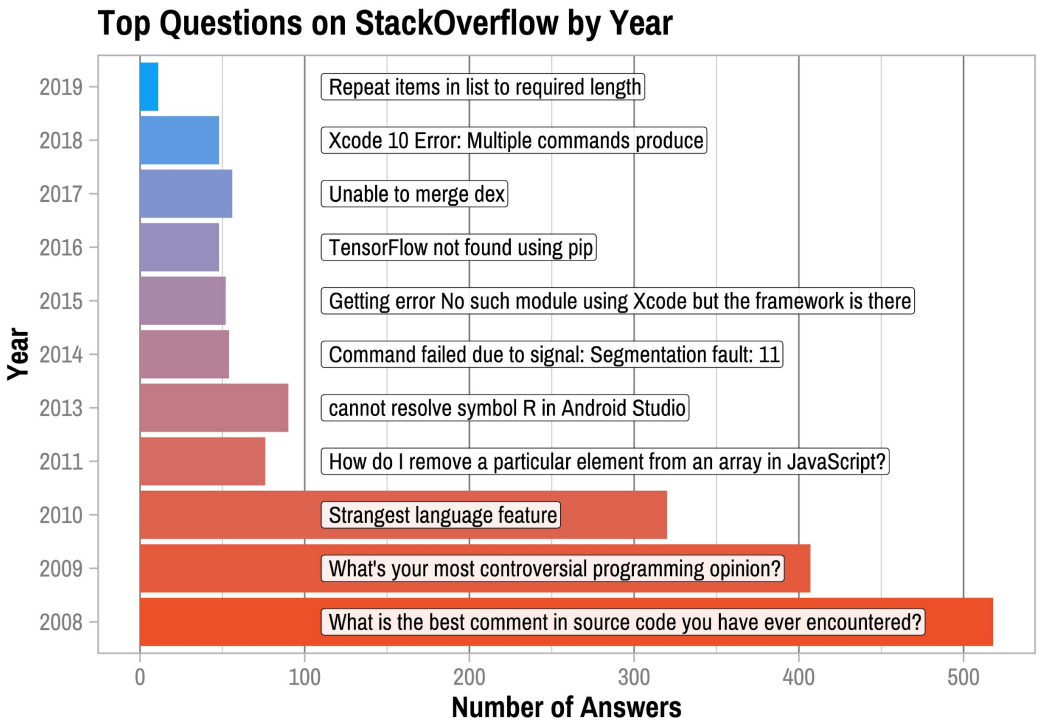
Challenges Faced:

- Syntax learning curve

Run time:

- ~ 2 minutes 45 seconds on Google Dataproc with 4 workers

Exploratory Analysis: Most Answered Question By Year



Type of top question asked changes as site has matured:
From open ended to specific issue

Data Used:

- Posts.csv (22.5 GB)

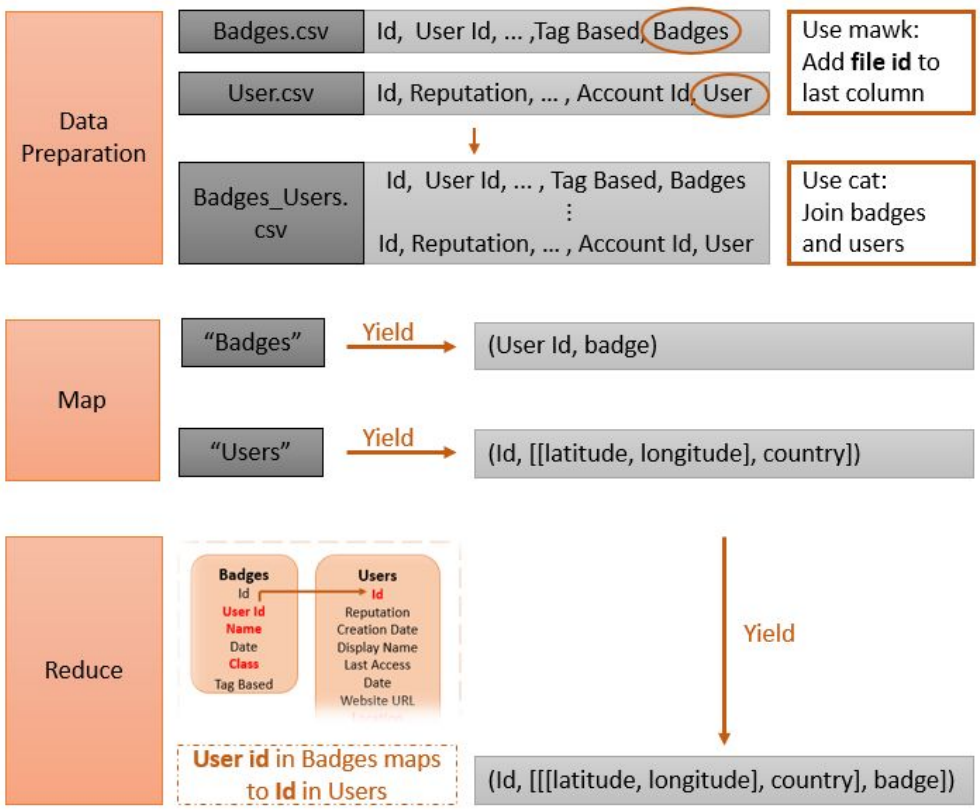
Techniques/Tools Used:

- Used mrjob to yield answer count and title of question for each year

Challenges Faced:

- New line “\n” characters left after data cleaning

Exploratory Analysis: Locations of Top Answerer



Data Used:

- Users.csv (1.79 GB) and Badges.csv (3.28 GB)

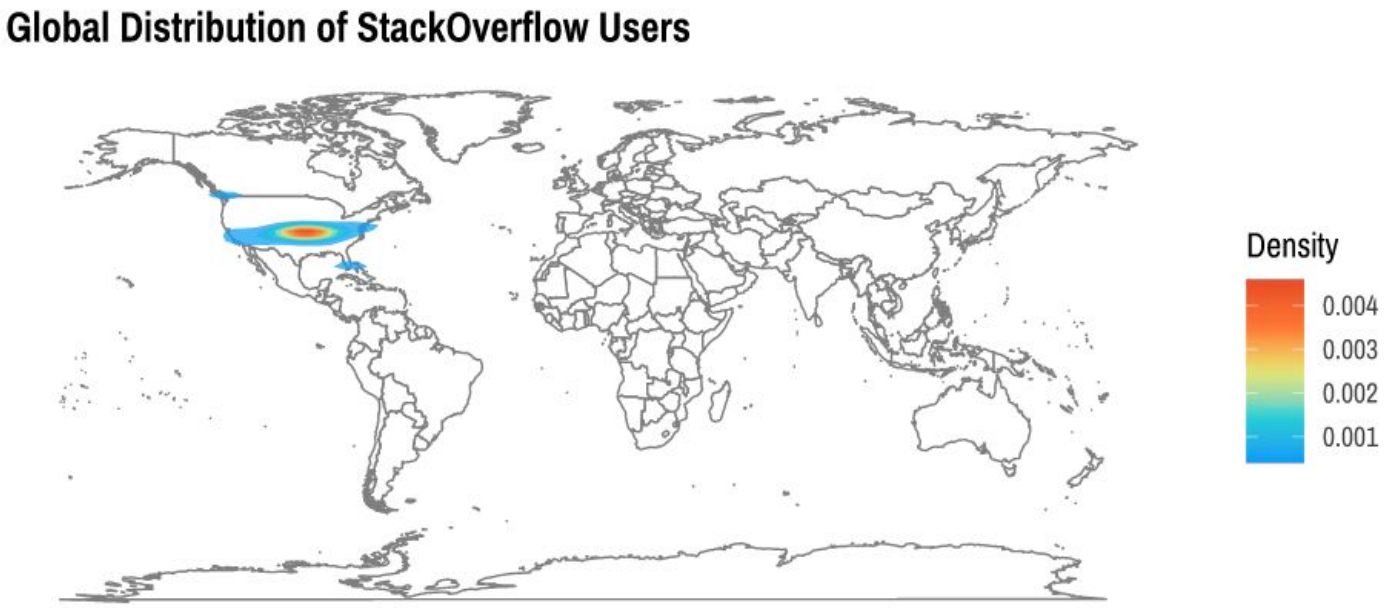
Techniques/Tools Used:

- Update each file using mawk and cat files - done with shell script
- Used mrjob to yield location coordinates and country name for user id of those with illuminator badge
- Transformed user input location to coordinates and country

Challenges Faced:

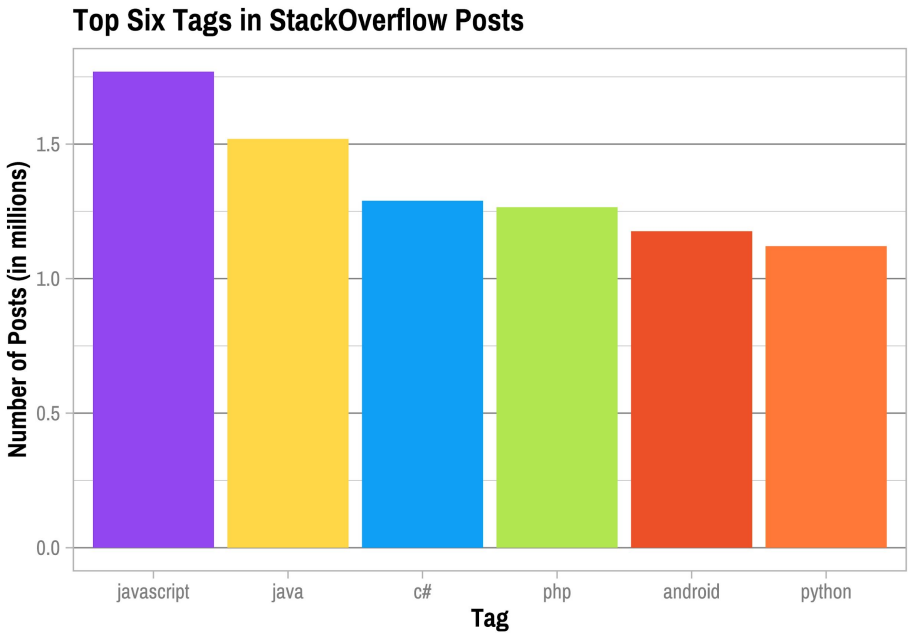
- Configuring external package
- Running on clusters

Exploratory Analysis: Locations of Top Answerer



From subset: Users that have most actively answered are predominantly from North AM region

Exploratory Analysis: Most Mentioned Tags



Data Used:

- Tags.csv (5.6 MB)

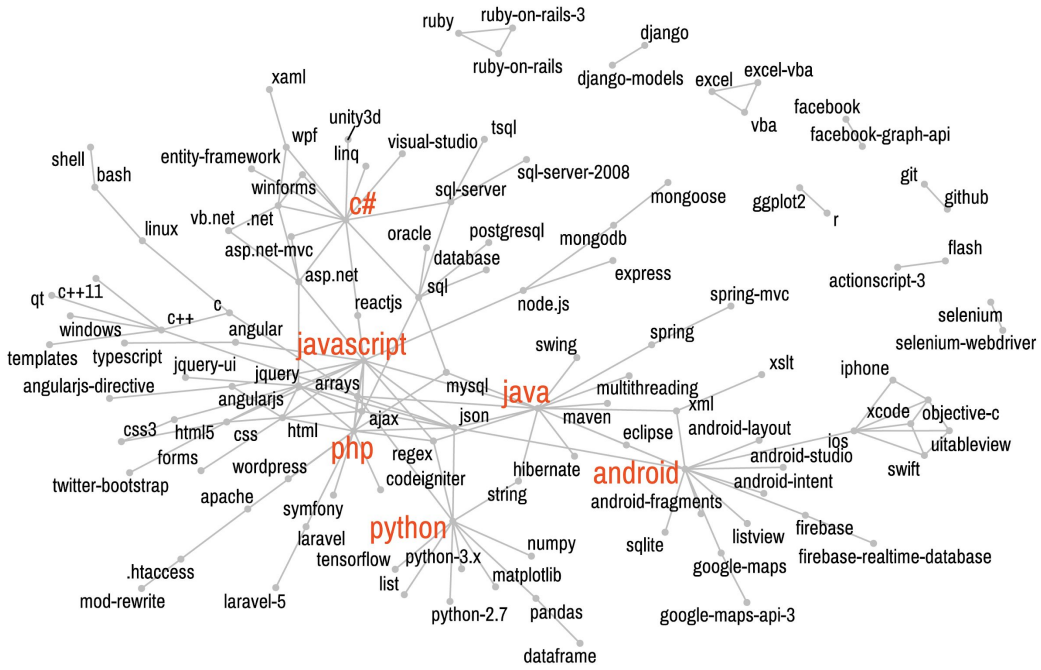
Techniques/Tools Used:

- Use mrjob to yield the count for a given tag
- Sort tags with highest count using heap and mrjob

Top most tagged language is Javascript

Exploratory Analysis: Network of Tags

Connections Between Tags in StackOverflow Posts



Popular Languages form central nodes

Data Used:

- Posts.csv (22.5 GB)

Techniques/Tools Used:

- Create bigram with adjacent tags of questions
- Use mrjob to yield the count of a pair of tags

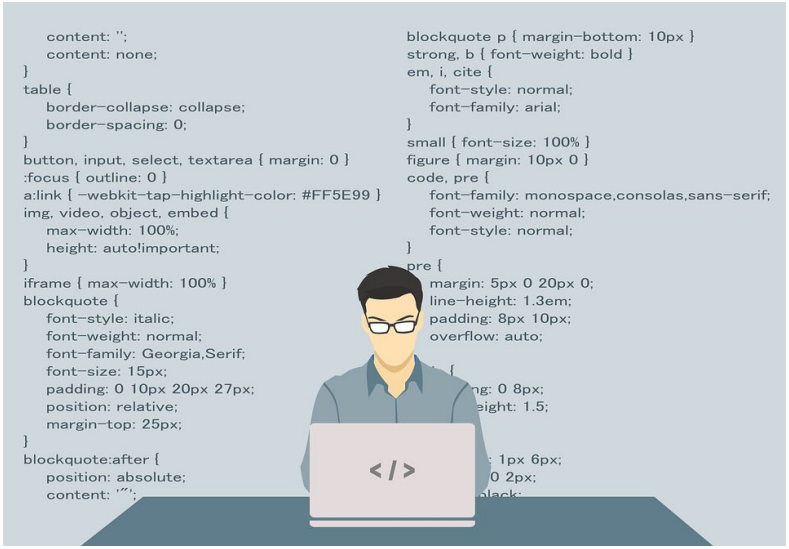
How do the answerers' sentiment correlate with the popularity of the language?

Popularity:

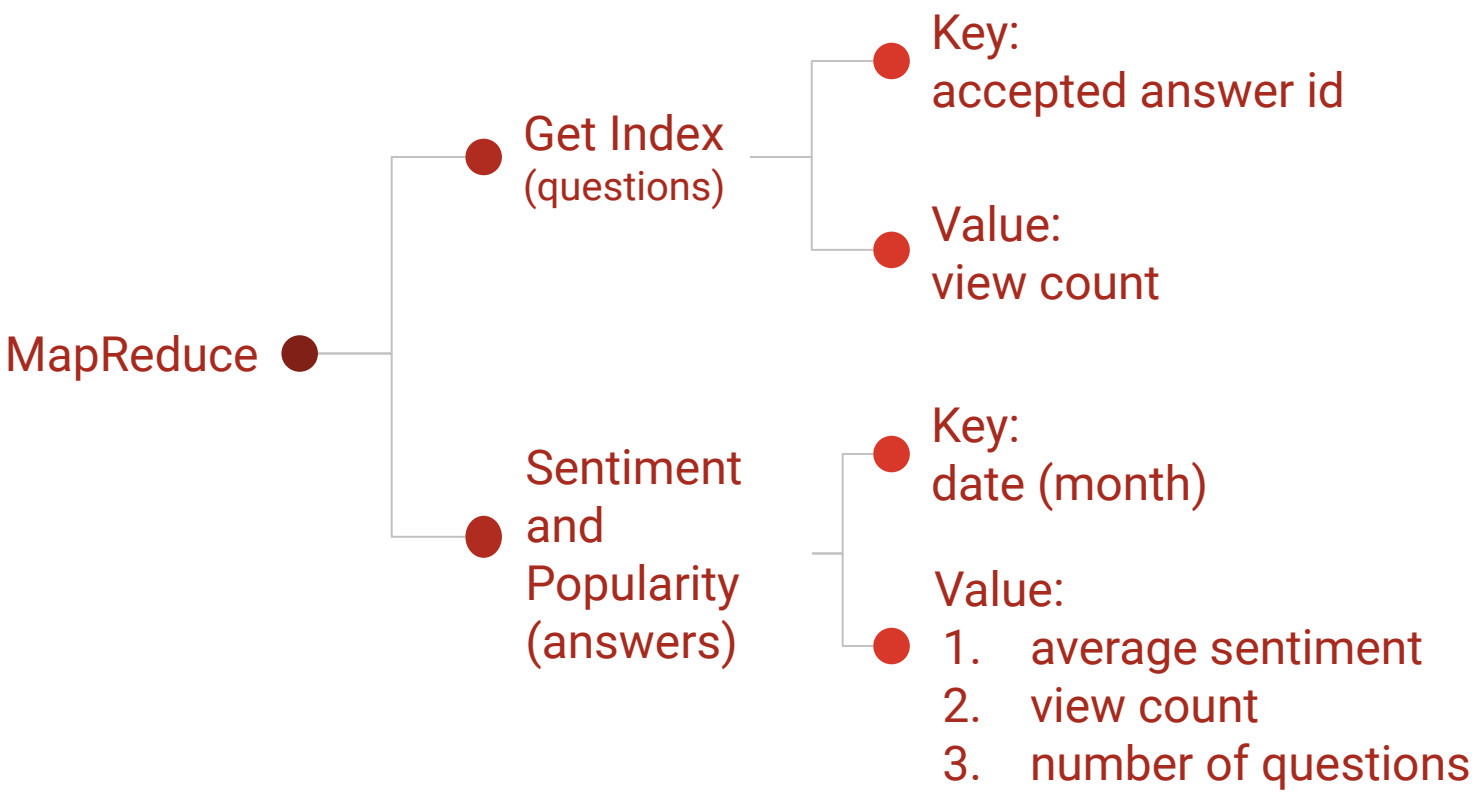
1. View Counts (Detrended)
2. Number of Questions

Sentiment:

VADER sentiment analysis (-1 to 1)

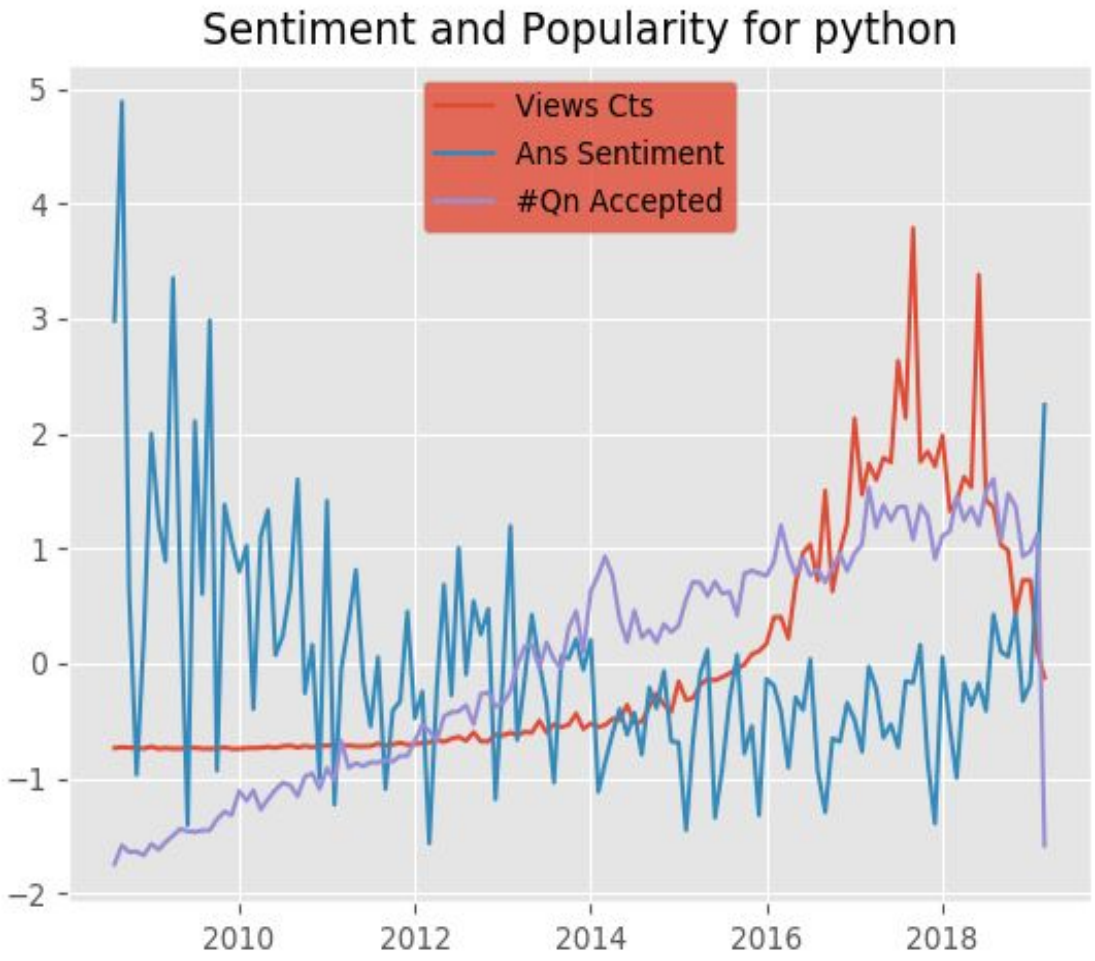


Getting Data



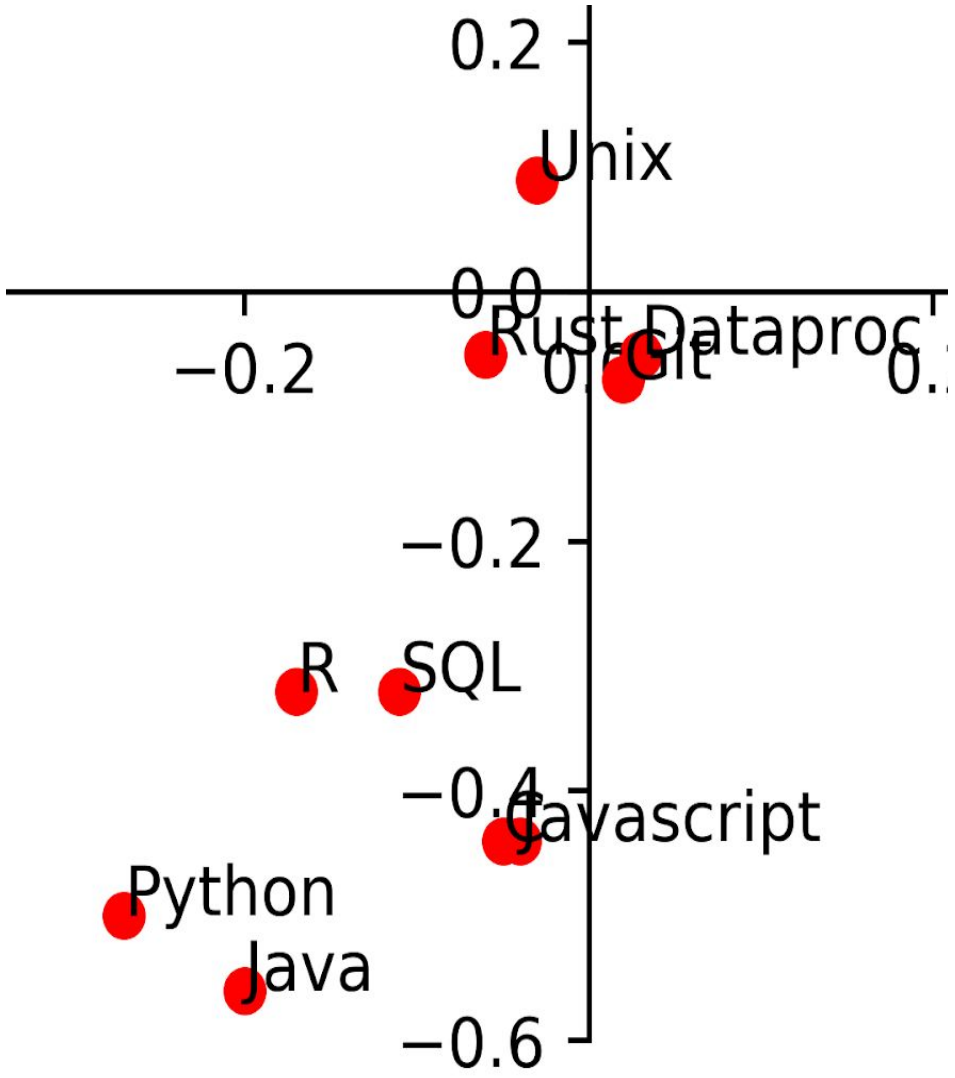
Visualization (Python Example)

Links to 10 Languages



Correlation Plots

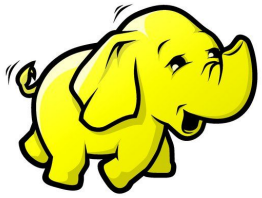
Language	View Counts v.s. Sentiment	Questions Counts v.s. Sentiment
C	-0.05	-0.44
Dataproc	0.03	-0.05
Git	0.02	-0.07
Java	-0.2	-0.56
Javascript	-0.04	-0.44
Python	-0.27	-0.5
R	-0.17	-0.32
Rust	-0.06	-0.05
SQL	-0.11	-0.32
Unix	-0.03	0.09
Mean(Std)	-0.088 (0.098)	-0.266 (0.228)



Conclusion

- Results show some preliminary evidence for the hypothesis
- Programming languages communities are more dynamic.
- Big data tools really help ease the analysis. (~ run time on Dataproc > 1 hours)

MRJob



Hadoop

DataProc

MPI

Spark

Images Source: Google Images