Make dat $$$

PROCESS BOOK

Project URL	: https://atsai1220.github.io/cs5630-team-project/
Repository	: https://github.com/atsai1220/cs5630-team-project
Data Source	: https://insights.stackoverflow.com/survey/

EDWARD EKSTROM
u0579294

ANDREW TSAI
u1038596

FARHAN NASRULLAH
u1146617

Version: 1.0
Date: 02 Dec 2017

Overview

Our project is not related to hip hop! **Make dat $$$** is actually a data visualization designed to capture an essence of the global software development scenario across the world. We used last four years of Stackoverflow's annual developer survey results to capture the insights from participating professional and aspiring developers around the world. This visualization is produced as the final project for CS 5630/6630 "Visualization for Data Science" course at the University of Utah.

Background

Launched in 2008. Stack Overflow is a popular website that serves as a platform for users to ask and answer questions on topics mostly related to computer programming and systems. Internet users, through membership and active participation, get to vote questions and answers up or down and edit questions and answers in a fashion similar to a Wiki or Digg. It is one of the largest platform of its kind, with more than 2 million registered users, most of whom are broadly interested, learning or working in the domain of software development.
Each year since 2011, Stack Overflow has asked developers about their favorite technologies, coding habits, and work preferences, as well as how they learn, share, and level up. The survey results are quite significant and interesting as they reveal a lot of insights on the state of the industry, technology and trends.

Motivation

As students and aspiring developers awaiting to enter the tech/development industry, we are constantly thinking about our possible careers. How much money do I get paid? Which languages are most coveted? Which technologies are rising in demand? Which job allows

more remote work? Does PhD actually pay more? Some of us students will probably go back to our country after finishing studies in the US, what's the trend in our home countries?
Since Stackoverflow captures a lot of these information in their surveys, we based our visualization project on results of the Stack Overflow survey for the last four years, 2014 to 2017.

One could easily ask why not take a look at the survey results directly at Stackoverflow? Sure, it can be done but there are a number of problems with that:

1. *Static results*

   The results are only published a year at a time.
2. *Incoherent report style*

   Data over the years are published in different formats over the years.
3. *No trends shown*

   Year wise results do not show any trends in data over the years
4. *No scope of interaction*

   Users do not have any scope of interaction with various factors and data from the survey results.

We address these issues present the results in a single webpage and allow users to choose between certain variables to see results and trends over the years, *with average salaries as the focus of the visualization.*

We want to help students make informed decisions about their future careers as we think that most students have a nebulous understanding of the varieties of opportunities that exist, much less the relative wages they might earn.

Objectives

In a nutshell, our objective are to present:

i. The relation of average salary with various factors like programming language, which country you are from, formal education and degree, work role etc.
ii. Allow the user to select their topic of interest and see its trend with average salaries over the years.
iii. Provide an easy to use and interactive interface for students and developers alike to make informed decision regarding their career.

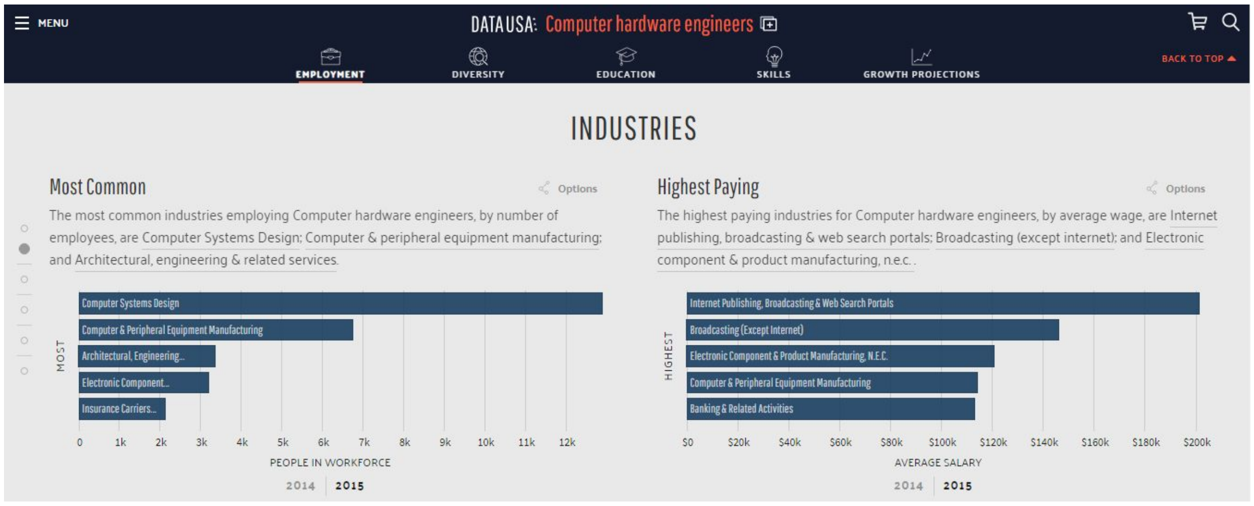
Even if we can help one student make a better choice for their future based on this visualization, we will consider our effort successful! Making a data-driven decision regarding career choice will result in:

- Greater job satisfaction.
- Higher salary.
- Greater potential for career growth and advancement.
- Cost and time savings by avoiding low-return educational/learning investments.

Related Work and Inspiration

We could not find any comparable resources online which captured responses from *actual* software developers and students, with a substantial magnitude over several years as Stackoverflow did. It can be said that the Stackoverflow 2017 survey results intrigued us into wanting to do deeper and subsequently choosing this as the topic for our project.

Once we started ideating on our project, we found datausa.io to do a nice job in presenting data of several sectors in a very informative and useful manner.


MENU
DATA USA: Computer hardware engineers
BACK TO TOP
EMPLOYMENT
DIVERSITY
EDUCATION
SKILLS
GROWTH PROJECTIONS
INDUSTRIES
Most Common
Options
Computer Systems Design
Computer & Peripheral Equipment Manufacturing
Architectural, Engineering...
Electronic Component...
Insurance Carriers...
MOST
0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k 11k 12k
PEOPLE IN WORKFORCE
2014 2015
Highest Paying
Options
Internet Publishing, Broadcasting & Web Search Portals
Broadcasting (Except Internet)
Electronic Component & Product Manufacturing, N.E.C.
Computer & Peripheral Equipment Manufacturing
Banking & Related Activities
HIGHEST
$0 $20k $40k $60k $80k $100k $120k $140k $160k $180k $200k
AVERAGE SALARY
2014 2015

The most common industries employing Computer hardware engineers, by number of employees, are Computer Systems Design; Computer & peripheral equipment manufacturing; and Architectural, engineering & related services.

The highest paying industries for Computer hardware engineers, by average wage, are Internet publishing, broadcasting & web search portals; Broadcasting (except internet); and Electronic component & product manufacturing, n.e.c..

Fig: Bar charts in datausa.io to present computer hardware engineer statistics

Data

The data we used were available year-wise in csv format in Stackoverflow website. We used data from 2014 to 2017 to keep the analysis limited to last four years. The files can be download from https://insights.stackoverflow.com/survey/

A quick merging of the yearly showed that there were about 150,000 responses altogether.

Data Processing

As much we were excited with the richness of the 2017 survey results, we faced significant challenges once we started to work with the data.

i. The questions varied tremendously over the years.
ii. The responses to similar questions were extremely different and not standardized.
iii. The Column/Field names in the csv files were dissimilar across years.
iv. Ordering of columns representing same data were different in each csv file.

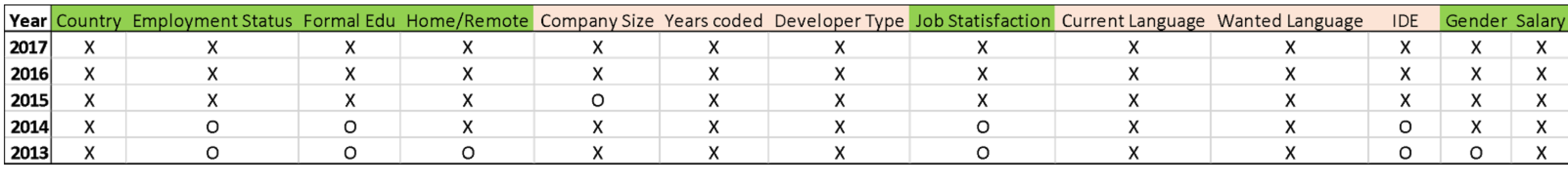
This made it quite difficult for us to present trends of all the different results at we intended to show. For example, the 2017 and 2016 data had the information if a developer was working full or part time, but the rest of the years did not have this information. This is just case out of many other data fields and thus posed a challenge for us on how to deal with these kind of irregular data. We faced further roadblock with the varieties of responses for the same questions, for example, the latest years had information of average salaries as a single number entry, whereas the earlier years had this as a range value. Column titles for same question or data entries were different too across yearly data. Just to point another area of dissimilarity in brief, where 2017 survey had about 150 columns of responses, the 2016 had only 65 columns!

Therefore, although we initially planned to use python to clean up and standardize our raw data, we could hardly do it because of the randomness of layout and responses. We believe this limited the scope of our project as we intended and stated in our Proposal. Finally we had to resort to the exhaustive process of standardize the exhaustive array of entries for each year individually with build in MS Excel tools, mostly *pivot tables* and *vlookup*. We took 2017 format as the standard format while standardizing the data format of other years.

Exploratory Data Analysis

To start with, we used the 2017 survey data since it was the latest dataset and that we drew our initial inspiration for the project from its insights presented in Stackoverflow. In our brainstorming session prior to submitting the proposal, we mostly ideated our plan for the project based on the data we saw to be available in the 2017 data. As it was revealed later, this was probably not the best idea as we should have considered data of other years too. Upon analysis, we saw that only a few fields were consistent over the years.

The summary of the consistent set of data looks like this:



Year	Country	Employment Status	Formal Edu	Home/Remote	Company Size	Years coded	Developer Type	Job Statisfaction	Current Language	Wanted Language	IDE	Gender	Salary
2017	X	X	X	X	X	X	X	X	X	X	X	X	X
2016	X	X	X	X	X	X	X	X	X	X	X	X	X
2015	X	X	X	X	O	X	X	X	X	X	X	X	X
2014	X	O	O	X	X	X	X	O	X	X	O	X	X
2013	X	O	O	O	X	X	X	O	X	X	O	O	X

*X means data is available and O means data is absent.
**Green marked columns indicated the responses were standardized.

Design evolution

In hindsight, we could contain the essence of our initial design ideas as we shared in our proposal. However, due to deep issues with the yearly data as we already mentioned, we had to forego significant aesthetic efforts and focus on making the visualization more functional to the user. We originally wanted to have the visualization with more variables and levels of interactions, which could not be maintained entirely as the data were inconsistent over the years.

Our initial designs looked like this:

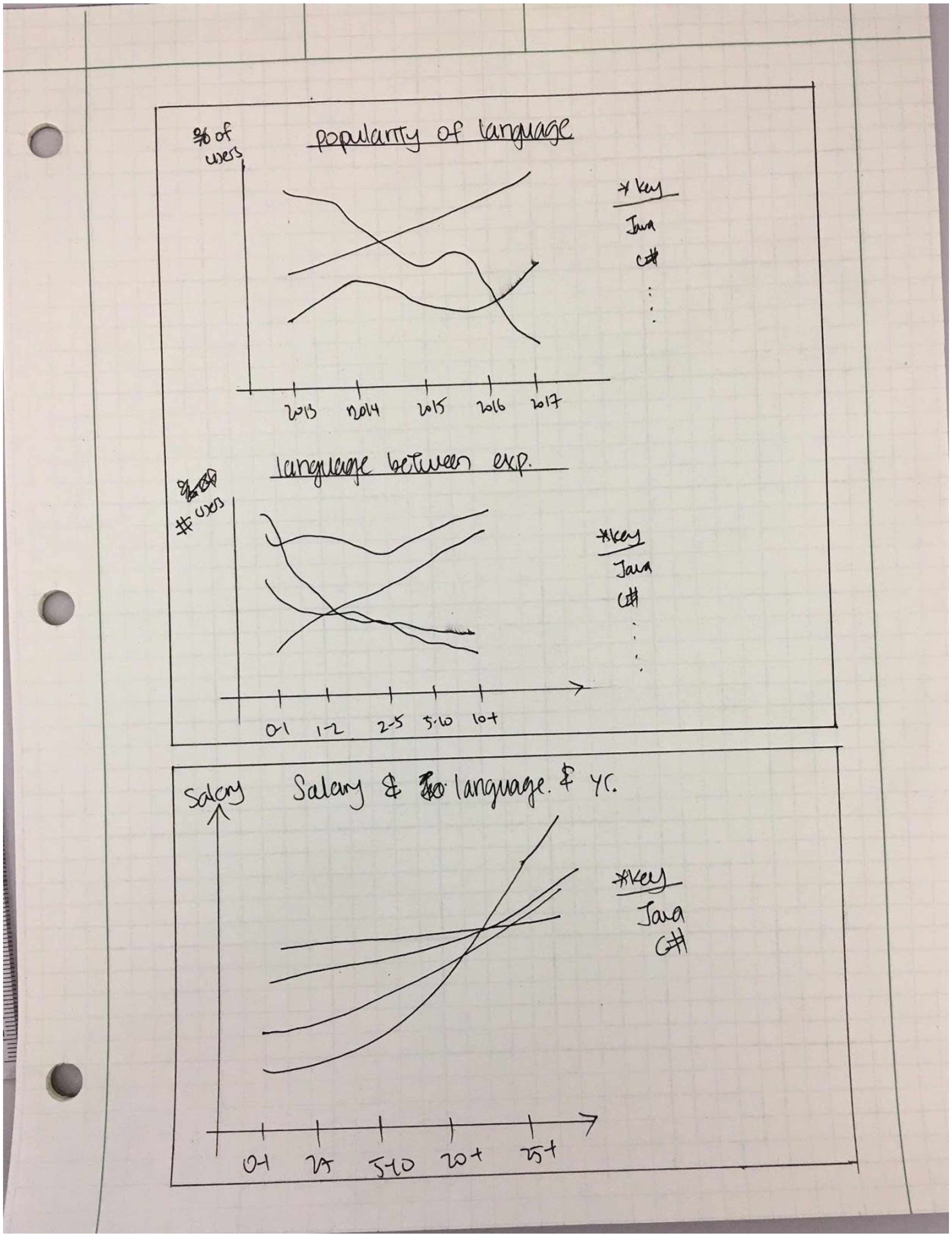


Fig: Design sketched for Proposal

Keeping this in mind and to get started, we started with a layout which had just one section to select the question and a bar chart to represent the answers. *We mistakenly did not keep screenshots of the earlier versions so we are unable to share them here.*

This quickly revealed two things:

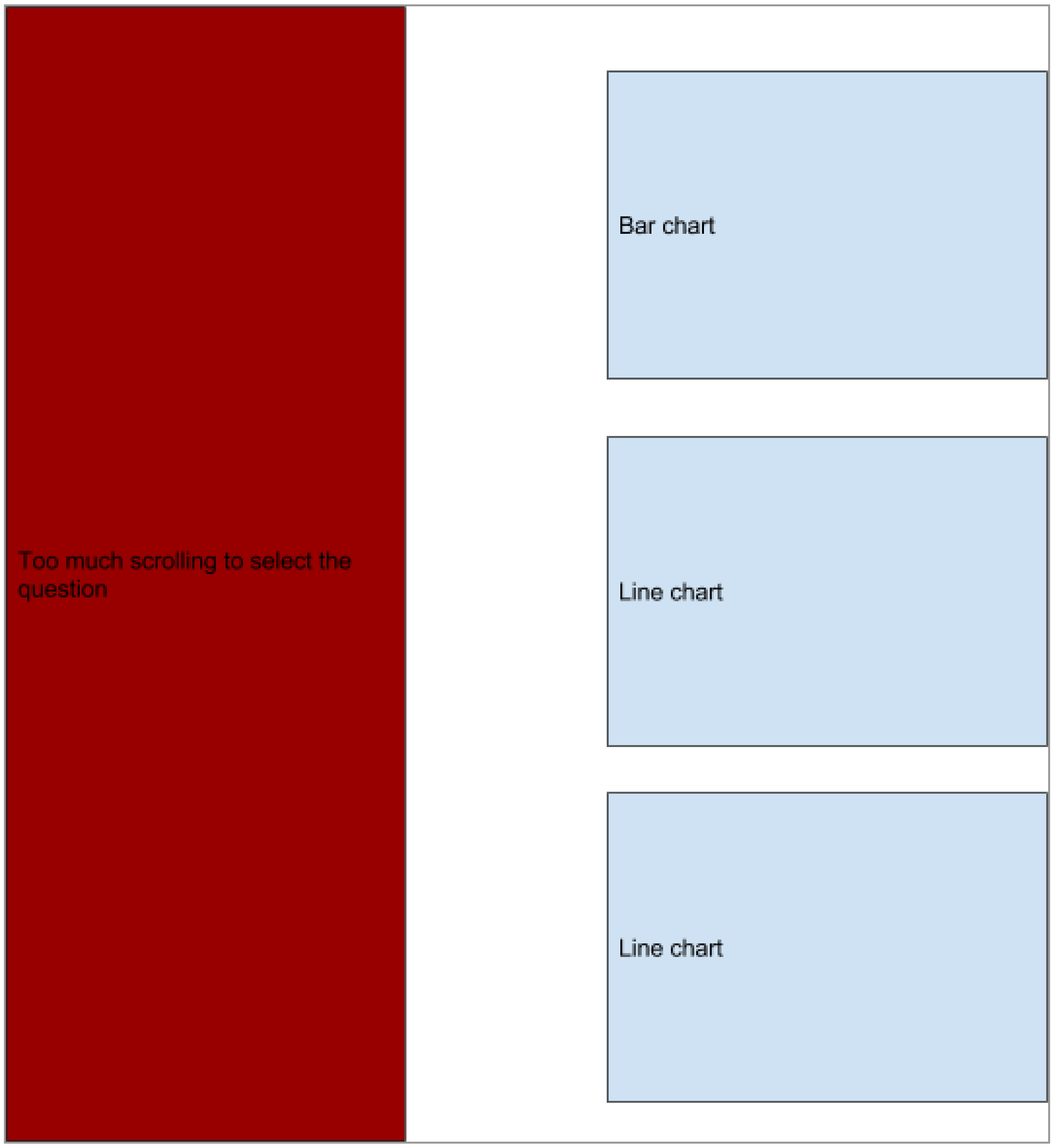
- There were fields whose responses were too varied and large to be accommodated in a single svg area. E.g. Counties which had more than 180 different values.
- We realized that bar charts might not be the best way to show data which were not there for all four years. Also, line charts would be much better to show trends than bar charts.

So we decided to employ bar charts to show the responses within a question, with its length representing the salary for the responses and line charts to show trends of the selections over the years.

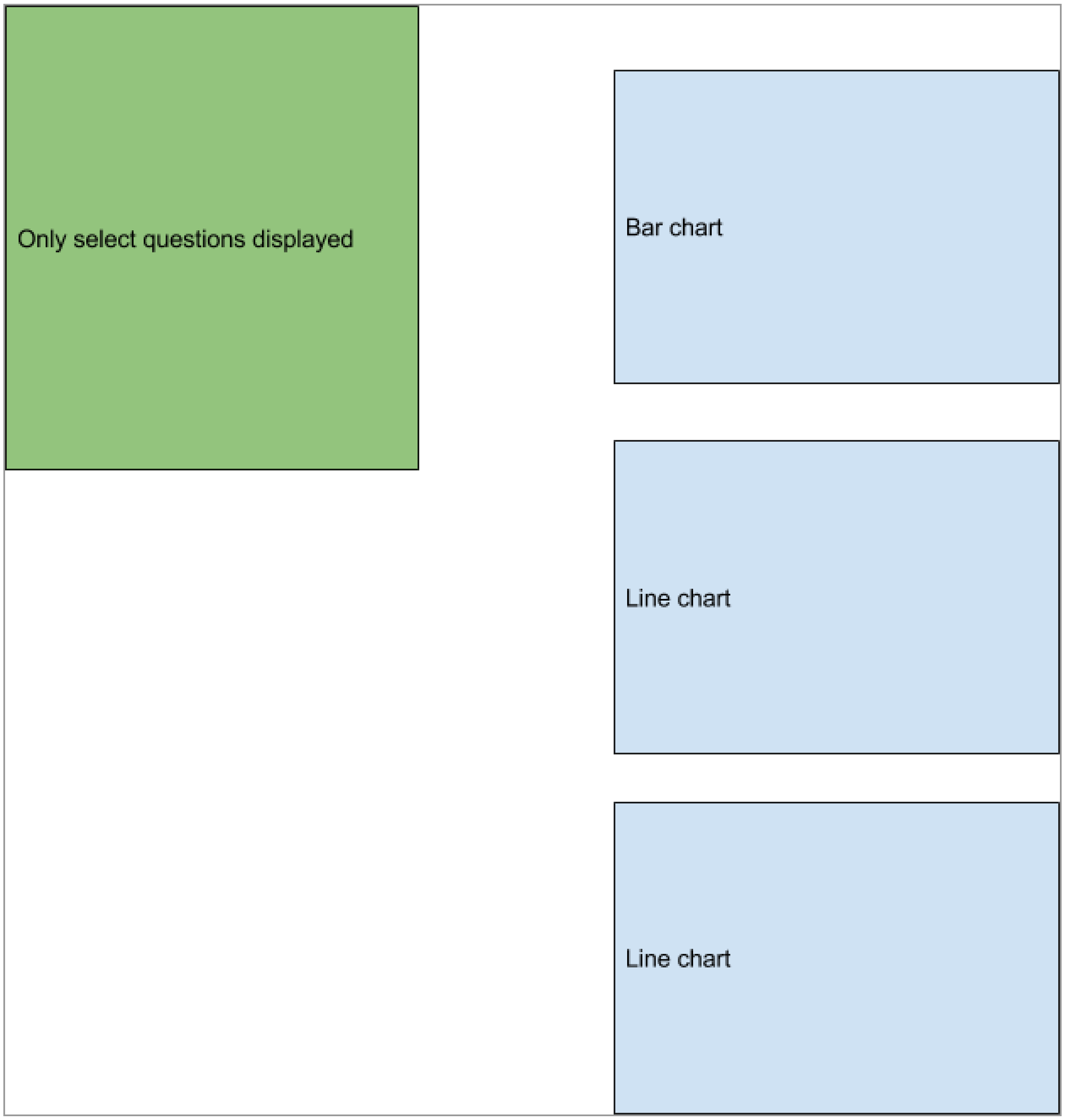
Implementation

As we prepared to implement our visualization we broke the code down into several files and classes. The first thing we implemented was the part that reads the data from the *survey_results_public.csv* file that we got from Stackoverflow. After reading in the data we had to do some major processing in order to have it easily consumable by the D3 code. We did that in the table class. We decided to do the processing lazily and only process a particular column if it was selected to be visualized. When the user selects the column to visualize, we map the responses by the name of response and assign a total salary and a count of salaries so that we could process the average later. This also allowed us to use the counts to figure out the percent of total responses by year.

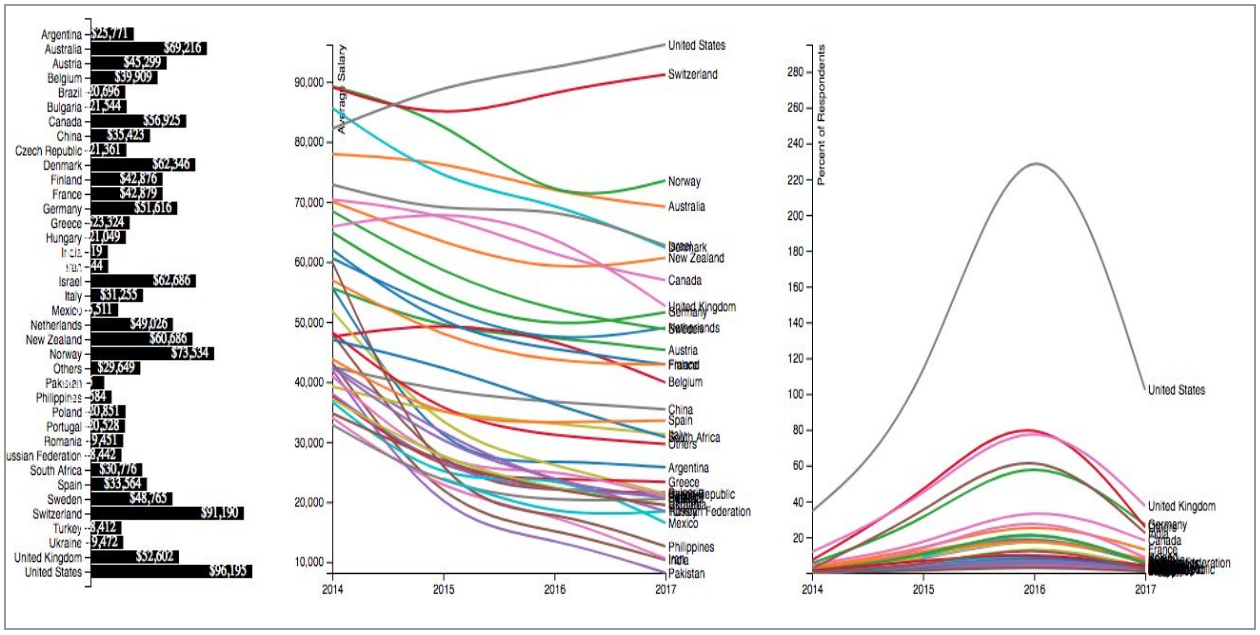
When we first implemented the visualization we showed every question in a large table on the left. This was not ideal because there were so many question that the user would have to scroll a lot in order to find the question they wanted to visualize. Furthermore, many of these questions might not be of key interest for the user.



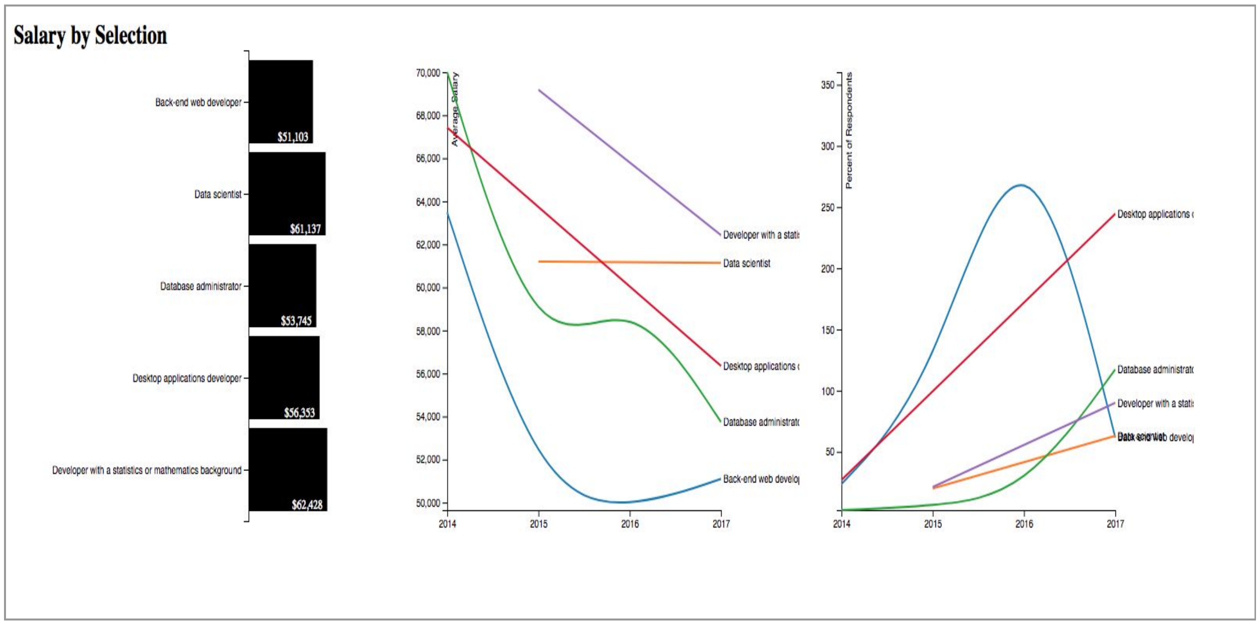
To fix this, we decided later on to select the most interesting questions that we thought the user would want to visualize instead of showing all of the 155 questions/fields that were in the 2017 originally. This made our UI better and provided a better experience for the user.



The next issue we faced was that when a user selected a question, sometimes there were over 50 responses to that question. This made our bar and line charts extremely cluttered and impossible to read.

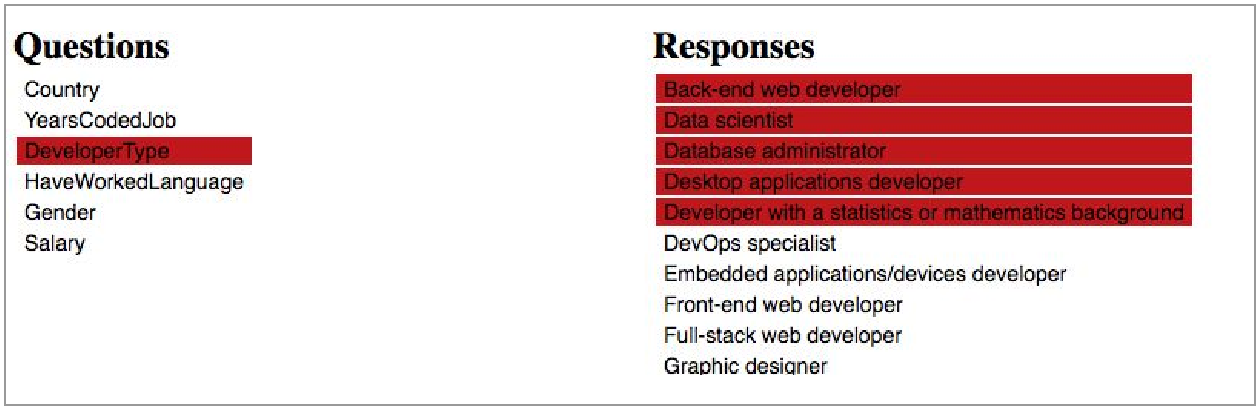


To fix this issue we decided to only display the top 5 responses when the user selected a question.



This fixed the clutter but introduced another issue. It was then possible that the user wouldn't be able to see some data they cared about! So we decided to put a second selection table that showed all of the responses with the original five selected by default.

The user could then click on responses to toggle their display on the charts or click on an already selected item to unselect it.



Questions

Country
YearsCodedJob
DeveloperType
HaveWorkedLanguage
Gender
Salary

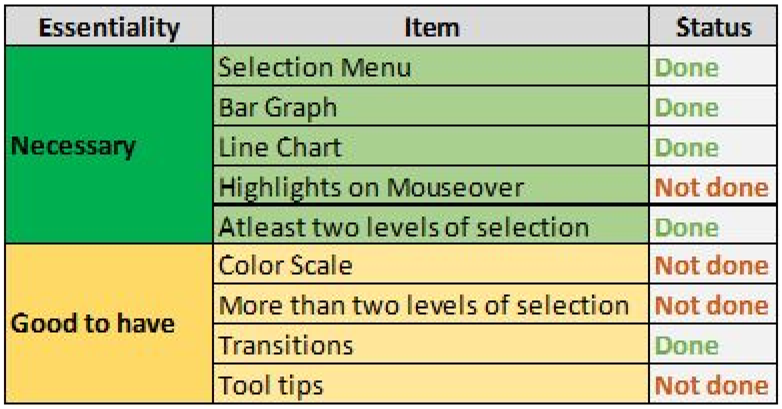
Responses

Back-end web developer
Data scientist
Database administrator
Desktop applications developer
Developer with a statistics or mathematics background
DevOps specialist
Embedded applications/devices developer
Front-end web developer
Full-stack web developer
Graphic designer

We thought this was a good solution because it gave more control to the user to see what they are interested in and to be able to perform their own analysis of the data, which is the primary intention of this visualization!

Evaluation and Insights

We definitely could have done much more with our visualization if we had more time but we think we did a decent job in this short time. If we compare with our original plans of some visualization features during proposal with final output, here is the status:

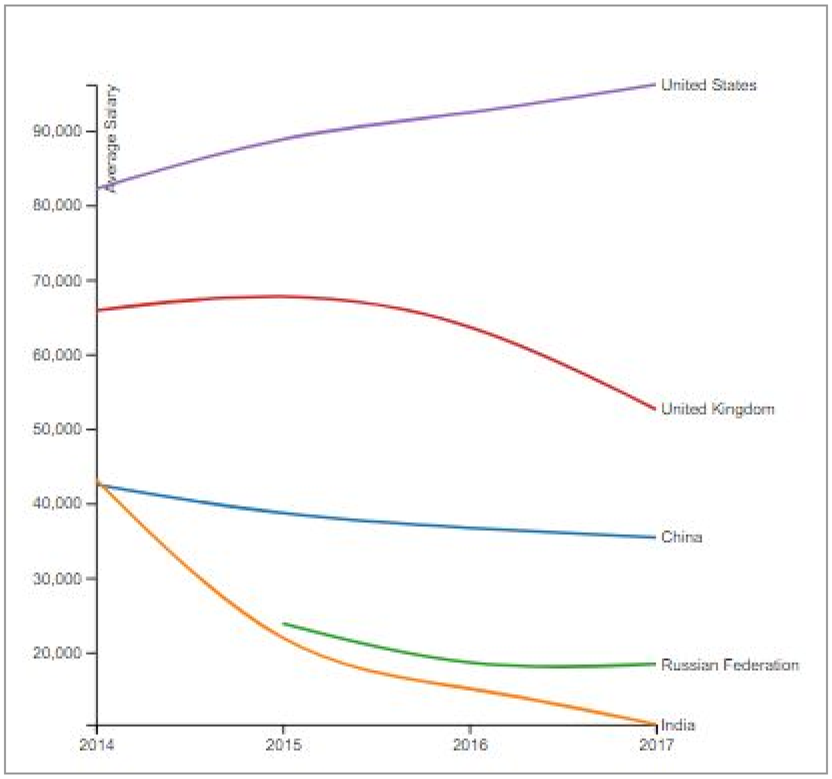


Essentiality	Item	Status
Necessary	Selection Menu	Done
	Bar Graph	Done
	Line Chart	Done
	Highlights on Mouseover	Not done
	Atleast two levels of selection	Done
Good to have	Color Scale	Not done
	More than two levels of selection	Not done
	Transitions	Done
	Tool tips	Not done

We consider our main success to be the correct capture of the data from the surveys into clean charts and enabling users to play with it for analysis. However, we would have really loved to enable selection between two distinct 'questions', but we could not do that mainly due to the issues with the available data not being consistent over the years.

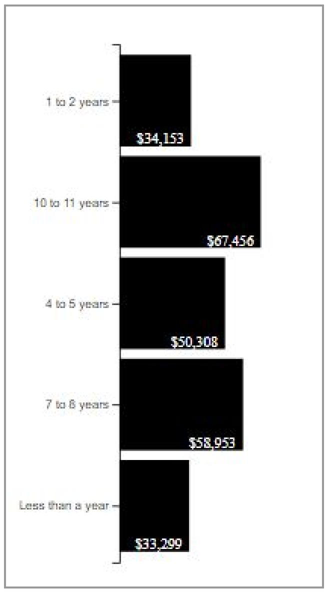
Some of the interesting insights are shared for the readers.

1. *United States is the best place to be if you want to work in Software Development*



Not only is the average salary in the US higher than some of the other countries shown in the chart, it is also on rise over the years. It was interesting to see the salary significantly going down in India, we can infer that a lot of it probably due to a significant increase in workforce (and freelancers), which drove the average cost of labor to go down.

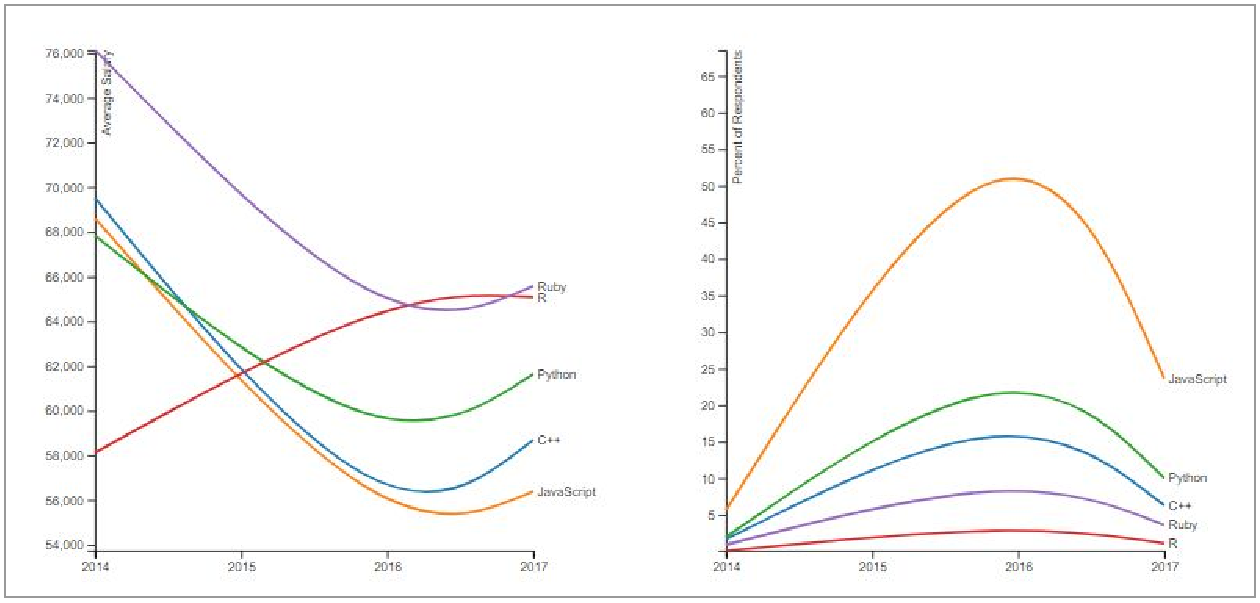
2. *With time, you make more money (duh!)*



This is probably a no brainer but we still decided to highlight this data as we think there is an explanation to the change in salaries.

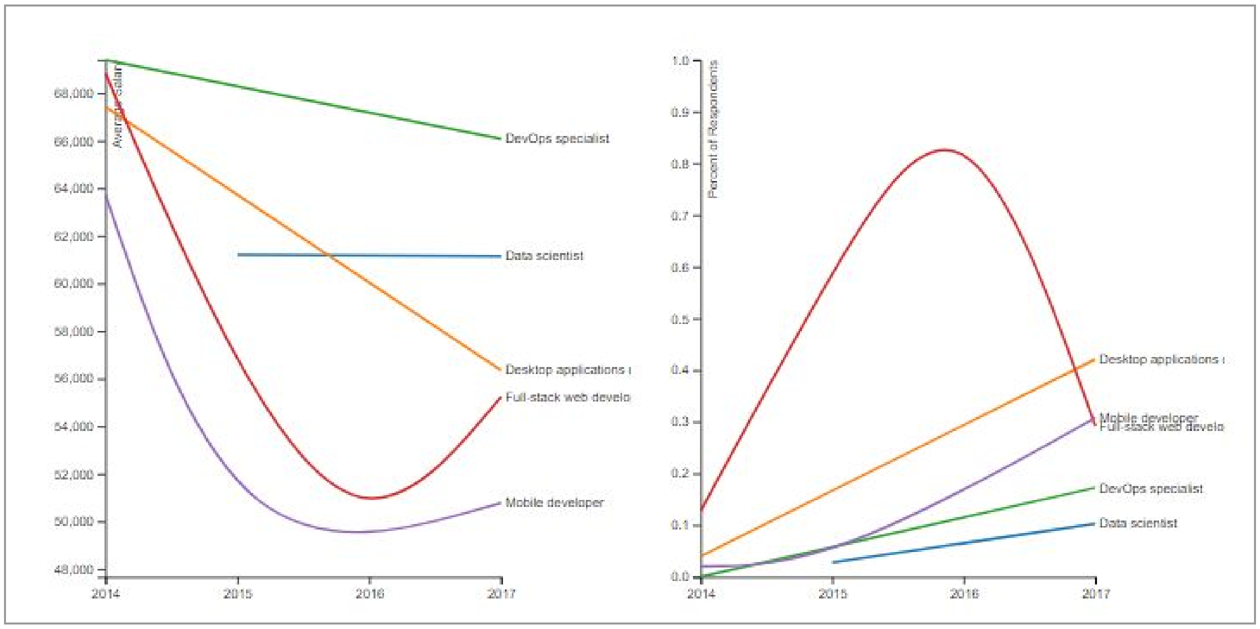
The entry level salaries appear quite low as the data includes responses from all countries. We see significant difference with 4-5 years experience (about 1.5X) probably because it mostly includes responses from countries with higher payscale, since we assume that emerging markets/workforce from India or China were not that matured 4-5 years ago.

3. *Javascript is the most popular language, but it doesn't make the most money!*



We see that languages like R and Ruby are very highly paid, however the ratio of the developers using these is quite low. This indicates these are very specialized skills. On the other hand, Javascript is the most popular language but it's popularity has decreased over the years and it makes the least money compared to other language skills.

4. Data Science pays well and is on the rise



Before 2015, the title Data Scientist was not even used by the survey respondents! Although low in proportions, the number of Data Scientists is growing every year.