STATISTICS SAVES THE DAY

AN ANLY482 PROJECT

IN COLLABORATION WITH SINGPATH.COM

TEAM 9

DARREN LIM FEI HONG

GOH GHEE GIN SHANE

LIM WEI YANG

PREPARED FOR PROFESSOR KAM TIN SEONG

Executive Summary

A team of analytics students have used simple box-plot diagrams to map the sequence of questions an educational programming website should use for the benefit of their users. The website is SingPath, a site that takes a self-directed learning approach to teach Java, JavaScript, and Python. Though the team initially began the project with the goal of developing a predictive analytical model, limitations of the website, database, and subsequently the data within the dataset rendered the model less than satisfactory, prompting the team to take an in-depth look into the website and database. The conclusion of their study indicates that the website should have its' questions reordered for the benefit of users, while the database needs to capture new information such as competencies and demographics to add dimensions for future analysis.

Table of Contents

Project Introduction

SingPath.com is an educational website that aims to impart the basics of Java, JavaScript, and Python to users of the site. It adopts an unstructured, self-directed learning approach (Smith, 2014) and allows students to pick and choose which questions to attempt and their leisure. By allowing students to control the pace of their education, the site aims to improve the efficiency of their learning in a fun, dynamic environment.

The initial proposal of the project was to build an analytical dashboard for the purposes of identifying the potential time taken by students to solve a specific question. As SingPath is often used by the client in real-life classroom scenarios, this would allow them to better estimate the time required by students to solve specific questions, allowing them to optimize their use of time within the classroom.

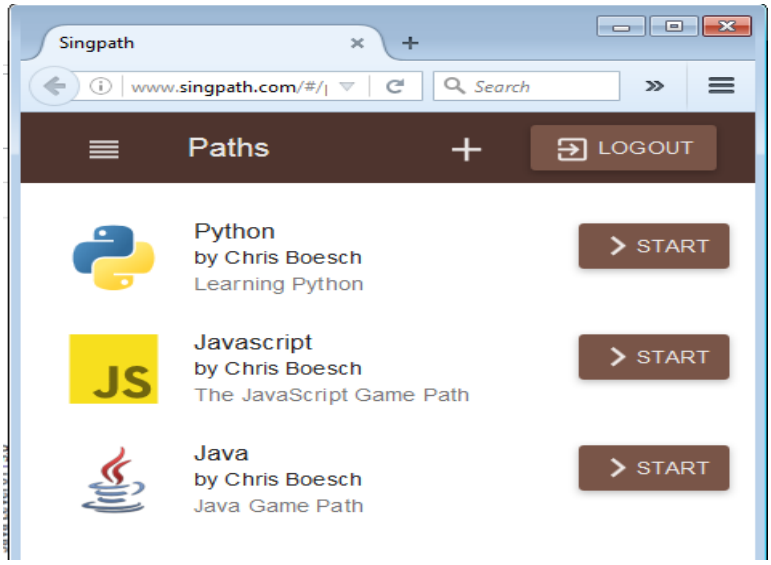


Figure 1: A view of the Languages of SingPath.com

The Dataset

Understanding Firebase

All data collected from Singpath.com is stored in Firebase.com in JSON format, which includes information about users, the questions they have attempted, the time they attempted each question, and the duration of each question attempt. Because data in Firebase is stored as JSON objects and thus in an unsuitable format for analysis, several steps were taken to obtain, convert, and clean the data in tabular format for analytical use.

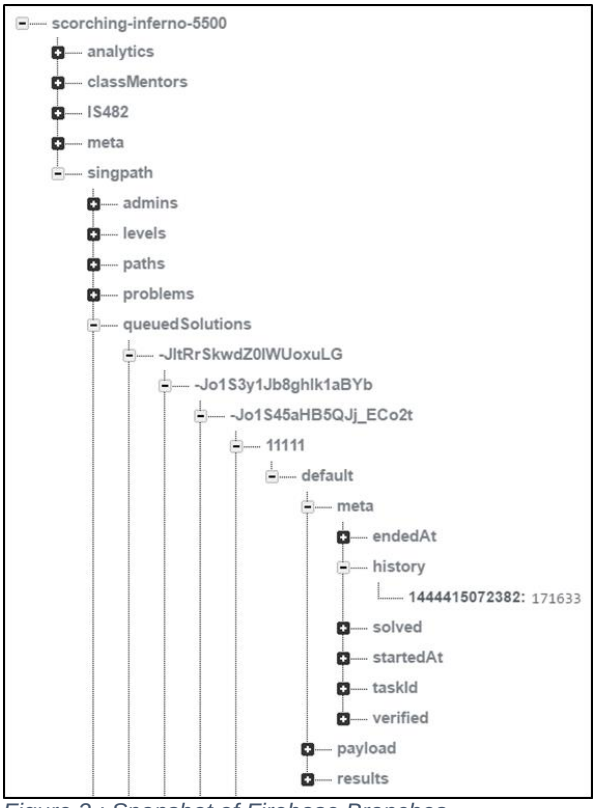


Figure 2 : Snapshot of Firebase Branches

All question attempts are stored in the "queuedSolutions" branch, and may be understood as:

- Question attempts
- Programming language's unique key
- Question level's unique key
- Question's unique key
- User's unique username
- Default
- Meta
- History
- Key-Value Pair: Start Time of First Attempt (key), Duration in milliseconds (value)

Subsequent branches only exist if additional attempts have been made, and a key-value pair is stored. The key is the start time of the subsequent attempt, and the value is the duration of the attempt in milliseconds.

Note that UNIX time is defined as the number of milliseconds that have elapsed since 1 January 1970, which is the standard used and understood by most programs. For example: 1444415072382 in UNIX time equates to 09 Oct 2015 6:24:32 PM and 382 milliseconds.

Extracting from JSON

We created a standalone HTML file[1] using the *AngularFire* library to manipulate the JSON data in Firebase as objects and arrays. We then used *ngCsv*, an AngularJS module, to save the objects and arrays into a single CSV file. Our pseudo-code used to accomplish this is as follows:

```
For each language,
    For each level of the language
        For each user attempt of the question
            Save the language level, question number, question title, username, start-time, end-time
            Also look up and save the full titles of each question, including the level and language
            Convert Unix time into a human-readable datetime format
            Convert Duration attempts from milliseconds to seconds (rounded up to nearest second)
```

Figure 3: Pseudo-Code for Data Conversion

[1] See Appendix: HTML Conversion Codes

Dataset Overview

Basic Dataset Information

Table 1: Basic Dataset Information

Language	Levels	Total Attempts
Java	2	8
JavaScript	2	258
Python	10	22,608

Each row within the dataset indicates a unique attempt by a user on a question. As such we understand that our dataset contains 22,874 (8 + 258 + 22,608) unique question attempts.

Split by question levels, majority of question attempts are on questions in Python Level 01. The graph shows an expected decreasing trend of number of attempts for Python: The higher the level, the less the number of attempts. However, Python Level 05 breaks the trend and has less question attempts than Python Level 06.

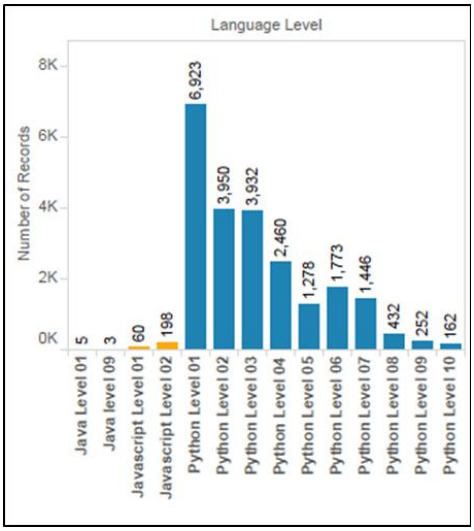


Figure 4: Attempts by Language Levels

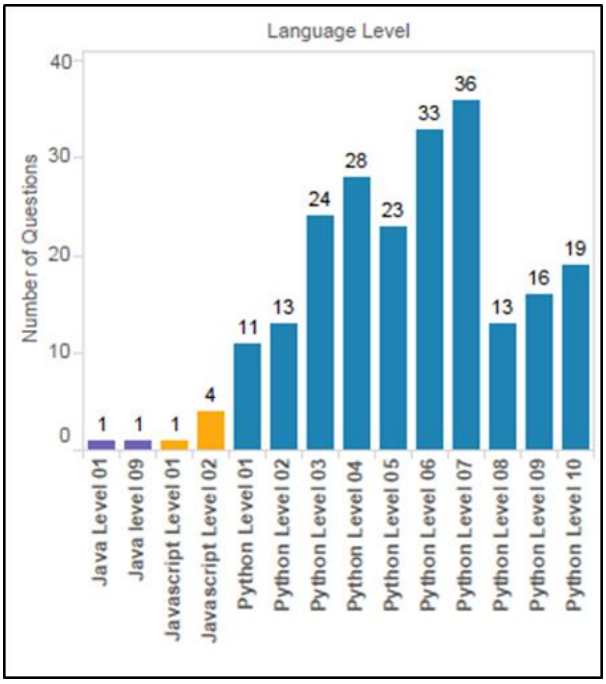


Figure 6: Number of Questions per Language Level

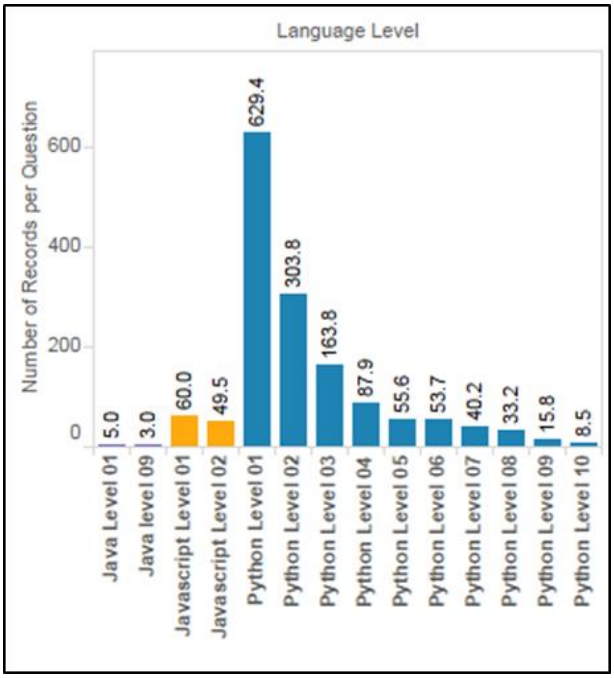


Figure 5: Average Attempts per Question per Language Level

However, we can factor in the number of questions of each level to calculate the number of question attempts per question by level (*see Figure 6*). Taking the average number of attempts per question within each level, we can then observe an inverse relationship between the language level and the average number of attempts per question per level (*see Figure 5*).

If we look at the number of question attempts by the questions themselves (*see Figure 7*), we can see that, with some variations, the number of attempts generally decreases as the question number increases. From the perspective of time, we see that most attempts were done on Saturday, while the least number of attempts were made on Sunday (*see Figure 8*). Finally, most attempts were made in the afternoon at 4pm or 1600hrs, while the least number of attempts were made between 5am to 9am or 0500hrs to 0900hrs (*see Figure 9*).

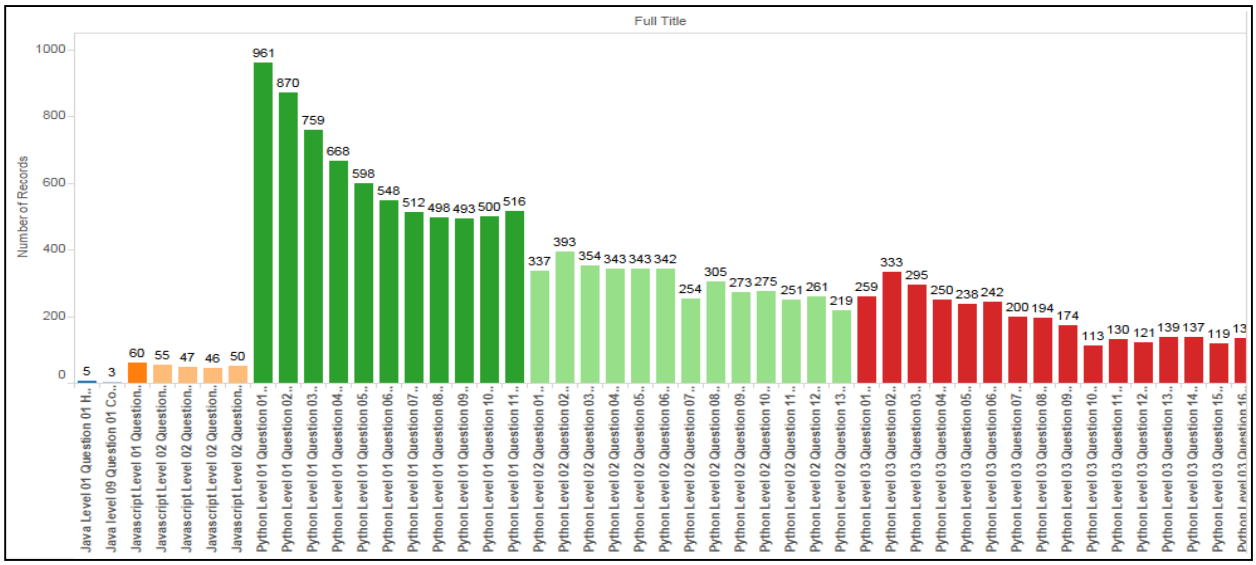


Figure 7: Question Attempts by Questions

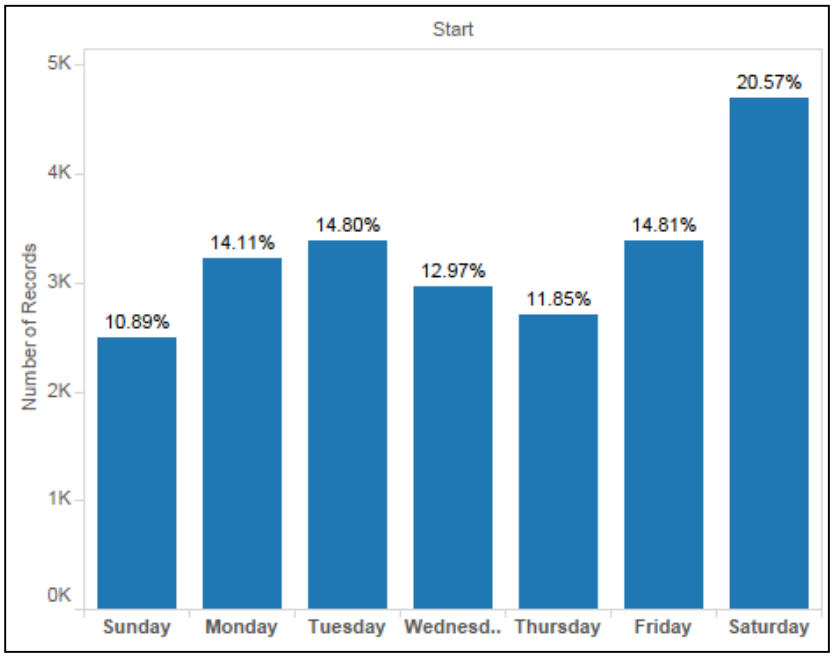


Figure 8: Attempts sorted by Days of the Week

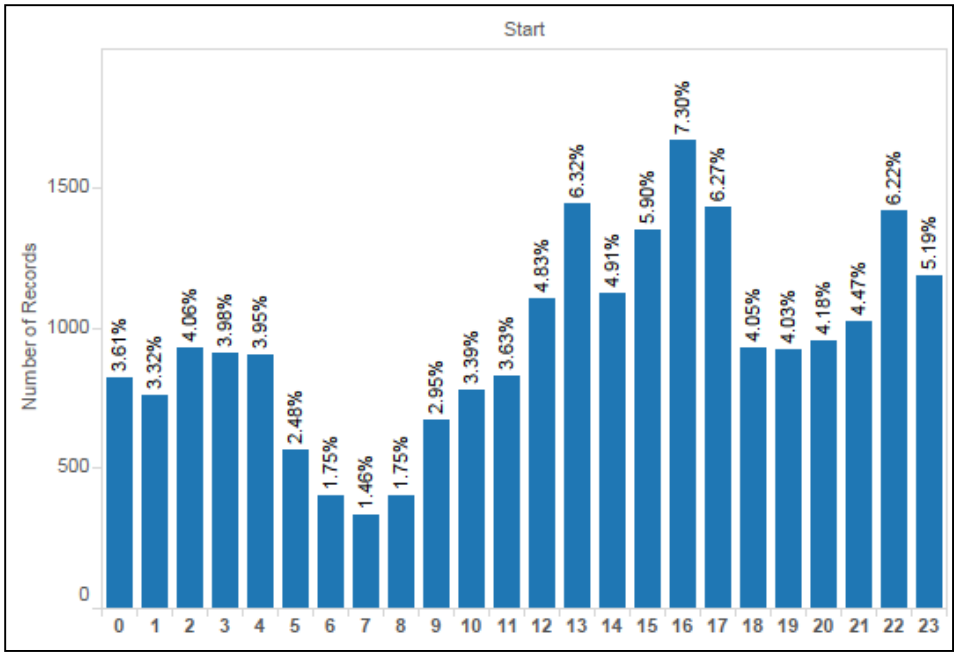


Figure 9: Attempts Sorted by Hours of the Day

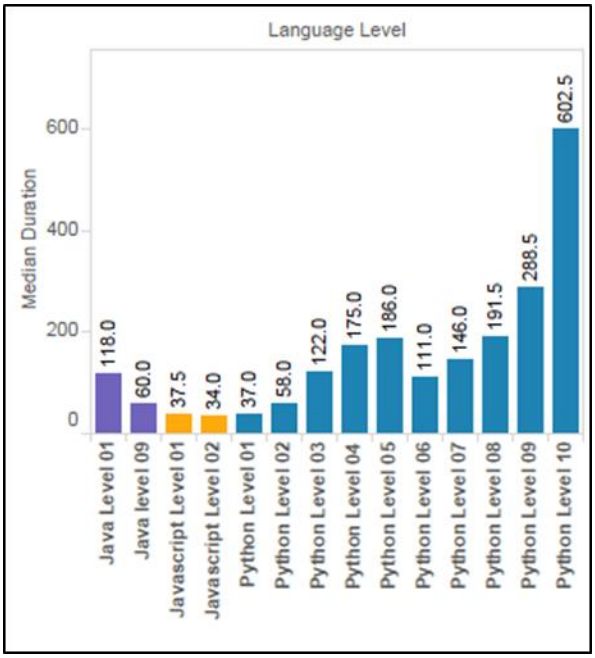


Figure 10: Median Durations by Language Level

Our group expected to observe the following trend: The higher the question level, the longer the durations of question attempts. This is because questions in a higher level ought to be more difficult and thus take a longer time to resolve. When we created a graph of the median duration on a Language Level (*see Figure 10*), we noticed that the languages of Java and JavaScript did not follow this pattern. Python questions followed our predicted trend for levels 1 to 5, however there was a sudden drop in level 6 which quickly rose again through to level 10. Upon closer inspection of attempt durations by individual questions, we realized that the distributions of durations were extremely irregular, which made any further meaningful analysis of the data impractical.

Phase 1: Building a Predictive Model

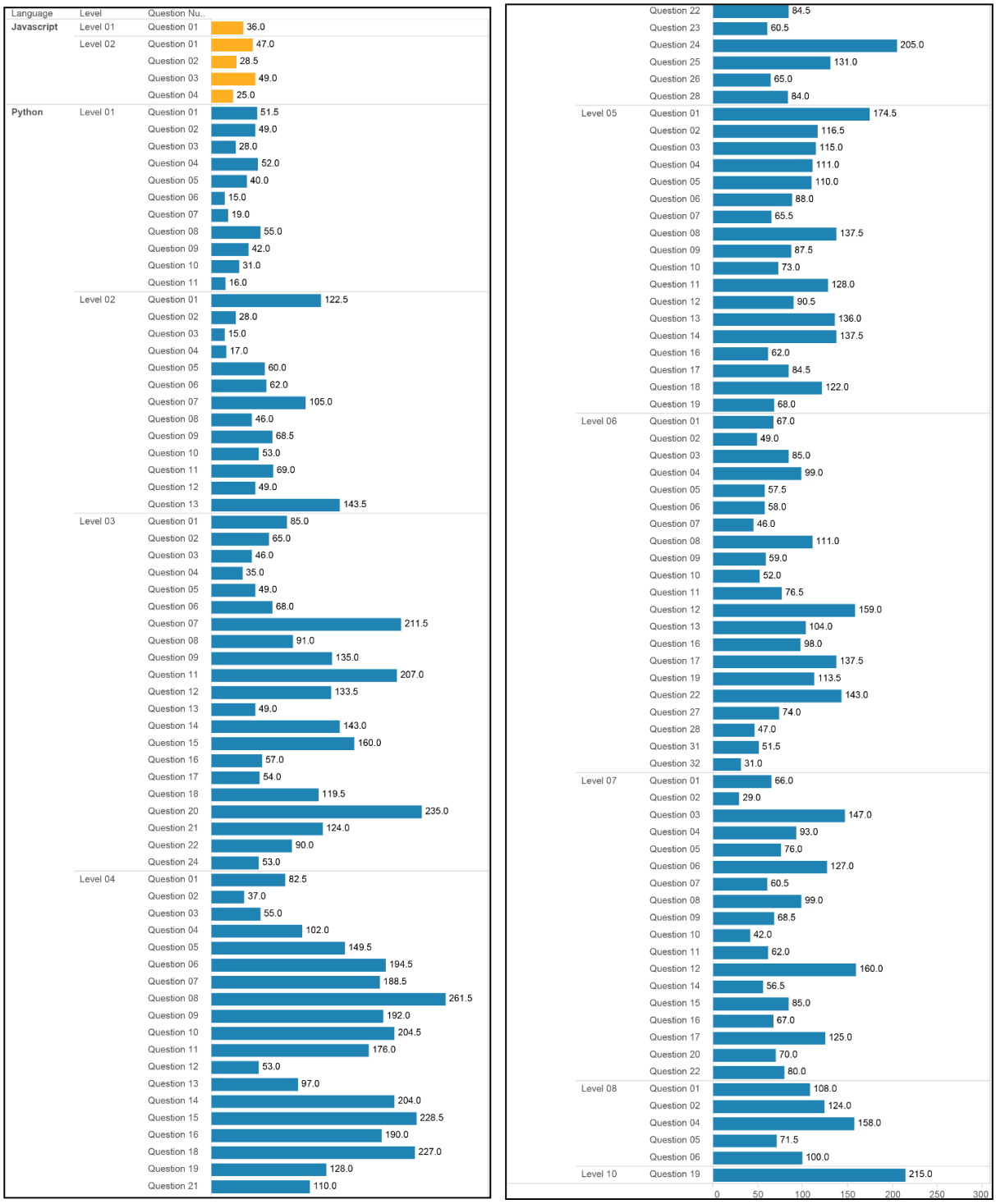


Figure 11: Median Duration for All Questions

Our team had hoped to develop a model to estimate the average duration an attempt would take for any question within the repository. In order to clean the data for analysis, we ignored all questions that had fewer than 30 attempts, as well as all attempts that took longer than 10 minutes or 600 seconds.

After cleaning the data and discovering the median duration for all questions, our team developed two predictive models for the duration of attempts. In the first model, we predict that all new users who attempt the question will probably achieve the same timing as the existing median duration. For our second model, we estimate the likelihood of a user meeting a specific timing based on their percentile ranking for other questions. A simplified model for the "Percentile-based" predictive model is as follows:

1. User A has already attempted 3 questions thus far: Questions U, V and W
2. For Question U, his duration of 42 seconds ranked in the 15th percentile
3. For Question V, his duration of 48 seconds ranked in the 20th percentile
4. For Question W, his duration of 57 seconds ranked in the 25th percentile
5. Therefore, his median percentile ranking is at the 20th percentile
6. For any given new 4th question that User A is about to solve, his attempt's duration is likely going to be placed at the 20th percentile amongst all the existing attempts' durations of that 4th question
7. Suppose User A is about to attempt Question X
8. The existing attempt durations of Question X are: 30s, 38s, 42s, 50s, 56s and 63s

The new model would then predict that User A would take 38 seconds to solve Question X, because 38 seconds is ranked at the 20th percentile amongst all existing attempt durations of Question X

Comparing the Models

To compare the accuracy of the models, we tested the data twice with two different models. For each test, we randomly split the dataset into two groups: 80% of the set for training, and 20% for testing. For each model, we tested the validity by measuring the "Sum of Squared Errors" (SSE), where we take the summation of the squared value of the difference between the actual and predicted results for all attempts. The lower the score, the more accurate our model would be.

For the median model, we performed the following steps:

1) For each question in the training dataset, calculate the median duration of all attempts for that question
2) For each attempt in the testing dataset, we use the calculated median duration of the corresponding question from the training dataset as the predicted duration of the attempt.
3) The resulting SSE value between the predicted and actual duration values is 49,127,814

For the percentile-based model, we performed the following steps:

1) Calculate the percentile ranks of durations of all attempts for each question in the training dataset
2) Calculate the median percentile rank of each user based on all his percentile ranks for all questions in the dataset
3) For each attempt in the testing dataset, we use the calculated median percentile ranking of the user to calculate the predicted duration of the attempt. This calculation is based on the existing duration distribution of that question in the dataset
4) The resulting SSE value between the predicted and actual duration values is 41,340,316

We also counted the number of times the predicted duration of the percentile model was closer to the actual duration than the predicted duration of the median model was, and found that the prediction of the percentile model was more "accurate" than the median model 2049 times (55.8%) while the median model was more accurate than the percentile model 1621 times (44.2%)

In conclusion, the calculations show that the new model is better than the default "model", but only marginally. We feel that this should be due to the limitations of the data discussed in the next section, and that if these limitations were rectified, the model should be able to give very personalized and accurate predictions.

Limitations of Predictive Model

a) All Duration Less Than 10 Minutes

Due to the heavily skewed and "dirty" data captured, our models had to be built using attempts that were less than 10 minutes in duration. This therefore rendered our model useless in the face of the actual dataset, where durations could range anywhere between 3 seconds to an entire day, or 24 hours.

b) Database Rows Overwritten

From initial observations of the dataset, we understood that the "Start Time" was the time at which students visited the question URL to attempt the question, and "End Time" is the time at which the students submitted their answers. However, given that some students had completed relatively complex questions in only 3 seconds, our team came to the conclusion that the database values of "Start Time" and "End Time" were being overwritten each time a student entered a page and submitted a new answer. Furthermore, the database had no method to track the histories of the students.

We shared this information with our client mid-way through the project, and so they modified the database to now track the duration of each attempt by every student on a single question. Though less than 0.1% of records now contain history, this is a promising beginning to tracking the progress of students through the site.

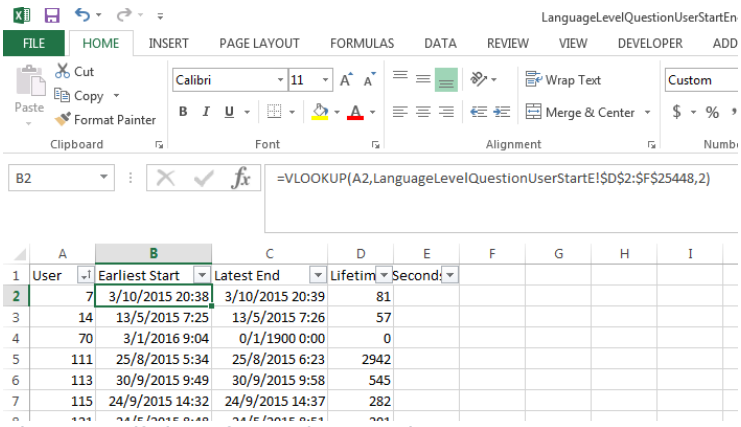
c) Dirty Data within Database

Records indicate that students may be as quick as 3 seconds to solve a complex question, whilst they may take as long as an entire day to resolve a relatively simple question. This "dirty" distribution of durations that is seen throughout the database indicates that there is something amiss with the website or the database.

Additional Analysis Attempts

Lifetime Analysis

In order to assess the total time spent by an individual on the site, the team needed to know the first recorded start time and the last recorded end time. To accomplish this, the team used Microsoft Excel and the sorting and VLOOKUP functions. Firstly the team extracted the unique user IDs and placed them in a separate table, sorted in alphabetical order. Secondly the database was sorted first in alphabetical order of user IDs, followed by sorting the start times from the largest to the smallest value. Thirdly, the list of unique user IDs has an additional row added called "Earliest Start", which uses the formula: "=VLOOKUP(A2,LanguageLevelQuestionUserStartE!$D$2:$F$25448,2)", where *A2* is the name of the user, *LanguageLevelQuestionUserStartE* is the name of the main data cube, and *2* is the column to retrieve data from in the main data cube.



	A	B	C	D	E	F	G	H	I
1	User	Earliest Start	Latest End	Lifetim	Second				
2	7	3/10/2015 20:38	3/10/2015 20:39	81					
3	14	13/5/2015 7:25	13/5/2015 7:26	57					
4	70	3/1/2016 9:04	0/1/1900 0:00	0					
5	111	25/8/2015 5:34	25/8/2015 6:23	2942					
6	113	30/9/2015 9:49	30/9/2015 9:58	545					
7	115	24/9/2015 14:32	24/9/2015 14:37	282					

Figure 12: Lifetime of Users in Seconds

This allows the earliest known start time for each user to be extracted, and once this formula is applied across the row, the formulae in the cells are then replaced by the values extracted, which results in the proper datetime information being displayed in the cell.

A similar process is applied to discover the latest end time, but the main data cube is now sorted first in alphabetical order by users, then sorted by end times from smallest to largest. The formula applied to the "Latest End" column is

"=VLOOKUP(A2,LanguageLevelQuestionUserStartE!$D$2:$F$25448,3)", which extracts the latest end time recorded in the data cube. A simple difference between the end and start times yields our objective column: "Lifetime_Seconds", which is the total seconds elapsed from the earliest start time to the latest end time of each user, which we can then define as the lifetime of a user. We discovered that over 150 students spent nearly 24 hours on the site before finally leaving, while 184 students did not even spend a minute coding on the web site before leaving. We then decided to supplement our findings by attempting to discover the drop-off points of the users.

Drop-Off Analysis

The data was grouped and sorted to indicate the questions with the highest dropout rates, defined as the last question a given user has attempted. The first three questions of Level 1 in Python had the highest rate of discontinuity, as these users were probably uninterested by the third question of the coding program and never returned to the site again.

	A	B
1	Last Attempted Question	Number of Users
2	Python Level 01 Question 01 Welcome	234
3	Python Level 01 Question 02 Your First Program	208
4	Python Level 01 Question 03 Starter Code	89
5	Python Level 01 Question 11 Wizard	85
6	Python Level 02 Question 01 Data Types	81
7	Python Level 01 Question 04 Expected Results	66
8	Python Level 03 Question 03 Product	50
9	Python Level 03 Question 01 Functions	46
10	Python Level 02 Question 07 Python Math - Again?	34

Figure 13: Count of Last Attempted Question

The next two questions with the highest drop-off rate are the last question of Level 1 Python, followed by the first question of Level 2 Python. From this we could advise our client to find more interesting ways to incentivize users to continue doing the questions at the end of a level and at the beginning of a new level.

One point of interest is that Question 3 of Level 3 Python has a higher discontinuity rate than that of Question 1 of Level 3 Python. This may be indicative that users are discouraged from attempting the first question, though the reason is not apparently clear, and given that this pattern repeats for some other levels in Python, further analysis is warranted.

Conclusions from Phase 1

Due to the heavily skewed and "dirty" data from the dataset, our team came to the realization that this may be an indication that there is either something amiss with the website, or in the way that data is collected within the database. This prompted us to take a closer look at the SingPath website, as well as re-examine how we were approaching our analysis.

Phase 2: Revised Approach

Examining the Source: SingPath

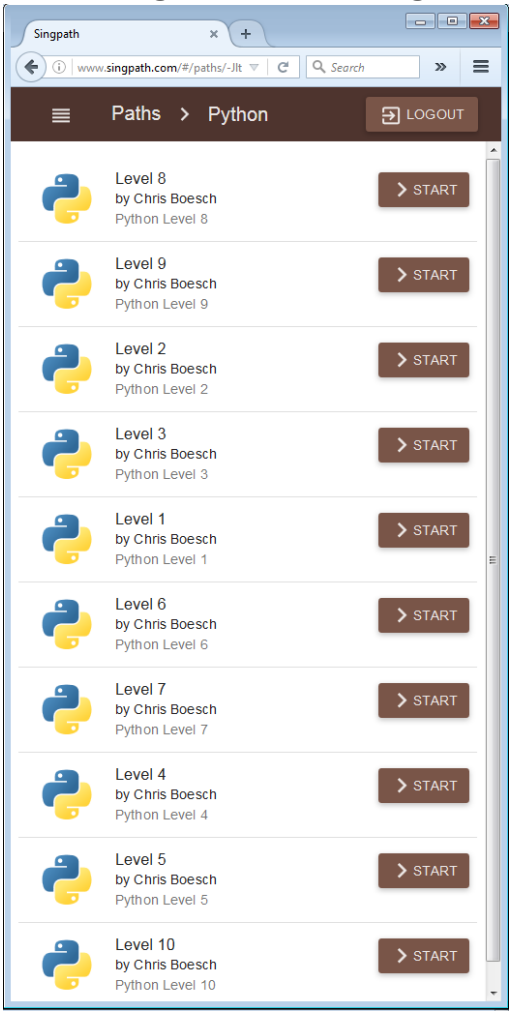


Figure 14: Python Levels Page

When we approached SingPath.com, we were first introduced to its' learning methods at the level selection page, after we had decided to pursue the Python language. From this page (*see Figure 14*), we can see that there are no guidelines provided for the student, no clear set of instructions on how to move forward. Students are given full autonomy to decide which levels they wish to attempt, and there are no safeguards to prevent students from diving straight away into the more advanced levels for the more difficult questions.

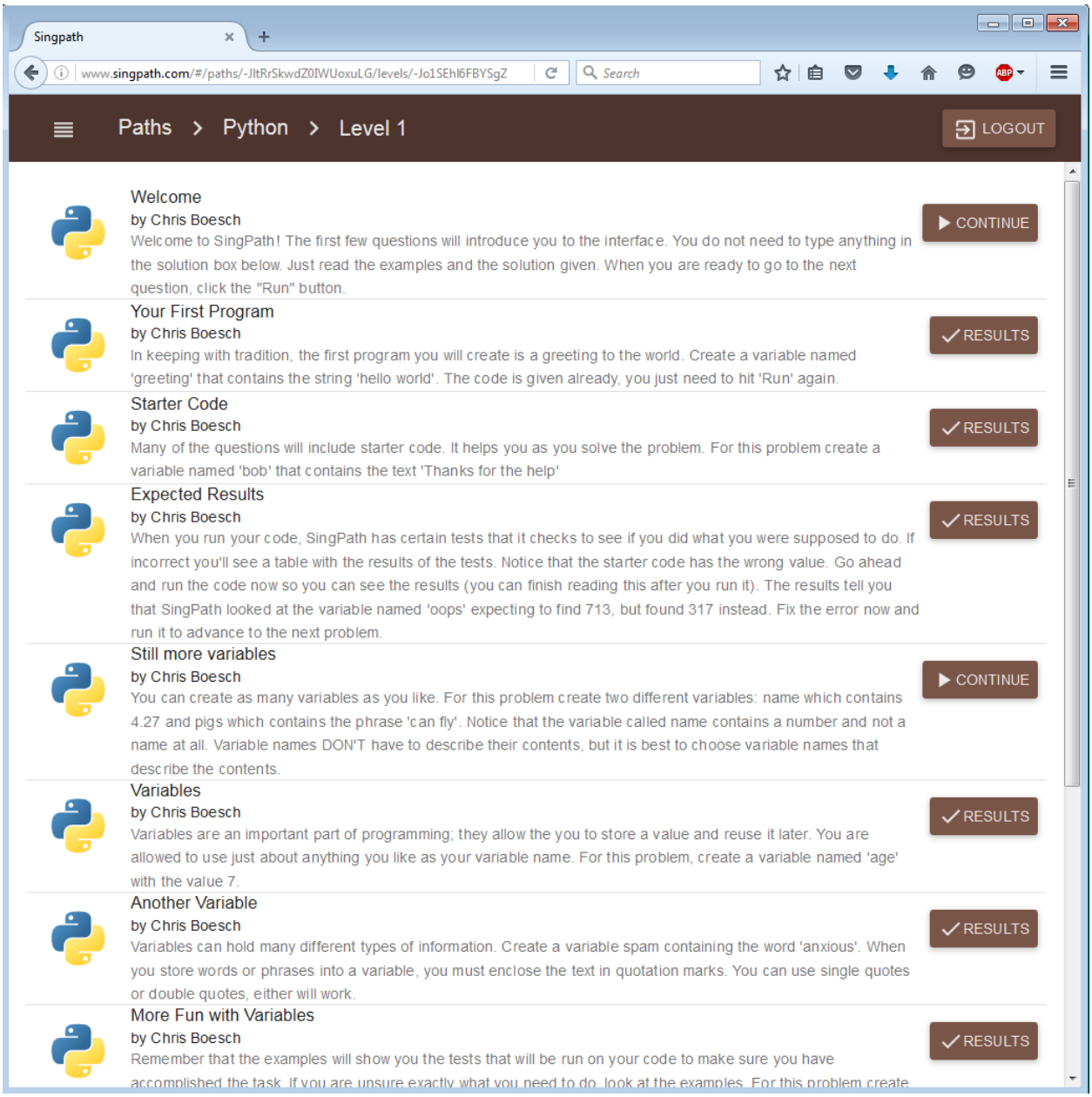


Figure 15: Python Level 1 Questions Page

This is made even more obvious in the questions page (*see Figure 15*), where the questions are arranged in a seemingly random order, though the first question displayed is “Welcome”, which is clearly meant to be tackled first.

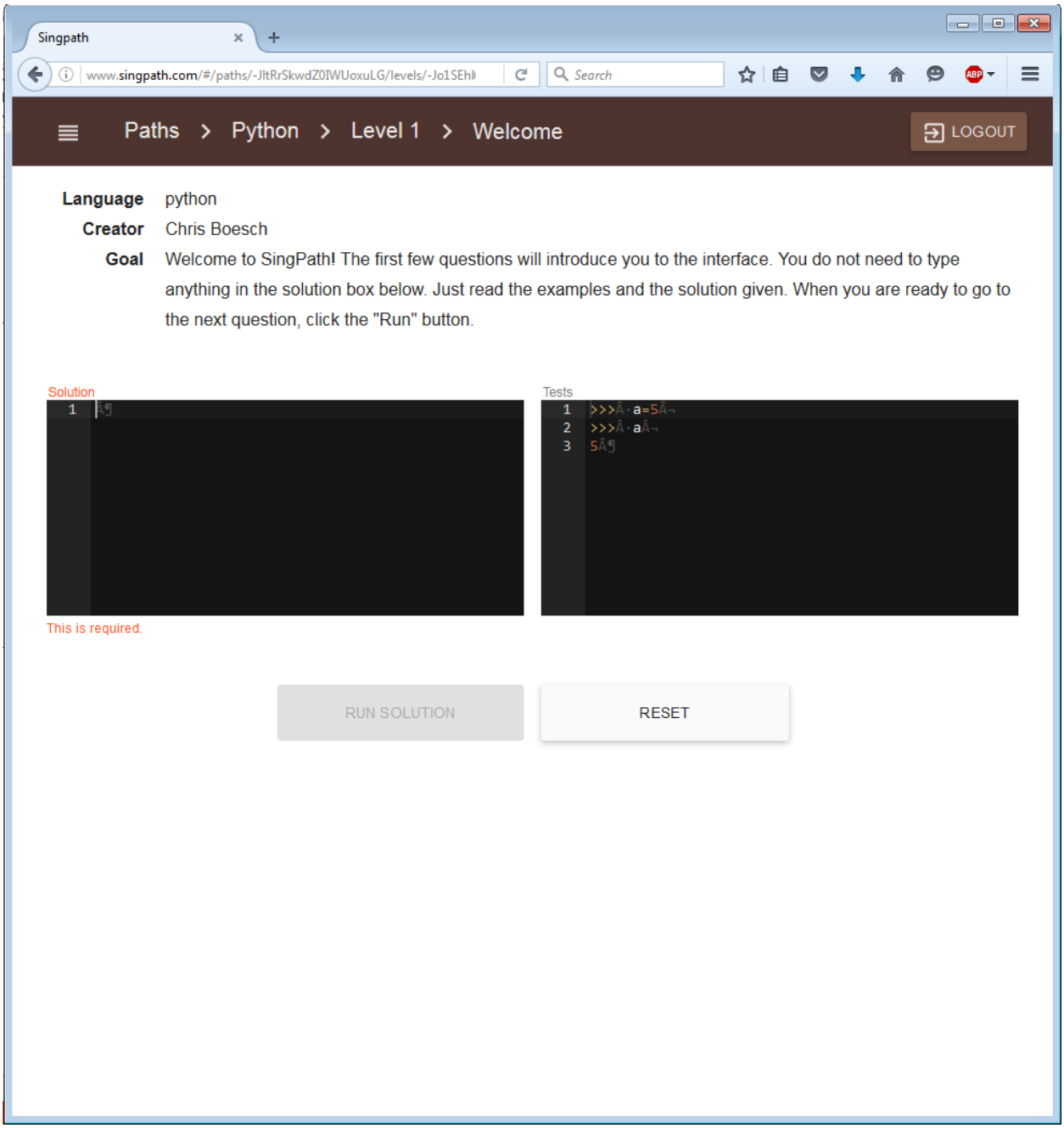


Figure 16: The first question of Python begins to show errors

When attempting question 1, we are informed that we need to simply click on the “Run Solution” button to proceed, however the button is inactive until we enter some code to proceed. Already the website is beginning to display errors which explains why some students cleared the questions in 3 seconds, whilst others would have taken a longer time.

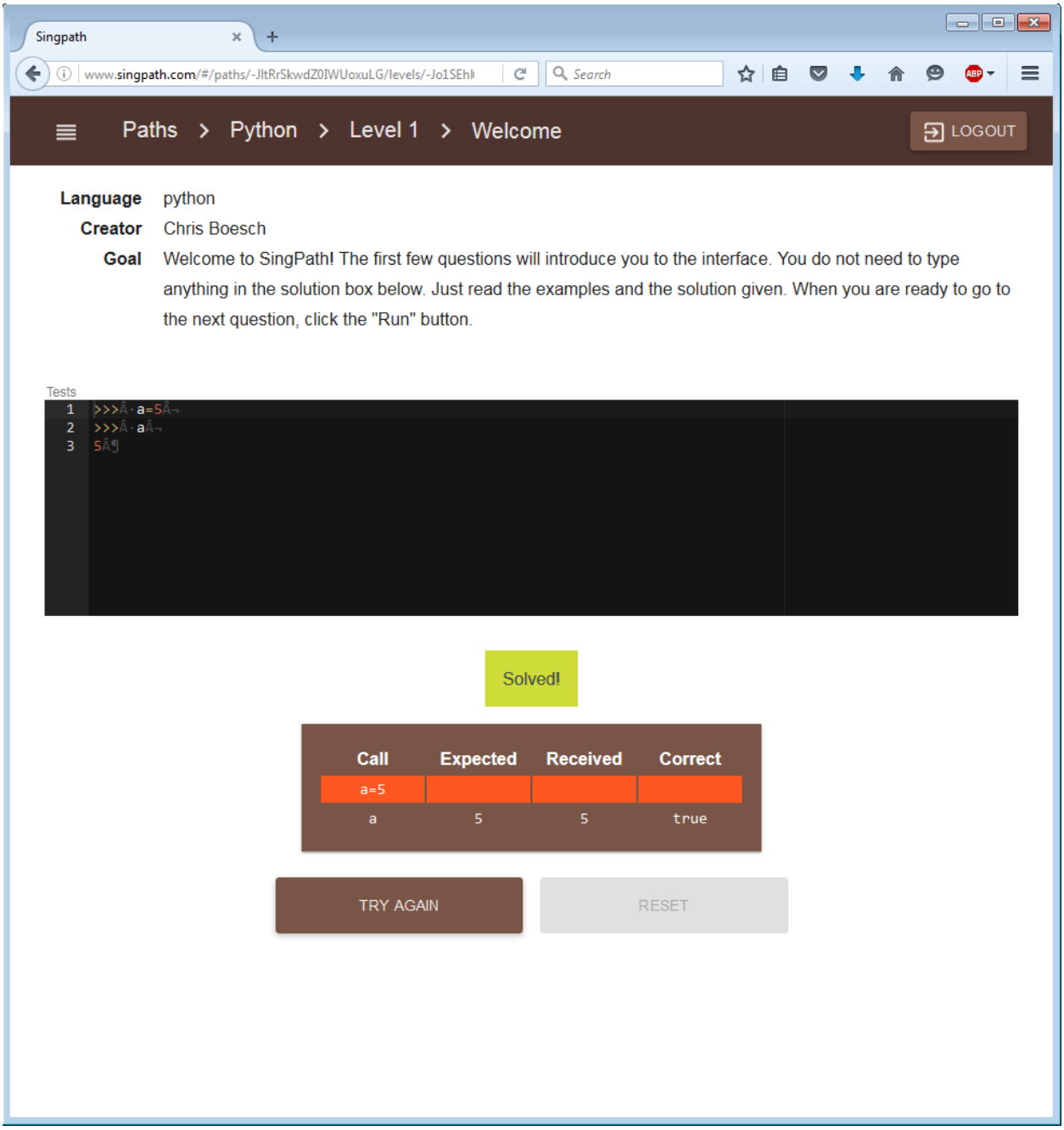


Figure 17: Successfully Completing Question 1

Upon successfully completing the question, there is no further prompting to proceed to the next question. Instead, we are presented with the option to try again, with no prompt to return back to the question selection screen to proceed with other questions. This evidence serves to solidify our hypothesis that the duration displayed in the database is heavily influenced by the web design of SingPath.

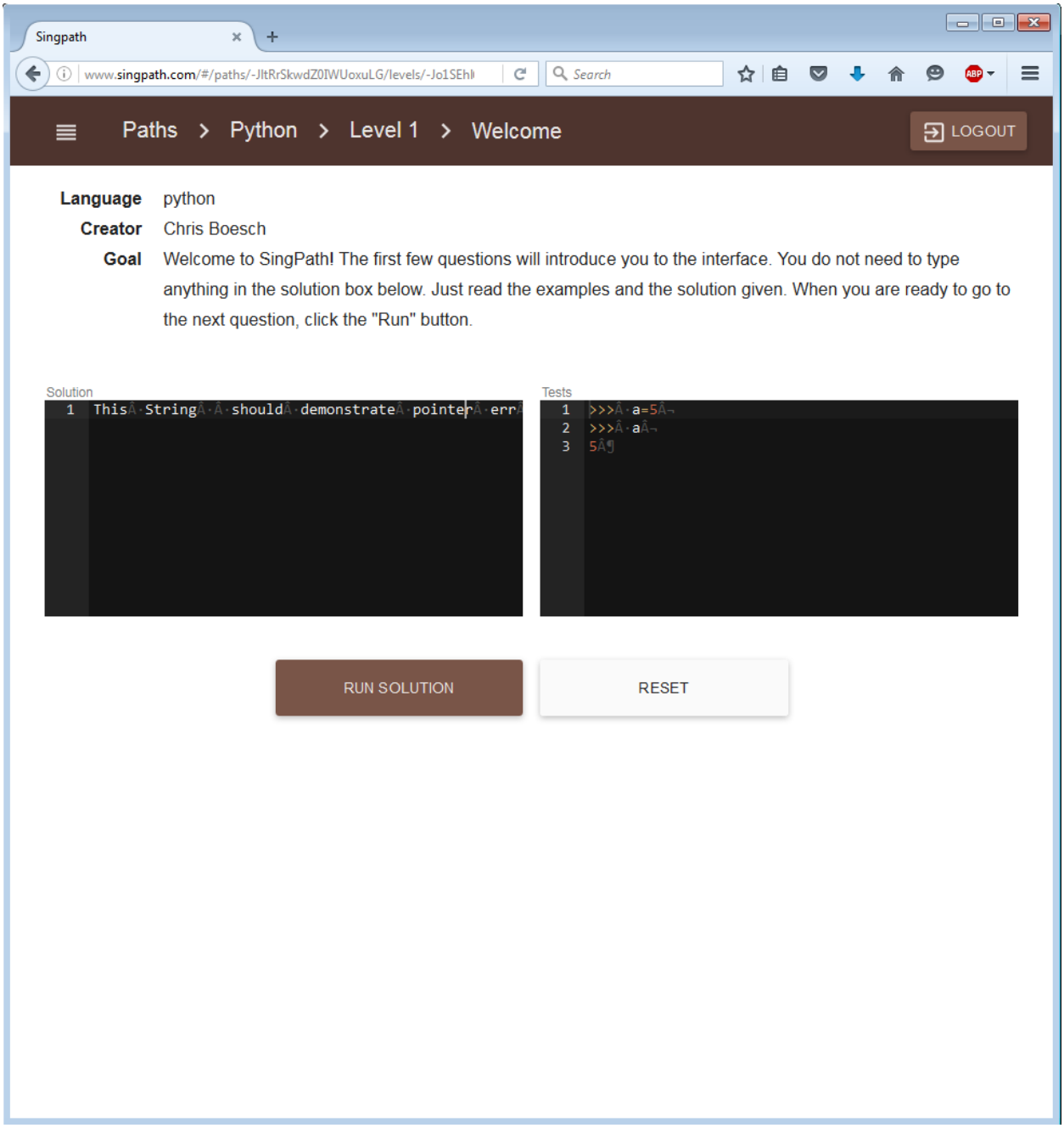


Figure 18: Issues with Pointer Position

While experimenting with the text boxes in SingPath, our team discovered that there are errors in the position of the pointer, as well as a curious display bug where spaces are presented as some character instead. In *Figure 18* above, we see that our cursor is in front of the 'r' character in the string "pointer", however our actual cursor is behind the string "err". These errors would be detrimental to more complex problems, as we would later discover, and further proves how the user experience on the site impacted the duration timings in the database, explaining why the spread of the data was so "dirty".

Learning Methodology

The learning pedagogy applied here appears to be the non-linear, Self-directed learning (Knowles, 1975) structure, which is a form of education meant to provide the student an opportunity to learn at their own pace, where the typical role of a teacher now becomes that of an advisor. Applying such a concept to the SingPath site has created the format it now utilizes, where questions are listed at random and students may choose the questions they wish to attempt at their own pace. While the merits of this pedagogy enable students to proceed at a pace of their own, students are potentially left floundering and wondering which questions they may attempt at all. Another risk this pedagogy carries is that students may attempt a question that is far too difficult, and give up on attempting subsequent questions as a result.

Discovering Limitations

Lack of an Enforced Environment

The reason for the vast number of outliers of attempt durations is because there is no enforced environment for users to complete their questions – users may attempt a single question over the course of a few hours, leaving midway to perform other tasks. This compromises the credibility of the duration each user took to answer each question.

Lack of Question Order

When users visit the website, they are presented with a list of levels and subsequently a list of questions, which they may then attempt in any order. The questions are numbered, ordered, and titled in a way that implies that users should proceed through the questions in some fixed pattern, however there is no enforcement of this. As such, users do not need to complete questions in sequence to “unlock” the next set of questions, which may result in users jumping from question to question in a random order. The result of this means it will be difficult to ascertain the progress of each individual student, as they may not be learning concepts in sequence and may therefore complete questions faster or slower than anticipated.

Lack of Database Dimensions

A lot more interesting analysis could have been done if the data included additional information. Such as the list of competencies tested for each question e.g. for-loops or if-else statements. This way, we could potentially see what the competencies are that users in general or even specific individuals tend to be weak at. Additional information of users such as their age, gender and education level would also allow us to potentially identify how different clusters of users fare based on their demographics. These additional information in the dataset would also help create a more accurate personalized prediction model.

Revised Cleaning

Our team therefore began working on cleaning up the data anew. This time, we looked at the distribution of duration for each question, building a box-plot diagram of each question.

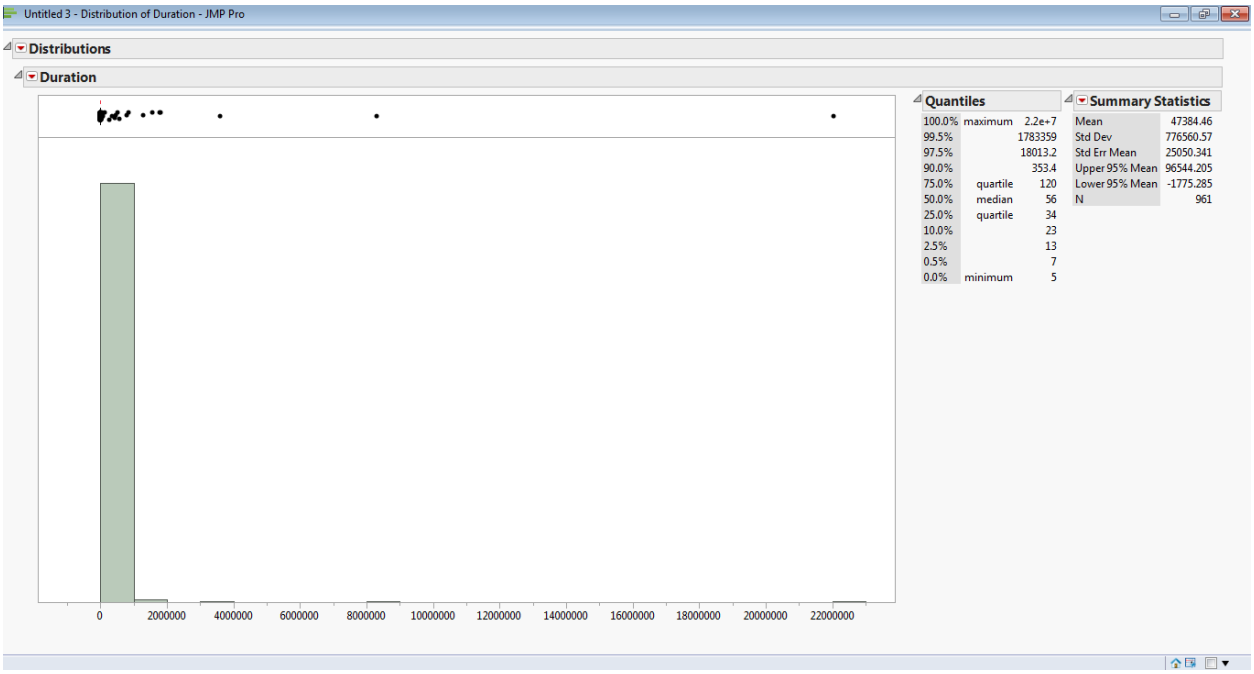


Figure 19: Unfiltered distribution of duration for Python Level 1 Question 1

We would begin with an overall view of the distribution of duration by questions, and for each question we would begin to filter out the data in an attempt to remove outliers. Usually this would result in our filtering out durations that exceed a specific amount.

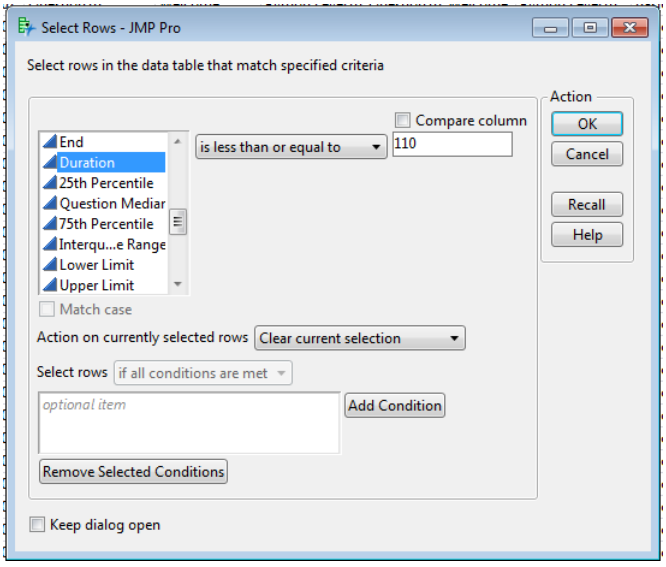


Figure 20: Filtering parameters

In this case, for Python Level 1 Question 1, our team removed all durations that exceeded 110 seconds. As such, the resulting box-plot diagram looked much cleaner.

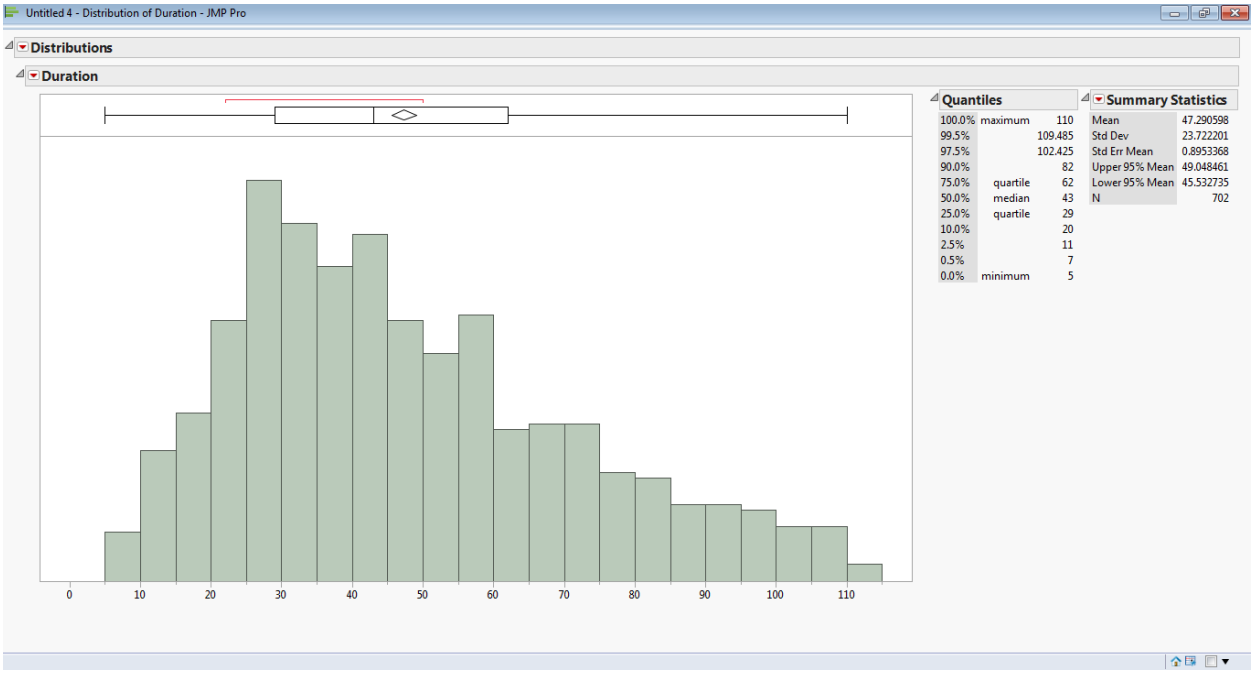


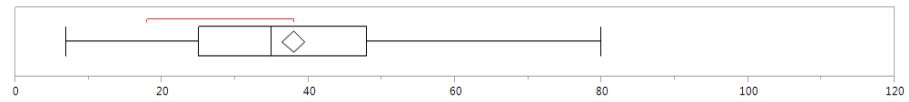
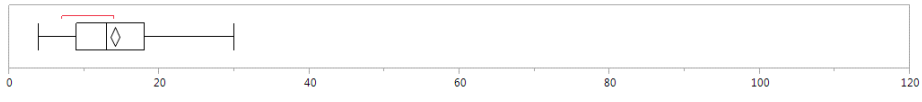
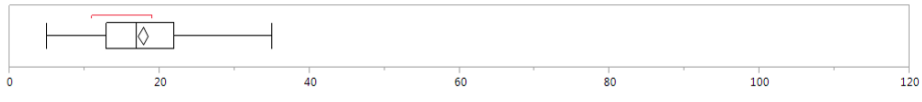
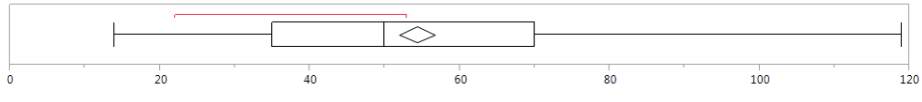
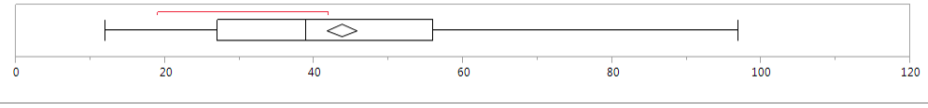
Figure 21: Distribution of filtered data for Python Level 1 Question 1

Our team continued to perform this cleaning process for all questions within Pythons level 1 and 2, in hopes of discovering a trend in the distribution of the questions. A set of all box-plot diagrams generated may be found in the Appendix under “New Findings: Box-Plot Diagrams of Duration for Python”.

New Findings

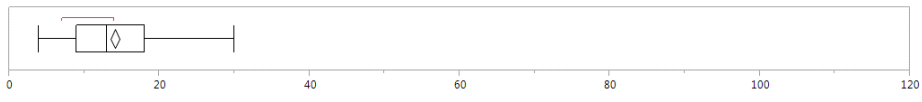
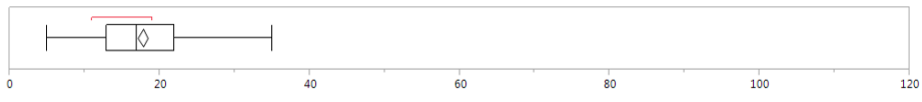
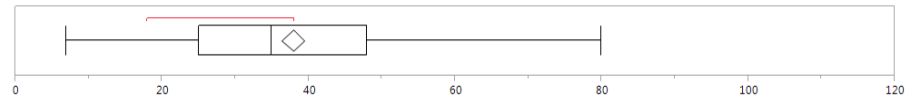
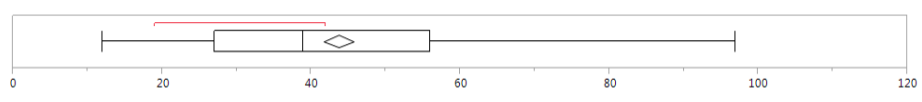
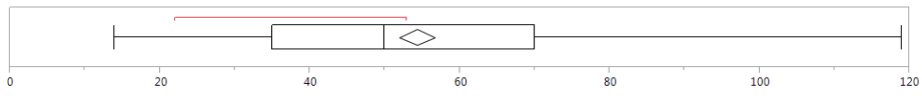
After our team attempted the questions on SingPath for ourselves, we discovered that there were several questions that were improperly ordered. For example, in Python Level 1, questions 5 to 9 were questions that clearly aimed to educate users on the use of variables in Python. However, a listing of the titles of the questions and their box-plot distributions revealed that they were arranged in a sub-optimal fashion.

Table 2: Original Arrangement of Questions

Question Number	Title	Box-Plot Diagram
5	Still More Variables	
6	Variables	
7	Another Variable	
8	More Fun with Variables	
9	Many Variables	

Our team then attempted to rearrange the questions in a format which would introduce students to the concepts of variables and gradually increase in difficulty. The results of our rearrangement (see Table 3) yielded an interesting observation with regards to the box-plot diagrams.

Table 3: Re-Arranged Question Sequence

Question Number	Title	Box-Plot Diagram
6	Variables	
7	Another Variable	
5	Still More Variables	
9	Many Variables	
8	More Fun with Variables	

Using the Median duration and spread of the results, the box-plot diagrams seem to indicate that the level of difficulty generally increases with the new arrangement of questions.

Sankey Diagram

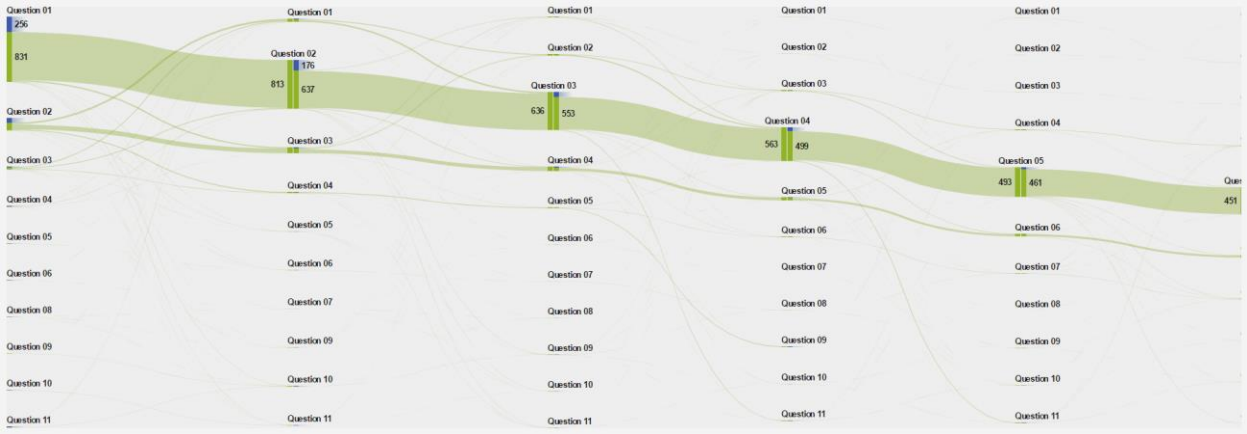


Figure 22: Sankey Diagram demonstrating the order of questions users attempted

Due to the non-linear, self-directed learning structure presented by SingPath, we would have anticipated that most students would jump between questions, seemingly at random, to tackle questions which they felt were of the appropriate level of difficulty or interested them. However, our Sankey diagram indicates that most students generally followed the sequence of questions presented to them on the website, with a handful of exceptions moving between questions at their own will.

This implies that students who attempt the questions on the site are being introduced to questions that do not necessarily meet the level of difficulty they expect, thus impacting the duration that they take in order to resolve a question.

Recommendations from Phase 2

1. Revise Order of Questions

Our team therefore proposes that the sequence of questions on the website be revised to allow students to attempt questions in an order that introduces them to concepts in a layered manner, gradually increasing in difficulty. This is based on the problem-based learning pedagogy, supported in other scholarly articles.

Problem-based learning identifies core competencies that students should demonstrate, and structure questions such that students must pick up these skills in sequence in order to proceed (Smith, 2014). There are varying levels of freedom in this pedagogy, where we may strictly restrict students to attempt questions only in the specified sequence, or where we may allow students to attempt any question as long as they have successfully clear some other question to demonstrate their capability.

The proposed questions and their respective orders may be found in Table 4 below. Do note that all questions analyzed come from the Python Language, levels 1 and 2, as these two levels offered the most complete dataset for analysis.

Table 4: Identified Question Groups in Python Levels 1 and 2

Level	Group	Question Numbers	Proposed Sequence	Description
1	Variables	5, 6, 7, 8, 9	6, 7, 5, 9, 8	This set of questions aims to educate the user on the use of variables
2	Integer Data Type	3, 5, 6, 7, 8	1, 11, 12, 3, 5, 6, 7, 8, 4, 13, 2, 9, 10*	Begins with an introduction to the Integer data type, followed by basic application examples.
2	String Data Type	2, 9, 10		Begins with an introduction to the String data type, followed by basic application examples.
2	Float Data Type	4, 13		Begins with an introduction to the Float data type, followed by basic application examples.

**note that for Python Level 2, additional questions 1, 11, and 12 have been included to demonstrate the flow of questions*

2. Add Dimensions to Firebase

Reviewing the data available from Firebase, there seems to be only one dimension viable for analysis: "Duration". This limits the amount of analysis that may be conducted on the dataset, and hinders analysts from developing a robust analytical model with multiple dimensions. To address this, the client should consider allowing Firebase to capture additional information with regards to the questions as well as the users on the site.

For the questions, the client may consider including information on the competencies demonstrated when a user completes a specific question. This would allow the client to track the learning journey each user takes while going through the site, as well as allowing the client to track which are the competencies which seem to prevent users from moving forward, causing them to stop attempting questions altogether. Another dimension would be to record the histories of all attempts on a single question for each user. The client has already begun to implement this by tracking the number of attempts and the duration a user takes for each attempt, however the correctness of the answer of the user for each attempt should also be captured in order to better ascertain the development and growth of a user.

For the users, basic demographic information should be captured, such as education level, race, age, and gender. Such information would tell the client a lot about the audience using their site, and enable the client to better customize his site. For example: If majority of the users on the site are students in the primary level of education, the language used on the site should probably be less formal and simpler to understand.

3. Debug and Update SingPath

There are multiple errors and bugs on the SingPath website, and this impacts the quality of the data collected on Firebase. Because of the errors and bugs, users are having a more difficult time answering questions, thus causing the distribution of duration to be messy and skewed.

Conclusions

Limitations Encountered

SingPath practices a self-directed learning structure, leaving the questions disorganized and in an order which may not be optimal for the learning experience of the users. Though students are not meant to tackle questions in sequence, our Sankey diagram (*see Figure 19*) demonstrates that this is not the case, and students are answering questions in the order that they are presented in.

There is also no enforced environment for the students, leaving them to dictate how much time they require to resolve the questions, which has led to several attempts on simple questions to take several hours or even a day to solve.

Attempts on questions within the database were being overwritten, leading to an incomplete picture of how students were using the website. The variables “Start Time” and “End Time” were being overwritten each time a student landed on the question page and submitted the question respectively, thus leading to situations where students apparently resolved complex questions in a matter of seconds.

Recommendations Generated

SingPath needs to be debugged and updated in order to resolve the issues such as pointer errors, missing prompts, and incomplete or incorrect instructions on the question pages. This would likely improve the data collected by the database as users will no longer need to struggle against the UI in order to complete the questions.

Reorganizing the questions on SingPath into a more appropriate order would likely yield a better learning experience for the students, as our team has discovered that there is a trend in the types of questions, their relative level of difficulty, and the resulting duration distribution among users (*see Tables 2 and 3*). A list of proposed orders of questions is shown in *Table 4*.

Finally, the database should capture additional information from the questions and users on the site. For questions, it currently only has a single dimension: Duration. Additional columns and dimensions such as competencies demonstrated would greatly improve the robustness of the data available. For users, demographic information would greatly enhance the understanding of the client of the users of the site, enabling them to better tailor the site to their tastes and preferences. Furthermore, information on the histories of question attempts should be recorded as well, with variables such as the duration of the attempt and the correctness of the submitted answer. This would allow for tracking of the progress of users on the site as they progress through the questions.

References and Supporting Literature

Below is a list of supporting scholarly articles which our team referred to whilst building this project.

- Smith, R. O. (2014). Beyond Passive Learning: Problem-Based Learning and Concept Maps to Promote Basic and Higher-Order Thinking in Basic Skills Instruction. Journal Of Research & Practice For Adult Literacy, Secondary & Basic Education, 3(2), 50-55.
- Davis, J. (2015). Education through Self-directed Learning. Australian Nursing & Midwifery Journal, 23(1), 26-27.
- Knowles, M. S. (1975). Self-directed learning.

Learning Journey

Our team began with this dataset and project in early December 2015, and it was the advice of our supervisor that prompted us to continue our search for alternative projects and datasets to use. We had continued our search for alternative datasets till February 2016, even approaching clients such as the National Library Board and National Environment Agency for data, but to no avail. And when we had finally knuckled down to proceed with the project, we worked closely with our supervisor as both parties knew we would need his experience to make this work.

It was during our interim report in early March 2016 where we had a review with our supervisor once again, and we collectively decided we could not responsibly handover the predictive model we had developed to our client, and attempted to analyze the project again from a new perspective. Fortunately, we were able to discover new aspects of the data through sheer grit and determination, and wound up with the project we have today. Through this project we had learned a valuable lesson: Complex analytical techniques are not always appropriate, and sometimes the simplest of statistical techniques can yield results. It was simple, basic statistics that allowed our group to derive recommendations for our client. And it was ultimately statistics that saved the day in this project.

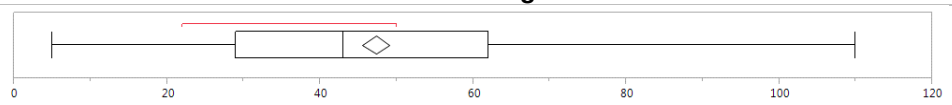
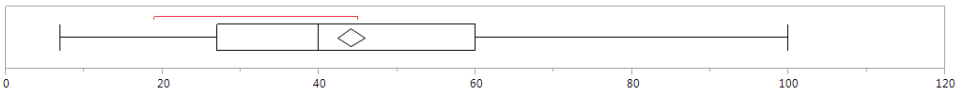
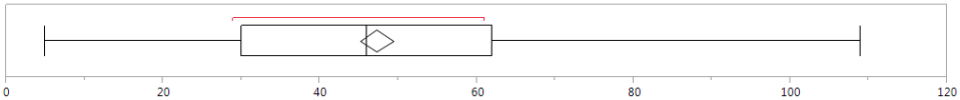
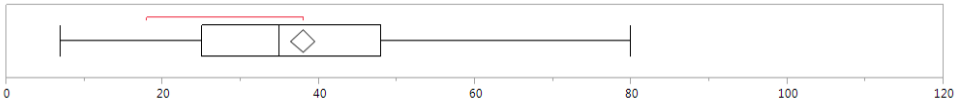
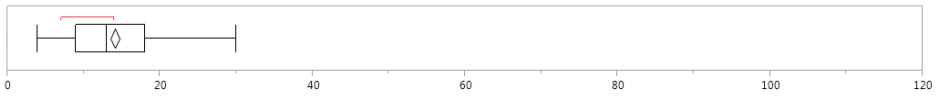
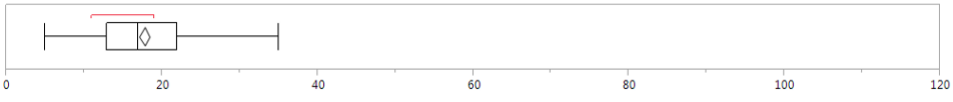
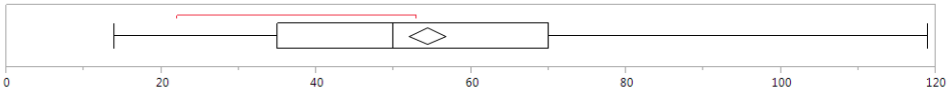
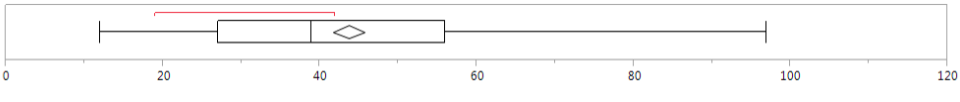
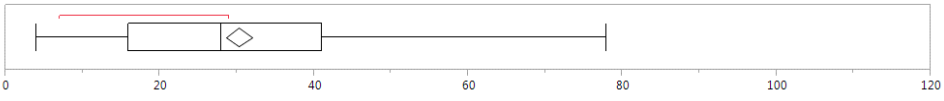
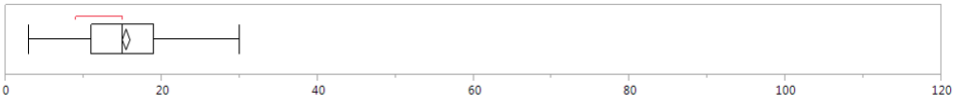
Appendix

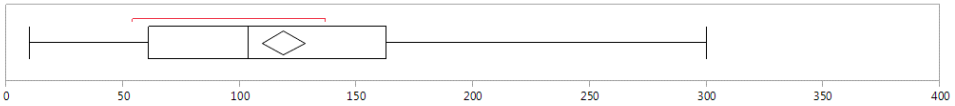
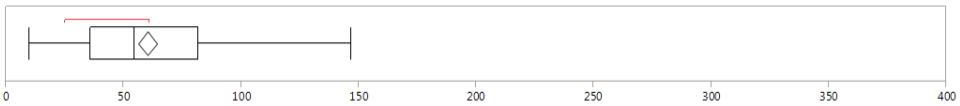
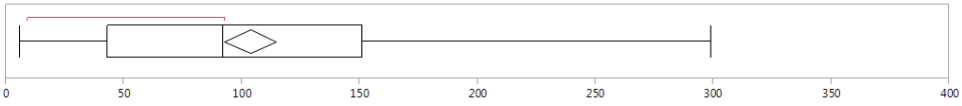
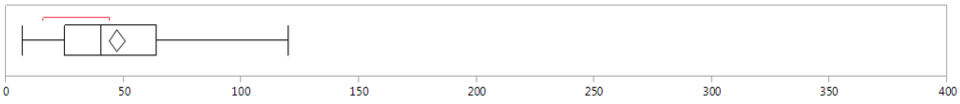
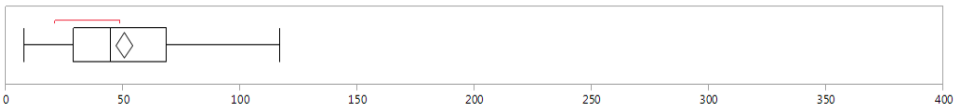
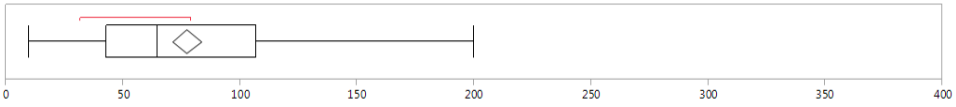
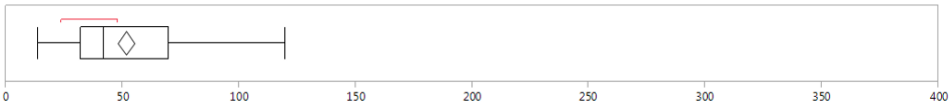
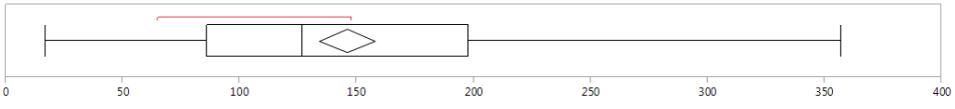
HTML Conversion Code

```
<!doctype html>
<html>
<head>
<link href="https://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css" rel="stylesheet">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular-sanitize.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/ng-csv/0.3.6/ng-csv.min.js"></script>
<!-- Firebase -->
<script src="https://cdn.firebase.com/js/client/2.2.4/firebase.js"></script>
<!-- AngularFire -->
<script src="https://cdn.firebase.com/libs/angularfire/1.1.3/angularfire.min.js"></script>
</head>

<body>
<div ng-app="myapp">
<div class="container" ng-controller="myctrl">
<div class="page-header">
<h1>ngCsv <small>example</small></h1>
</div>

<div class="form-group">
<label for="filename">Filename </label>
<input type="text" id="filename" class="form-control" ng-model="filename">
</div>

<div class="form-group">
<label for="separator">Field separator</label>
<input type="text" id="separator" class="form-control" ng-model="separator" ng-init="separator=','">
</div>

<div class="form-group">
<label for="decimal-separator">Decimal separator</label>
<input type="text" id="decimal-separator" class="form-control" ng-model="decimalSeparator" ng-
init="decimalSeparator='.'">
</div>

<button class="btn btn-default" ng-csv="getArray" csv-header="getHeader()" filename="{{ filename
}}.csv" field-separator="{{separator}}" decimal-separator="{{decimalSeparator}}">Export to CSV with
header</button>
</div>
</div>

<script>
var myapp = angular.module('myapp', ["ngSanitize", "ngCsv", "firebase"]);
myapp.controller('myctrl', function ($scope, $firebaseObject) {
    var ref = new Firebase("https://scorching-inferno-5500.firebaseio.com/singpath");
    $scope.data = $firebaseObject(ref);
    $scope.data.$loaded().then(function() {
        console.log("loaded");
        $scope.filename = "test";
        $scope.getArray = [];
        var language, level, questionNumber, question, user;
        var languageTemp, levelTemp, questionTemp;
        var problems = {};
        var levels = {};
        var questionsKeys = [];
        angular.forEach($scope.data.problems, function(languageValue, languageKey) {
            levels = {};
            angular.forEach(languageValue, function(levelValue, levelKey) {
                questionsKeys = [];
                angular.forEach(levelValue, function(questionValue, questionKey) {
                    questionsKeys.push(questionKey);
                });
                levels[levelKey] = questionsKeys;
            });
```

```
            problems[languageKey] = levels;
        });
        angular.forEach($scope.data.queuedSolutions, function(languageValue, languageKey) {
            languageTemp = languageKey;
            language = $scope.data.paths[languageTemp].title;
            angular.forEach(languageValue, function(levelValue, levelKey) {
                levelTemp = levelKey;
                level = $scope.data.levels[languageTemp][levelTemp].title;
                if (level.length == 7) {
                    level = level.substr(0,6) + "0" + level.substr(6,1);
                }
                angular.forEach(levelValue, function(questionValue, questionKey) {
                    questionTemp = questionKey;
                    questionNumber = "Question " +
(problems[languageTemp][levelTemp].indexOf(questionTemp) + 1);
                    if (questionNumber.length == 10) {
                        questionNumber = questionNumber.substr(0,9) + "0" + questionNumber.substr(9,1);
                    }
                    question = $scope.data.problems[languageTemp][levelTemp][questionTemp].title;
                    angular.forEach(questionValue, function(userValue, userKey) {
                        if(userValue.default.meta.history != null && userValue.default.meta.endedAt !=
null && userValue.default.meta.solved) {
                            var line = {};
                            line[1] = language;
                            line[2] = level;
                            line[3] = questionNumber;
                            line[4] = question;
                            line[5] = language + " " + level + " " + questionNumber + " " + question;
                            line[6] = language + " " + level;
                            line[7] = userKey;
                            for (var key in userValue.default.meta.history) {
                                var startDate = new Date(parseInt(key));
                                var startDate = startDate.getFullYear() + '/' + ('0' +
(startDate.getMonth() + 1)).slice(-2) + '/' + ('0' + startDate.getDate()).slice(-2) + ' ' + ('0' +
startDate.getHours()).slice(-2) + ':' + ('0' + startDate.getMinutes()).slice(-2) + ':' + ('0' +
startDate.getSeconds()).slice(-2);
                                var endDate = new Date(userValue.default.meta.endedAt);
                                var endDate = endDate.getFullYear() + '/' + ('0' + (endDate.getMonth()
+ 1)).slice(-2) + '/' + ('0' + endDate.getDate()).slice(-2) + ' ' + ('0' + endDate.getHours()).slice(-
2) + ':' + ('0' + endDate.getMinutes()).slice(-2) + ':' + ('0' + endDate.getSeconds()).slice(-2);
                                line[8] = startDate;
                                line[9] = endDate;
                                line[10] = Math.round((userValue.default.meta.endedAt - key) / 1000);
                                break;
                            }
                            $scope.getArray.push(line);
                        }
                    });
                });
            });
        });
    });
    $scope.getHeader = function () {return ["Language", "Level", "Question Number", "Question Title",
"Full Title", "Language Level", "User", "Start", "End", "Duration"]};
});
</script>
</body>
</html>
```

New Findings: Box-Plot Diagrams of Duration for Python

Level	Question	Box-Plot Diagram
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
1	8	
1	9	
1	10	
1	11	

Level	Question	Box-Plot Diagram
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	
2	8	
2	9	
2	10	
2	11	
2	12	
2	13	

Project Management

About the Stakeholders

About the Team

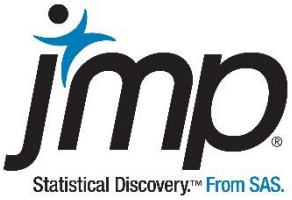
Technologies Used

Database: *Firebase*

Our client utilizes Firebase to store his data, and he will be sending us the data in JSON format. Storing text-based data in Firebase is quick and efficient, and having experienced pulling data on Webstorm, our team is confident that issues with this database will be minimal.

Languages: *HTML, CSS, JavaScript*

Languages used in the development of our application will be HTML, CSS, and JavaScript. HTML is a standardized system for customizing data and basic visuals on web pages, whilst CSS is a style sheet language for describing the presentation of a document. JavaScript is an object-oriented programming language for generating advanced visualization and interactions on web pages.



Analysis Tool: *JMP Pro 11*

Created in 1980 by the JMP Business unit of the SAS Institute, JMP Pro 11 is a desktop application software that allows users to analyze and visualize data with various tools integrated into its' framework. Tools available include, but are not restricted to:

- Predictive Modeling
- Cross-Validation
- Model Comparison
- Advanced Multi-Variate Techniques