

Process Book for LineUp D3 Implementation

By:

Wentao Du (wdu4@usfca.edu)

Siwei Zhang (szhang67@usfca.edu)

Kaijie Zhou (kzhou9@usfca.edu)

Github Repository:

The project is managed and stored on Github. Repo link:

<https://github.com/VictorDu/LineUp-D3>.

Proposal

Background and Motivation

Ranking is really popular these days. We use the ranking to choose our graduate program, our job position, our cars or even the restaurant we are going. Most of ranking tables are based on one attributes, such as a major of a University or MPG for a car. But sometimes, we need to compare different items based on multiple attributes. For instance, when we want to choose a nice University, we may take the quality of courses, the position and the employer reputation into consideration. A single ranking table cannot provide us with all these information at once. Furthermore, different attribute may have different weight for us. We may see the employer reputation much important than the position. It is difficult for us to do such comparison based on ranking tables. It would be great if there is a tool that can compare different items by multiple attributes with weight.

LineUp is a tool that can help us do this analysis. This software is described in the paper: [LineUp: Visual Analysis of Multi-Attribute Rankings](#).

The technique is really cool and can be helpful in many ways. However, the implementation for the paper is java based and can only be run on PC or Mac, which is not that convenient. We think that It would be much more useful if we can do a web-based implementation so that it can be run on any devices with a browser.

Project Objectives

LineUp is a novel and scalable visualization technique that uses bar charts. It supports the ranking of items based on multiple heterogeneous attributes with different scales and semantics, and let users to interactively combine attributes and flexibly refine parameters to explore the effect of changes in the attribute combination. This approach is generalizable and includes a wider set of considerations beyond expert use in a scientific domain. The functionality of it includes rank encoding, cause of rank encoding, multiple attributes support, filtering support, flexible mapping of attribute values to scores, scalability adapting, interactive refinement and visual feedback, and multiple ranking comparison.

The goal of this project is to implement a web-based LineUp by Javascript and D3.js. We'll try to implement the base functionality of LineUp such as interactive ranking and ranking scalability according to the paper, and use it to analyze our dataset. The primary question this project is trying to answer is that how multiple attributes contribute to a ranking and how changes in one or more attributes contribute to the rank.

Data Set

The data set we use to develop and test our project is [QS World University Rankings® - 2013/2014 TOP 100](#) which is used by the LineUp software as default data set.

In this data set, there are 100 items, each of which indicates an University. Each item contains the rank of the University in 2012 and 2013, as well as 7 scores that can be used to evaluate the University, including academic reputation, faculty/student ratio, International student and faculty score, and the total score. Each of them is an aspect of the quality of the University.

This data set is only used by developing. Our goal is to implement a general version LineUp. We can use it to show different ranking data set.

For test cases, We are also going to use is the [Cars93.csv](#) , which lists 93 cars on sale in the USA in 1993. It contains the price, MPG, horsepower, engine size, etc, which can be used to rank each car make.

Another data set we are going to use is the [Diamond.csv](#). It contains several attributes of diamonds such as carat, clarity, price and size

Data Processing

When the data set is read into the LineUp, we first clean up the data set and strip out NaN attributes.

The LineUp is a tool for ranking. So, the data we mainly focus on is numeric data. Numeric data will first be normalized and transformed into "score". All scores has a same fixed range, which makes it easier to compare and rank. Then the software will rank each item either by the mean or the max value of all its attributes, and show the result base on this ranking result.

In the process of normalizing, user can choose a normalizing function. The function can set the range of the score, or filter out some unexpected data. By default, data are just normalized into range 0~1.

In the ranking process, there is two ways to do. One way is to compare the items by the total score of all its weighted attributes (serial combination). The other way is to compare by the max value of all the attributes (parallel combination).

Visualization Design

In the LineUp, data is represented as stacked bar chart in a table. Each row indicates an item and each column indicates an attribute. The length of bar depends on the normalized data of that attribute and the width of that column. The width of each column indicates the importance of that attribute. User can change the width according to their will.

When user change the width of a column, the ranking will change. When the rank of an item goes up, the rank column of that item will get green. If it goes down, the column will get red. User can easily find out the change through this color changing.

Another visualization technique LineUp use is the “snapshot”. The snapshot just makes another ranking beside the original one. User can apply different ranking strategies on the two ranking tables. Between two snapshots, user can find the position of same items through the “line up”, which is a line that connect the same item between the two table. The snapshot and lines make it much easier to find out the change of ranking in different strategies.



Must-Have Features

Normalize

Normalizing data of data set into a fixed range (e.g. 0~1).

Parallel / serial combination

Sort the items by the maximum attribute value or by the total value of all the attributes.

Snapshot

Generate another rank table with the same or different ranking strategy.

Connection

Connect the same item in different rank tables.

Ranking change

Change the ranking when the weight(or width) of an attribute change. When the rank of an item goes down, the color of its rank change red. If it goes up, the color change green.

Optional Features

Memo pad

An area to store attributes or user-defined columns that is deleted.

Data mapping

A dashboard that can be used to define the mapping between raw data and score range.

Grouping

Attributes can be grouped into different groups. Each group has their own stacked bar chart.

Schedule

3/24	Project Proposal
3/31	Revised Proposal and Annotated Bibliography
4/12	Alpha Release: <ul style="list-style-type: none">• Implement a basic static ranking table based on the Top100 dataset with default normalization (range 0~1).• Implement the snapshot and connection
4/26	Beta Release: <ul style="list-style-type: none">• Add ranking change feature to the ranking table.• Make option for user to choose the combination and the weight of each attribute.
5/9	Code Freeze: <ul style="list-style-type: none">• Fully test basic features on different data sets.• Implement optional features if possible.
5/10-5/12	Final Project Presentation: <ul style="list-style-type: none">• Do the slides and prepare for the presentation
5/19	Report, Source Code, Presentations slides, and User Manual

References

S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, M. Streit. LineUp: Visual Analysis of Multi-Attribute Rankings. IEEE Transactions on Visualization and Computer Graphics, 19(12): 2277 - 2286, 2013

M.Behrisch, J.Davey, S.Simon, T.Schreck, D.Keim,and J.Kohlhammer. Visual comparison of orderings and rankings. In Proceedings of the EuroVis Workshop on Visual Analytics (EuroVA '13), 2013

L. Byron and M. Wattenberg. Stacked graphs - geometry & aesthetics. IEEE Transactions on Visualization and Computer Graphics, 14(6):1245 –1252, 2008.

C. G. Healey. Choosing effective colours for data visualization. In Proceedings of the IEEE Conference on Visualization (Vis '96), pages 263–270. IEEE Computer Society Press, 1996.

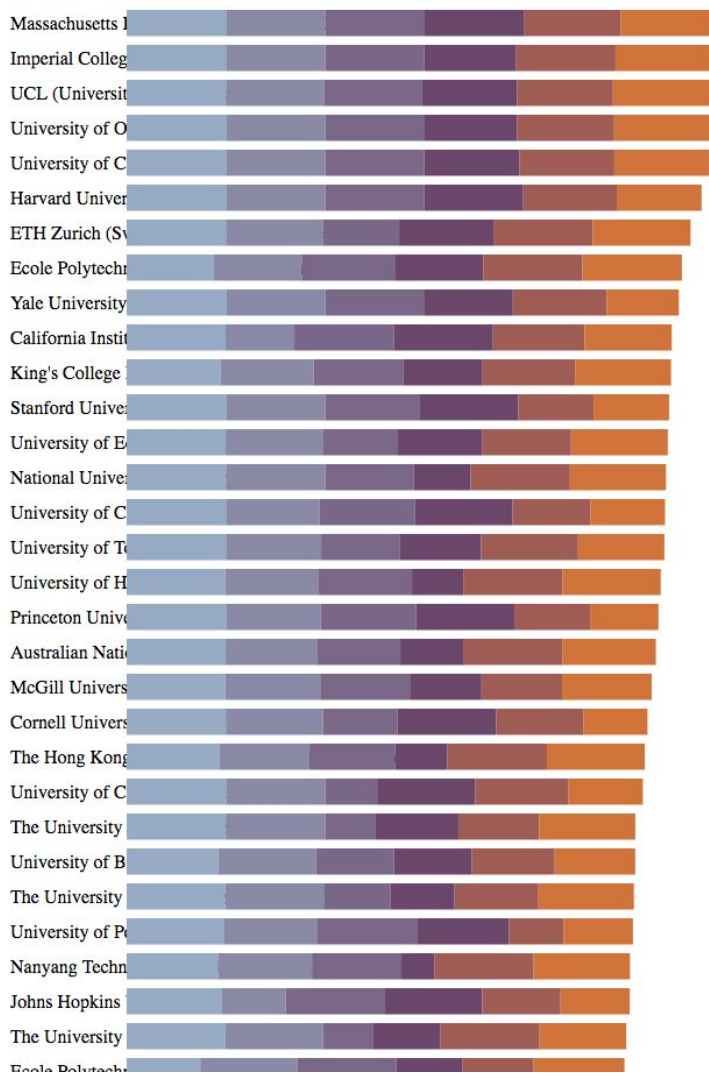
P.Kidwell,G.Lebanon,andW.S.Cleveland.Visualizingincompleteand partially ranked data. IEEE Transactions on Visualization and Computer Graphics, 14(6):1356–1363, 2008.

Alpha Release

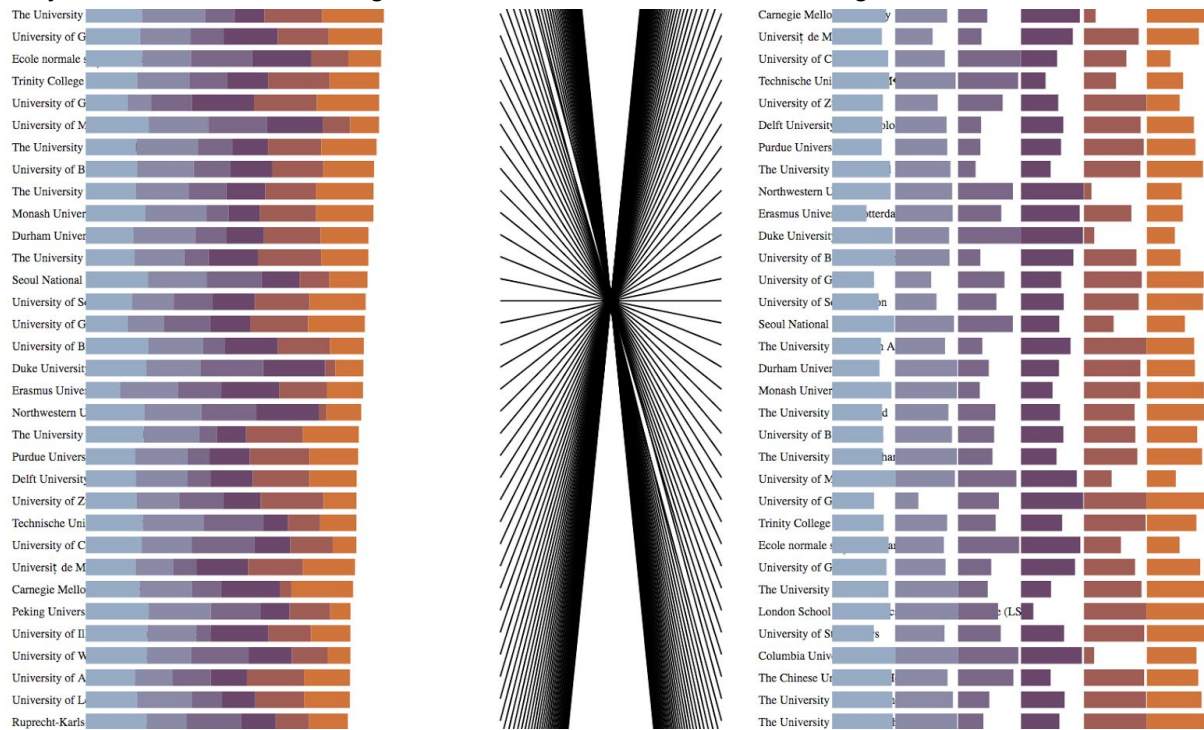
For Alpha release, we built an outline for the project.

Achievements

The main part of this Lineup project is several tables of stacked horizontal bar chart. We implement the horizontal bar chart referring the [example](#) online. In our implementation, the bars in each table can be sorted by different sorting function. In the demo, we give 2 different sorting functions: ascending and descending order.



Besides, we also implement lines to connect the same item in different tables. To show it clearly, we make the ascending ordered table next to the descending ordered table.



From the demo above, you can also see another kind of table which contains independent bar charts for each attributes instead of one stacked bar chart for each item. This can be an option in the future for users to choose.

Future Work

The demo now is static. In the future, we need to add animation to this graph.

- Make the boundary of each attribute draggable.
- Implement the run-time sorting feature.
- Implement the highlight feature for each item.

Beta Release

In this release, we implement several core functions for Lineup based on the Alpha release.

Achievements

The main update for this release is that we add several animation to implement the core functionality of the Lineup.

1. Draggable edge and run-time ranking:

We implemented a draggable edge for each attribute. The width of each column or attribute indicates the weight of the attribute in the ranking function. Users can change the weight of each attribute by dragging the edge and change the width of that column.

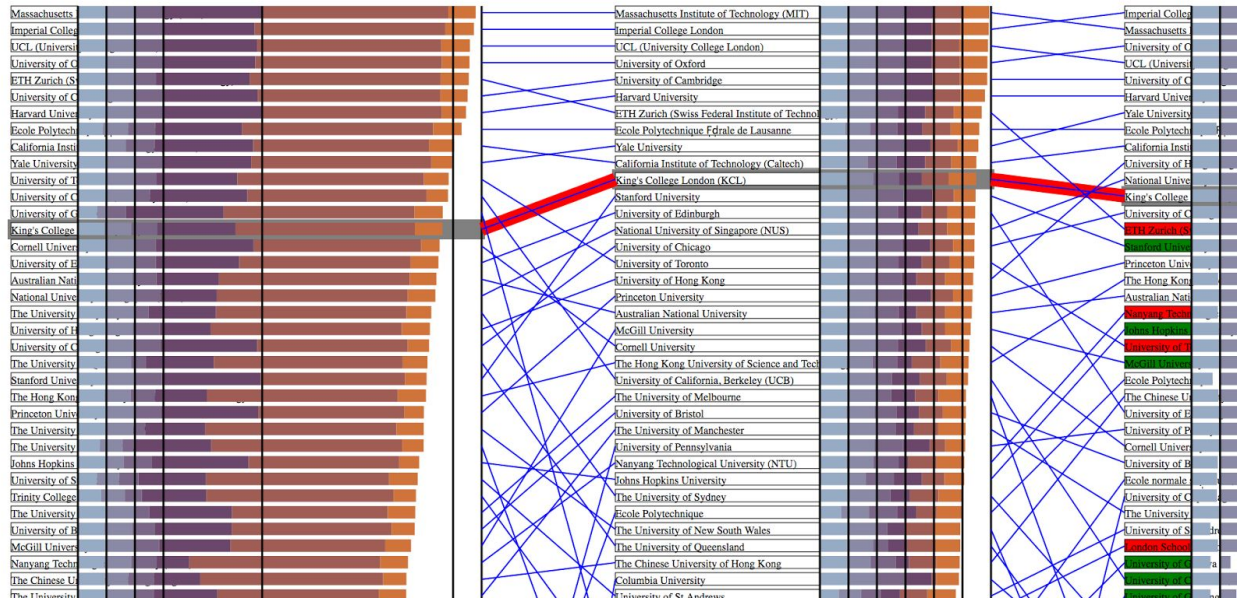


We also implemented the run-time ranking feature for the project. The change of an attribute's weight will trigger the re-ranking of the table, and each item will move to find their new position in the table. For the movement of the bar chart, we referred this [example](#).

3. Interactive “line up”:

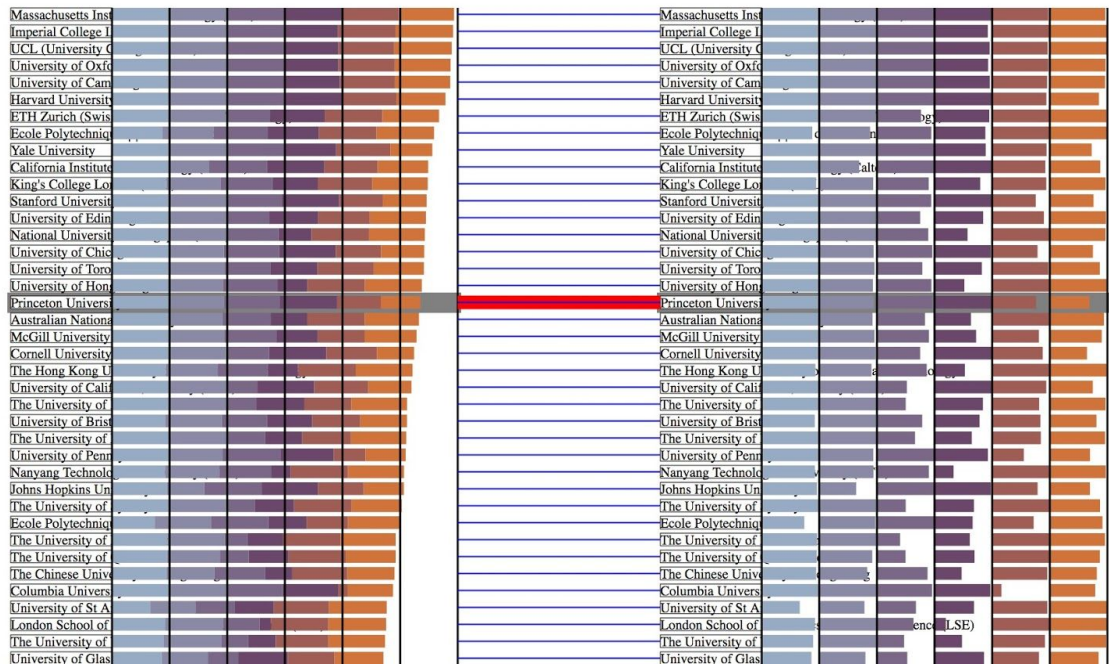
When the ranking of one table is changed, it is helpful to line up each item in this table with the same item in another table. Though this line up function, users can easily compare different rankings.

To see the change of a specific item in different ranking, we add highlight feature to the bar chart tables. When we point at an item in one of the tables, this item in all tables as well as the lines will be highlighted.



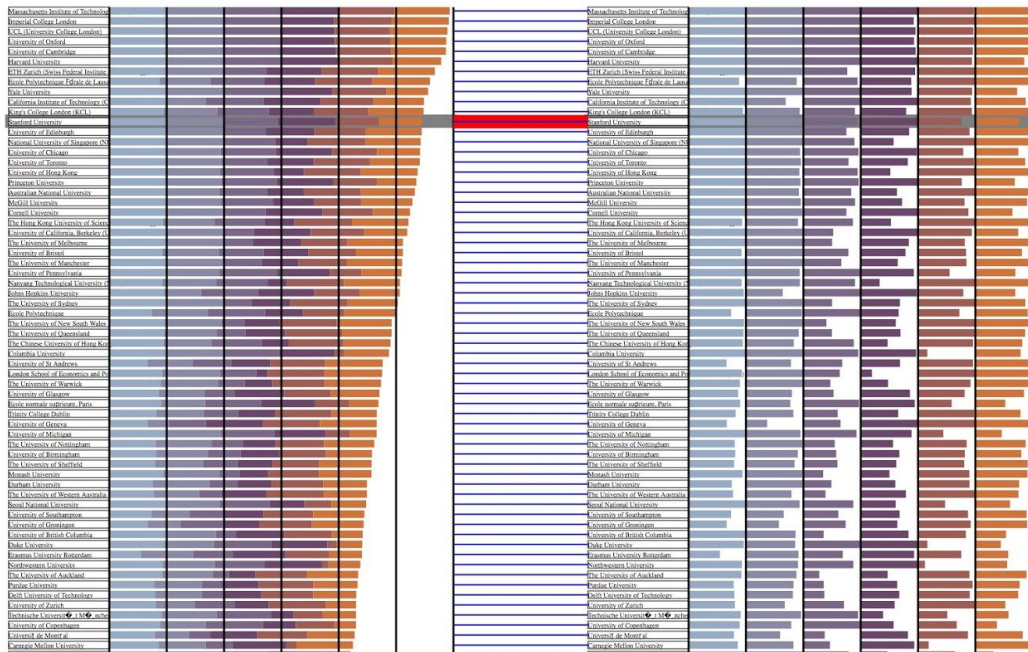
4. Optimize the view of the project:

Another thing we did is to optimize the view of the page. In the code, we made the font and table size configurable. We finally decide to use the configuration looks like this:



It is quite clear and balance.

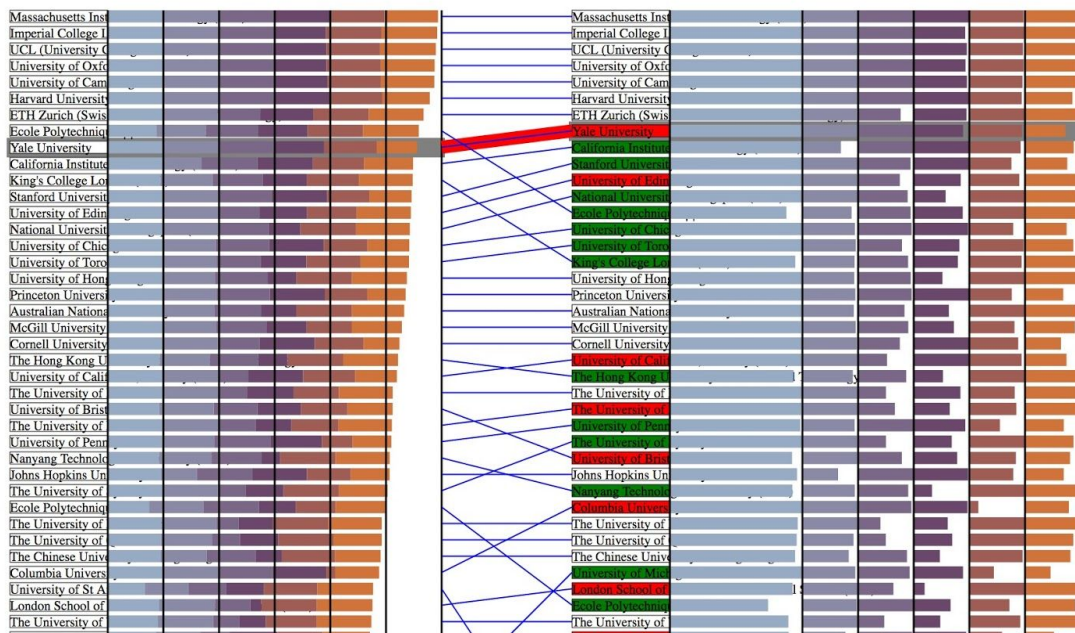
We also saved some bad configuration examples:



For the example above, it is too dense. Although it can show lots of items in one page, it is hard to recognize the labels.



The example above is way to loose. We can read the labels much better, but we may also get a long page to drag down.



The example above gives an example of short lines between tables. Actually, the distance between the two tables should not be too short. If the two tables are close to each other, the lines may become a mass, and hard to recognize.

Future Work

For the future work, we'll try to optimize the project.

- Add labels to each attributes
- Make this visualization compatible with the other two data sets.

Final Release

For this release, we optimized the user interface and modified the code in a more generic way.

Achievements

1. Data Preprocessing Function

The goal of our project is to implement a visualization technology. So it is important to make the project compatible with different datasets

The first thing we did is to generalize our javascript function. We restricted the input for our ranking generate function, and provided the interface for index to call. We decided to use Json files as the input of our function. First of all, Json file contains key-value pairs. We can get multiple configuration parameters through a single Json file. This will make the programming much easier. Also, Json file is easy to read and understand for users, and, since it is wide used in network, we can easily handle this format through different languages. This is really important because we need to use other languages to preprocess the dataset.

After unifying the input of the function, the next thing we need to do is to make the input datasets follow the rule of our input. The format of datasets are various, even though all of them are csv files. We cannot restrict the format of input datasets, so we need to do the preprocessing to change it into the format we want before sending it to the javascript.

We decide to write a R script to do the preprocessing, because R has more build in functions to deal with datasets and data processing. The idea of the preprocessing is just convert each attributes to scores that can be used to do the ranking. The problem is that some time the scores make no sense. For instance, in the cars dataset, the make of each car is not suitable for scoring. So we need user tell us which attributes they want to be scored and used for ranking. These information are also based on a Json configuration file, which should be provided by the users.

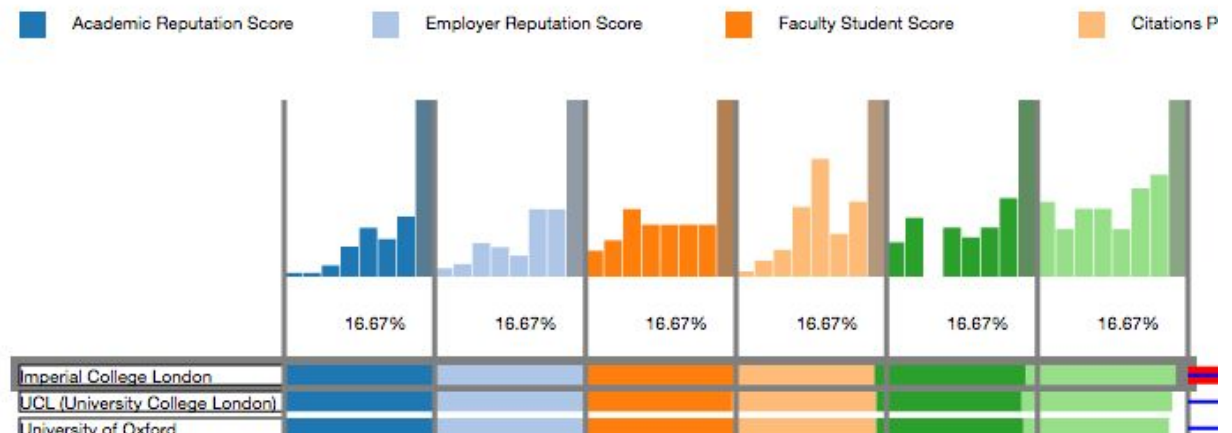
So, the workflow here is: a. User provide a json config and a csv dataset to the R script.

- b. R script process the data based on the user's configuration and dataset, then generate a new formatted dataset and a new json config for javascript.
- c. Our javascript function accept the new format dataset and the new json file to show the ranking.

The whole workflow is connected by json, which is easier for the extension in the future.

2. Header for Ranking Table

Another thing we did is that we add the title column for the ranking table.

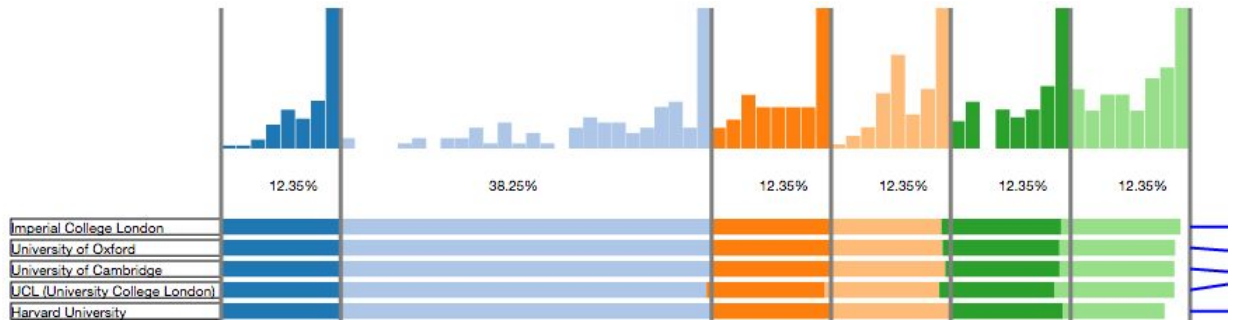


As we can see from the figure above. We use color and legend to indicate each column or attribute, and the title of each column will contains two parts: histogram and the weight of the attribute.

The histogram is used to see the distribution of this attribute among all the items. User can see the overall scores of this ranking, and have a brief understanding of this dataset. We also make the high light synchronized with the histogram to let user know where this item at in the whole dataset. It is really useful for user to analyze data.

We show the weight of the attribute by two ways. One is the width of the column, which is defined in the Lineup paper. To make it more clear and accurate for user to see the weight, we also show them in percentage. We make the total weight as 100%, and make the attributes divide this total weight. At the beginning, all weights are the same. So you can see in the figure above, each weight is 16.67%.

When user change the weight, the number will change. The histogram will also change its width and show the distribution more in detail. You can see the change below:



3. User Interface

In this release, we also create a control panel for users to easily control what to show.

Lineup D3 Impl BY : Wentao Du Siwei Zhang Kaijie Zhou

☒ University Rank
 ☐ Cars
 ☐ Diamond

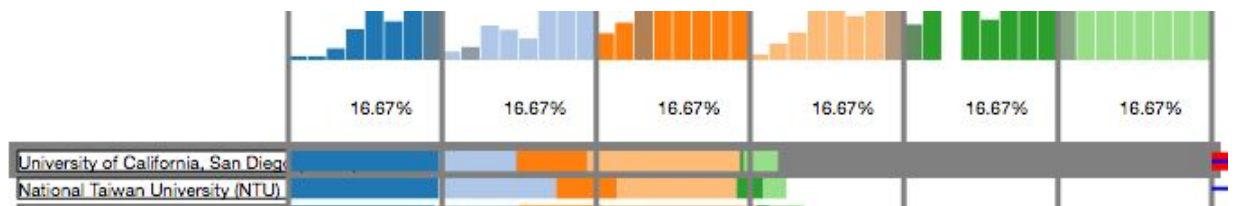
Ranking1: Stacked bar
 Ranking2: Separated bar
 Ranking3: Hide

☐ Ascending order
 Submit

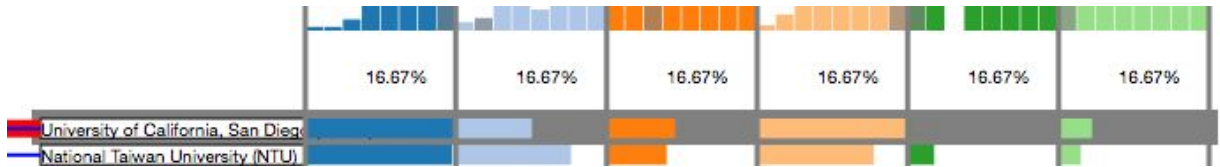
The panel is built in Bootstrap, and give user options to show the ranking.

For dataset option, we now have three example datasets to show: University ranking, Cars, and Diamonds. We can add more datasets in the future and it's not difficult to do so, for all the data cleaning functions and showing functions are generic.

On the index page, we can now show three ranking at same time at most. The limitation here is just for better performance. Four or more ranking can make the page super width, and it is painful to scroll the page at that width. User can choose whether to hide or show each ranking or show in which format. The format of the bars are stacked or separated. Stacked bars will show the bar in a form of the sum of all weighted attributes just like this:

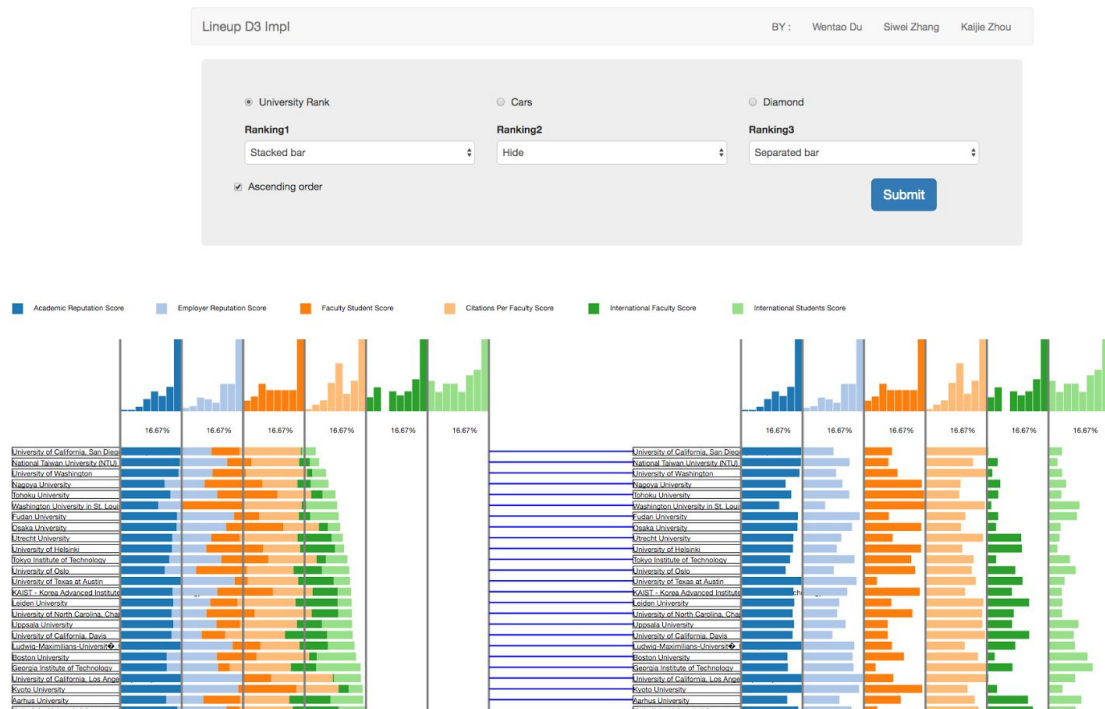


The separated bars will show the bar for each attribute separately. This form will help user to see the score of each column more clear.



There is also an option about the order of showing. By default, the ranking will be in descending order, because mostly we want the score to be high. But we also support user to show the ranking in ascending order just by checking the option.

So, when everything set will, user can submit their configuration by clicking the “Submit” button. Then the page will be shown as this:



Future Work

In this release, we improved the project basing on the basic function we implemented in last release. But there are still something to do after this.

1. We can try to add more dataset to our project to see how it works.
2. We can adjust the font and color and refine the display of the project

Also, based on the feedback from the final presentation, we need to modify the user interface alittle. We need to add the default visualization of the index, and change the select bars into radios. And remove the “submit” button.

Post Final Update

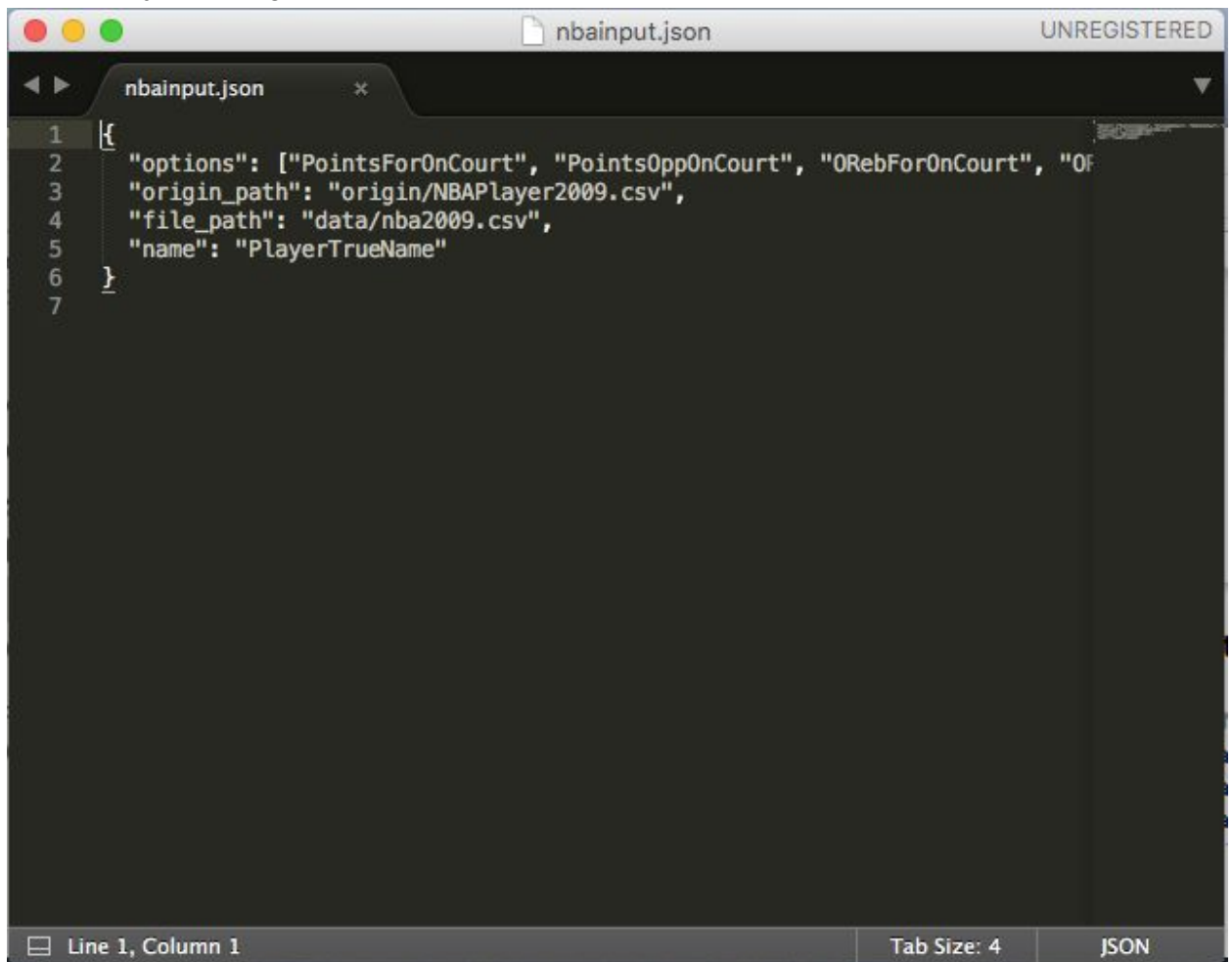
From the final presentation, we got some feedback from the professor and classmates. So we made some updates to our project, including a new dataset and some modification on the control panel.

Achievements

1. NBA Player 2009 Dataset

To how our project compatible with different dataset, we decide to test it with one more dataset. We choose a dataset about the performance of NBA players in 2009, which we found online. The dataset has lots of attributes and is really complicated.

I created a json configuration file for it:

A screenshot of a code editor window titled 'nbainput.json' with a 'UNREGISTERED' label in the top right corner. The editor shows a JSON configuration file with the following content:

```
1 {  
2   "options": ["PointsForOnCourt", "PointsOppOnCourt", "ORebForOnCourt", "OF  
3   "origin_path": "origin/NBAPlayer2009.csv",  
4   "file_path": "data/nba2009.csv",  
5   "name": "PlayerTrueName"  
6 }  
7
```

The status bar at the bottom indicates 'Line 1, Column 1', 'Tab Size: 4', and 'JSON'.

Then make it together with the csv file as inputs to the script.

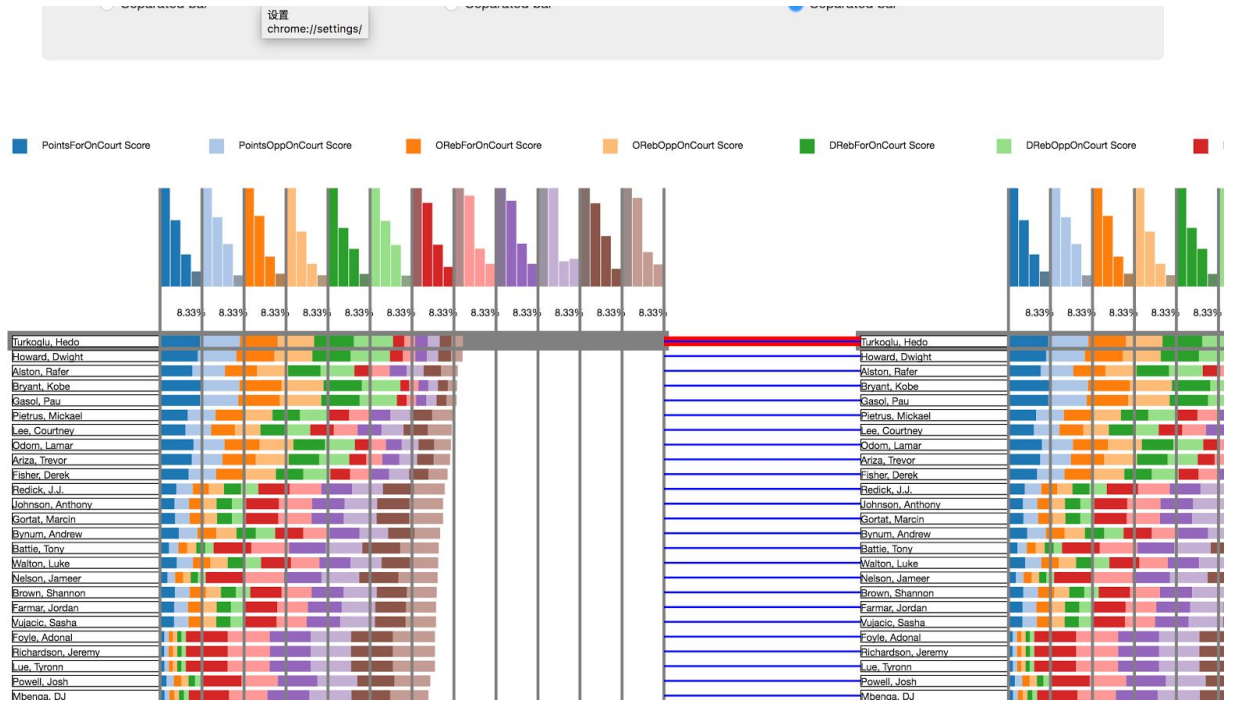
```
1. bash
[kaijiezhou@Kaijies-MacBook-Pro LineUp-D3 (post-final)]$ Rscript clean_lineup_data.R origin/nbainput.json data/nba2009.json
Loading required package: methods
{
  "options": ["PointsForOnCourt Score", "PointsOppOnCourt Score", "ORebForOnCourt Score", "ORebOppOnCourt Score", "DRebForOnCourt Score", "DRebOppOnCourt Score", "PointsForOffCourt Score", "PointsOppOffCourt Score", "ORebForOffCourt Score", "ORebOppOffCourt Score", "DRebForOffCourt Score", "DRebOppOffCourt Score"],
  "origin_path": "origin/NBAPlayer2009.csv",
  "file_path": "data/nba2009.csv",
  "name": "PlayerTrueName"
}
[kaijiezhou@Kaijies-MacBook-Pro LineUp-D3 (post-final)]$
```

It works. The output of the command is the new configuration file.

To add the new dataset to the index page is not hard because we've made the JS function generic enough, and the html code is scalable. I just add few lines of code and it shows.

We can actually add the upload functionality to let user upload their customized dataset. But that needs a server to support. Since we have no time to write a web server for it, we have to leave those things hard coded on our index page.

The ranking shows like this:



2. New Control Panel

According to the suggestion from our professor, we changed the select bar into radios for the bar style selection.

To make the style selection more clear to user, I change the selection bar to radios. Also, we removed the submission button, and make the change every time user make the change.

The new user interface is like this:

Lineup D3 Impl BY : Wentao Du Siwei Zhang Kaijie Zhou

Data Sets

☒ University Rank ☐ Cars ☐ Diamond ☐ NBA Player 2009

Ranking display

☐ Ascending order

Ranking1

☐ Hide ☒ Stacked bar ☐ Separated bar

Ranking2

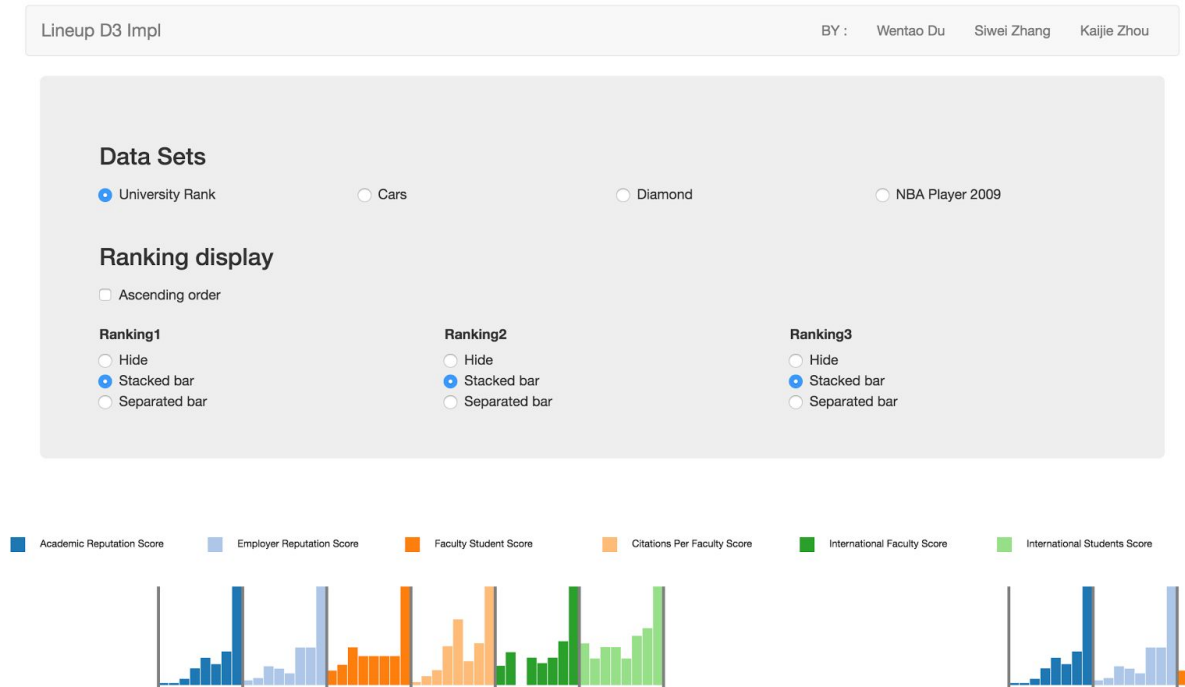
☐ Hide ☒ Stacked bar ☐ Separated bar

Ranking3

☐ Hide ☐ Stacked bar ☒ Separated bar

We also give the index default values, so that the user can see some default ranking when they first come to the index.

This is the index page when user first come and see.



Future Work

This is the final version of this simulator. In the future, there is also something we can do to improve the project.

1. Integrate the R script into the project. Make the script clean the data and pass the data for showing automatically.
2. Let user upload their own configuration file and dataset for showing.

The above targets all need a web server to support. So a web server is needed to be implemented to achieve these goals.

Useful Links

1. Project demo link: <http://victordu.github.io/LineUp-D3/>
2. Project Github: <https://github.com/VictorDu/LineUp-D3>