

# LineUp: Visual Analysis of Multi-Attribute Rankings

Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit

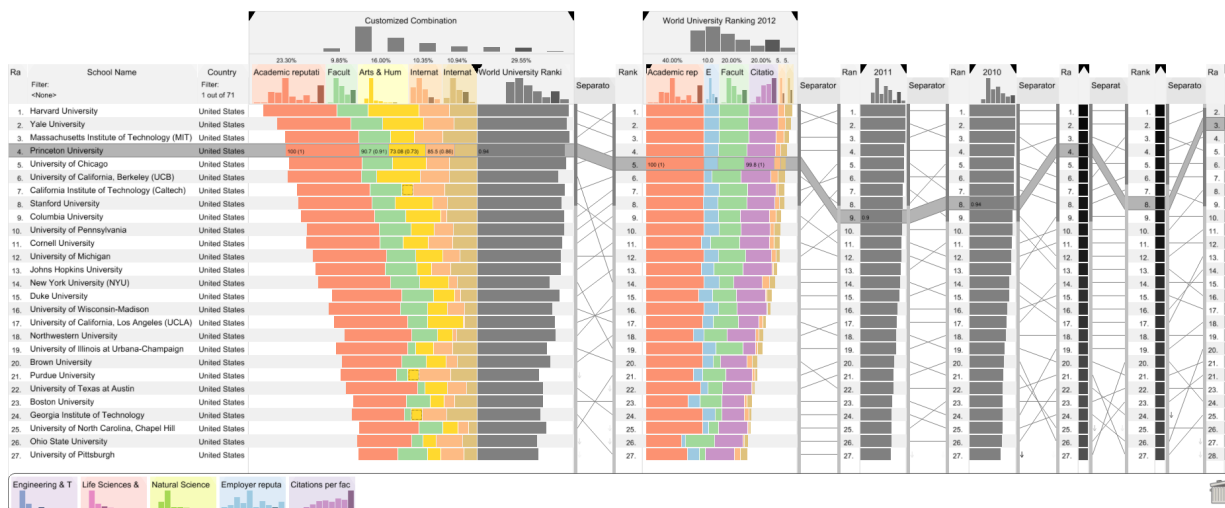


Fig. 1. LineUp showing a ranking of the top Universities according to the QS World University Ranking 2012 dataset with custom attributes and weights, compared to the official ranking.

**Abstract**—Rankings are a popular and universal approach to structuring otherwise unorganized collections of items by computing a rank for each item based on the value of one or more of its attributes. This allows us, for example, to prioritize tasks or to evaluate the performance of products relative to each other. While the visualization of a ranking itself is straightforward, its interpretation is not, because the rank of an item represents only a summary of a potentially complicated relationship between its attributes and those of the other items. It is also common that alternative rankings exist which need to be compared and analyzed to gain insight into how multiple heterogeneous attributes affect the rankings. Advanced visual exploration tools are needed to make this process efficient. In this paper we present a comprehensive analysis of requirements for the visualization of multi-attribute rankings. Based on these considerations, we propose *LineUp* - a novel and scalable visualization technique that uses bar charts. This interactive technique supports the ranking of items based on multiple heterogeneous attributes with different scales and semantics. It enables users to interactively combine attributes and flexibly refine parameters to explore the effect of changes in the attribute combination. This process can be employed to derive actionable insights as to which attributes of an item need to be modified in order for its rank to change. Additionally, through integration of slope graphs, LineUp can also be used to compare multiple alternative rankings on the same set of items, for example, over time or across different attribute combinations. We evaluate the effectiveness of the proposed multi-attribute visualization technique in a qualitative study. The study shows that users are able to successfully solve complex ranking tasks in a short period of time.

**Index Terms**—Ranking visualization, ranking, scoring, multi-attribute, multifactorial, multi-faceted, stacked bar charts

## 1 INTRODUCTION

We encounter ranked lists on a regular basis in our daily lives. From the “top at the box office” list for movies to “New York Times Best-sellers”, ranked lists are omnipresent in the media. Rankings have the

- Samuel Gratzl is with Johannes Kepler University Linz.  
E-mail: samuel.gratzl@jku.at.
- Marc Streit is with Johannes Kepler University Linz.  
E-mail: marc.streit@jku.at.
- Alexander Lex is with Harvard University.  
E-mail: alex@seas.harvard.edu.
- Hanspeter Pfister is with Harvard University.  
E-mail: pfister@seas.harvard.edu.
- Nils Gehlenborg is with Harvard Medical School.  
E-mail: nils@hms.harvard.edu.

*Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.*

For information on obtaining reprints of this article, please send e-mail to: [tvcg@computer.org](mailto:tvcg@computer.org).

important function of helping us to navigate content and provide guidance as to what is considered “good”, “popular”, “high quality”, and so on. They fulfill the need to filter content to obtain a set that is likely to be interesting but still manageable.

Some rankings are completely subjective, such as personal lists of favorite books, while others are based on objective measurements. Rankings can be based either on a single attribute, such as the number of copies sold to rank books for a bestseller list, or on multiple attributes, such as price, miles-per-gallon, and power to determine a ranking of affordable, energy-efficient cars. Multi-attribute rankings are ubiquitous and diverse. Popular examples include university rankings, rankings of food products by their nutrient content, rankings of computer hardware, and most livable city rankings.

When rankings are based on a single attribute or are completely subjective, their display is trivial and does not require elaborate visualization techniques. If a ranking, however, is based on multiple attributes, how these attributes contribute to the rank and how changes in one or more attributes influence the ranking is not straightforward to understand. In order to interpret, modify, and compare such rankings, we need advanced visual tools.

When interpreting a ranking, we might want to know why an item has a lower or a higher rank than others. For the aforementioned university rankings, for example, it might be interesting to analyze why a particular university is ranked lower than its immediate competitors. It could either be that the university scores lower across all attributes or that a single shortcoming causes the lower rank.

Another crucial aspect in multi-attribute rankings is how to make completely different types of attributes comparable to produce a combined ranking. This requires mapping and normalizing heterogeneous attributes and then assigning weights to compute a combined score. A student trying to decide which schools to apply to might wish to customize the weights of public university rankings, for example, to put more emphasis on the quality of education and student/faculty ratio than on research output. Similarly, a scientist ranking genes by mutation frequency might want to try to use a logarithmic instead of a linear function to map an attribute.

Another important issue is the comparison of multiple rankings of the same items. Several publications, for example, release annual university rankings, often with significantly different results. A prospective student might want to compare them to see if certain universities receive high marks across all rankings or if their ranks change considerably. Also, university officials might want to explore how the rank of their own university has changed over time.

Finally, if we can influence the attributes of one or more items in the ranking, we might want to explore the effect of changes in attribute values. For example, a university might want to find out whether it should reduce the student/faculty ratio by 3% or increase its research output by 5% in order to fare better in future rankings. If costs and benefits can be associated with these changes, such explorations can have an immediate impact on strategic planning.

Interactive visualization is ideally suited to tailoring multi-attribute rankings to the needs of individuals facing the aforementioned challenges. However, current approaches are largely static or limited, as discussed in our review of related work. In this paper we propose a new technique that addresses the limitations of existing methods and is motivated by a **comprehensive analysis of requirements of multi-attribute rankings considering various domains**, which is the first contribution of this paper. Based on this analysis, we present our second contribution, **the design and implementation of LineUp, a visual analysis technique for creating, refining, and exploring rankings based on complex combinations of attributes**. We demonstrate the application of LineUp in two use cases in which we explore and analyze university rankings and nutrition data.

We evaluate LineUp in a qualitative study that demonstrates the utility of our approach. The evaluation shows that users are able to solve complex ranking tasks in a short period of time.

## 2 REQUIREMENT ANALYSIS

We identified the following requirements based on research on the types and applications of ranked lists, as well as interviews and feedback from domain experts in molecular biology, our initial target group for the application. We soon found, however, that our approach is much more generalizable, and thus included a wider set of considerations beyond expert use in a scientific domain. We also followed several iterations of the *nested model for visualization design and validation* [17] and looped through the four nested layers to refine our requirements (i.e., *domain problem characterization*). We concluded our iterations with the following set of requirements:

**R I: Encode rank** Users of the visualization should be able to quickly grasp the ranks of the individual items. Tied ranks should be supported.

**R II: Encode cause of rank** In order to understand how the ranks are determined, users must be able to evaluate the overall item scores from which the ranking is derived and how they relate to each other. In many cases, scores are not uniformly distributed in the ranked list. For example, the top five items might have a similar score, while the gap to the sixth item could be much bigger. Depending on the application, the first five items might thus be much more relevant than the sixth.

To achieve this, users should see the distribution of overall item scores and their relative difference between items and also be able to retrieve exact numeric values of the scores. If item scores are based on combinations of multiple attribute scores (see R III), the contribution of individual attributes to the overall item score should also be shown.

**R III: Support multiple attributes** To support rankings based on multiple attributes, users must be able to combine multiple attributes to produce a single all-encompassing ranking. It is also important that these combinations are salient. To make multiple attributes comparable, they must be normalized, as described in R V. In the simplest case, users want to combine numerical attributes by summing their scores. This combined sum of individual attribute scores then determines the ranking. However, more complex combinations, including weights for individual attributes and logical combinations of attributes, are helpful for advanced tasks (see Section 4.2).

**R IV: Support filtering** Users might want to exclude items from a ranking for various reasons. For example, when ranking cars, they might want to exclude those they cannot afford. Hence, users must be able to filter the items to identify a subset that supports their task. Filters must be applicable to numerical attributes as ranges, nominal attributes as subsets, and text attributes as (partial) string matches.

### **R V: Enable flexible mapping of attribute values to scores**

Attributes can be of different types (e.g., numerical, ordered categorical), scales (e.g., between 0 and 1 or unbounded) and semantics (e.g., both low and high values can be of interest). The ranking visualization must allow users to flexibly normalize attributes, i.e., map them to normalized scores. For example, when using a normalization to the unit interval  $[0, 1]$ , 1 is considered “of interest” and 0 “of no interest”. While numerical attributes are often straightforward to normalize through linear scaling, other data types require additional user input or more sophisticated mapping functions.

Numerical attributes can have *scales* with arbitrary bounds, for instance, from 0 to 1 or  $-1$  to 1. They might also have no well-defined bounds at all. However, for a static and known dataset the bounds can be inferred from the data. For instance, while there is no upper limit for the number of citizens in a country, an upper bound can be inferred by using the number of citizens in the largest country as bound. In addition, values in a range can have different meanings. For example, if the attribute is a *p*-value expressing statistical significance, it ranges from 0 to 1, where 0 is the “best” and 1 the “worst”. In other cases, such as log-ratios, where attribute values range from  $-1$  to 1, 0 could be the “worst” and the extrema  $-1$  and  $+1$  the “best”. Additionally, users might be interested in knowing the score of an attribute without the attribute actually influencing the ranking, thereby providing contextual information.

**R VI: Adapt scalability to the task** While it is feasible to convey large quantities of ranked items using visualization, there is a trade-off between level of detail (LoD) and scalability. Where to make that trade-off depends largely on the given task. In some tasks only the first few items might be relevant, while in others the focus is on the context and position of a specific item. Also, some tasks may be primarily concerned with how multi-attribute scores are composed, while in other tasks individual scores might be irrelevant. A ranking visualization technique must be designed either with a specific task in mind or aim at optimizing the trade-off between LoD and scalability.

**R VII: Handle missing values** As real-world data is often incomplete, a ranking visualization technique must be able to deal with missing values. Trivial solutions for handling missing values are to omit the items or to assign the lowest normalized score to these attributes. However, “downgrading” or omitting an item because of missing values might not be acceptable for certain tasks. A well-designed visualization technique should include methods to sensibly deal with missing values.

**R VIII: Interactive refinement and visual feedback** The ranking visualization should enable users to dynamically add and remove attributes, modify attribute combinations, and change the weights and mappings of attributes. To enable users to judge the effect of modifications, it is critical that such changes are immediately reflected

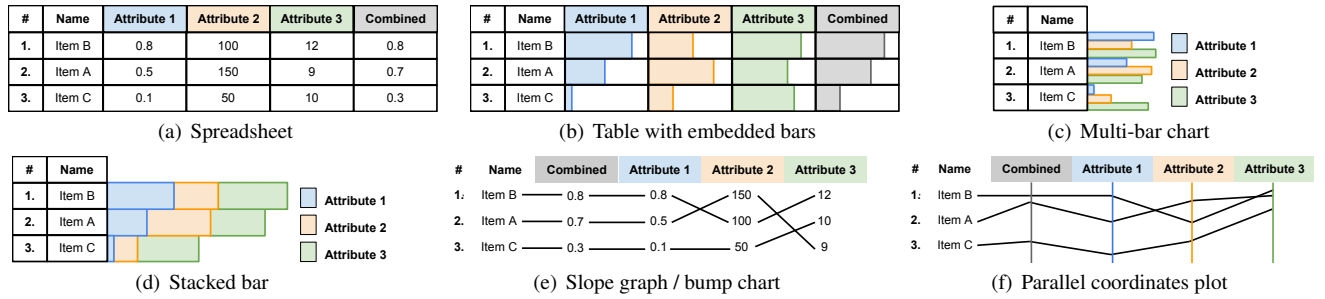


Fig. 2. Illustration of different ranking visualization techniques.

in the visualization. However, as these changes can have profound influences on the ranking, it is essential that the visualization helps users to keep track of the changes.

**R IX: Rank-driven attribute optimization** Optimizing the ranking of an item is an important task. Instead of analyzing how the ranking changes upon modifications of the attribute values or the weights, it should be possible, for example, to optimize the settings (i.e., values and/or weights) to find the best possible ranking of a particular item. Identifying the sensitivity of attributes, i.e., how they are influencing the ranking, for example, for finding the minimum attribute value change needed to gain ranks, is another rank optimization example.

**R X: Compare multiple rankings** An interactive ranking visualization that fulfills R I - R IX is a powerful tool addressing many different tasks. However, in some situations users are interested in putting multiple rankings into context with each other. An example is the comparison of competing university ranking methodologies. Observing changes over time, for instance, investigating university rankings over the last 10 years, is another example that requires the comparison of multiple ranking results.

### 3 RELATED WORK

Due to the ubiquitous presence of rankings and their broad applicability, a wide variety of visualization techniques have been developed for, or have been applied to show, ranked data. Based on the requirements introduced in the previous section, we discuss the design space of visual encodings suitable for ranking visualization, as outlined in Figure 2, and as some specific ranking visualization techniques.

#### 3.1 Spreadsheets

The most basic way to present a set of ordered items is a ranked list showing the rank together with a label identifying the item. While simple ranked lists allow users to see the rank of the item (R I), they do not convey any information about what led to the rank – which violates R II. It is trivial to extend a ranked list by multiple columns resulting in a table or spreadsheet addressing the multi-attribute requirement (R III), as shown in Figure 2(a). A detailed discussion of the design of spreadsheets was published by Few [7]. Established general purpose tools such as *Microsoft Excel* are feature-rich and well known by many users. These tools provide scripting interfaces that can help to address requirements R I - R VII. While scripting provides a great deal of flexibility, it is typically only mastered by advanced users. The major drawback of spreadsheets, however, is that they lack interactive visualizations of the tabular data (R VIII). Also, spreadsheets typically lack the ability to compare multiple lists intuitively (see requirement R X). A comparison of lists can only be achieved by linking several spreadsheets in a multiple coordinated views fashion [25], which is not supported by most spreadsheet applications. In such a setup, however, answering questions related to the essential requirements to encode the rank (R I) and to encode the cause of the rank (R II) is tedious and time-consuming, especially as the number of ranked items increases.

Reading numerical values in a spreadsheet and comprehending the data, however, is a cognitively demanding, error-prone, and tedious task. It is therefore more effective to communicate trends and relationships by encoding the values in a graphical representation using visual

variables such as position, length, and color, etc. We discuss below how visual variables can be used to create visual representations that can cope with ranking data. In line with Ward et al. [35], we divide the related work into techniques that are point-based, region-based, or line-based.

#### 3.2 Point-Based Techniques

Using position as a visual variable is considered to be the most effective way of encoding quantitative data [16]. Simple *scatterplots* can be used to compare two rankings (see R X). A scatterplot, however, can focus either only on communicating the rank itself (R I), by mapping the rank to interval scales, or on the cause of the rank (R II), by encoding the attribute value pairs in the position of the dots. While we can overcome the limitation of only comparing two rankings by using a *scatterplot matrix (SPLOM)*, neither scatterplots nor SPLOMs can deal with R III, the multi-attribute requirement, which makes them an inefficient solution for complex rankings.

#### 3.3 Region-Based Techniques

According to various established sources [16, 5, 2], length is another very effective visual variable for encoding quantitative data. For representing ranked lists, simple bar charts, for instance, can show the value of multiple items for a single attribute. To make use of the pre-tentive processing capabilities in interpreting relative changes in the length of bars (i.e., height), they are usually aligned to a common axis. Aligning the bars also redundantly uses position in addition to length. In cases where ranks are determined based on a single attribute, using bars to encode the attribute values is an effective way to communicate the cause of the rank, satisfying R II.

Bars can be used to encode multiple attributes (R III) in three different ways: by aligning bars for every attribute to a separate baseline, as shown in Figure 2(b), by showing multiple bars per item (one for each attribute) on the same baseline, see Figure 2(c), or by using stacked bar charts, see Figure 2(d).

An early implementation of the first approach is the *table lens* technique [23], which embeds bars within a spreadsheet. Besides being one of the first focus+context techniques, it allows users to set multiple focus areas and also supports categorical values. As in a spreadsheet, users can sort the table items according to an arbitrary column. John et al. [13] proposed an extension of the original table lens technique that adds two-tone pseudo coloring as well as hybrid clustering of items to reduce strong oscillation effects in the representation. The major benefit of using bar charts with multiple baselines is that it is easy to compare one attribute across items. In contrast, by switching to a layout that draws multiple bars per item side by side on the same baseline, the comparison across attributes for a single item is better supported, but comparing the bars across items becomes more difficult.

Stacked bar charts are appropriate when the purpose of the visualization is to present the totals (sum) of multiple item attributes, while also providing a rough overview of how the sum is composed [7]. Stacked bar charts are usually aligned to the baseline of the summary bar. However, shifting the baseline to any other attribute results in *diverging stacked bar charts*, see Figure 4(b), which were discussed by Willard Brinton as early as 1939 [3]. Changing the baseline makes it difficult to compare the sum between different stacked bars, but easier to relate the values of the aligned attribute. Diverging stacked bar

charts are also known to be well suited to visualizing survey data that is based on rating scales, such as the Likert scale [24].

Compared to spreadsheets, bar-based techniques scale much better to a large number of items (R VI). While the minimum height of rows in spreadsheets is practically determined by the smallest font that a user can read, bars can be represented by a single-pixel line. It is even possible to go below the single pixel limit by using overplotting. This aggregation of multiple values to a single pixel introduces visual uncertainty [11]. While details such as outliers will be omitted, major trends will remain visible. Combined with additional measures such as a fish-eye feature, bar-based approaches are an effective technique for dealing with a large number of items.

### 3.4 Line-Based Techniques

Line-based techniques are also a widely used approach to visualizing rankings. In principle, lines can be used to connect the value of items across multiple attributes (R III) or to compare multiple rankings (R X). Although a wide array of line-based techniques exist [35], only a few of them are able to also encode the rank of items (R I).

The first technique relevant in the context of ranking visualization are *slope graphs* [33, p.156]. According to Tufte, slope graphs allow users to track changes in the order of data items over time. The item values for every time point (attribute) are mapped onto an ordered numerical or interval scale. The scales are shown side by side, in an axis-like style without drawing them explicitly, and identical items are connected across time points. By judging differences in the slope, users are able to identify changes between the time points. Lines with slopes that are different to the others stand out. Note that slope graphs always use the same scale for all attributes shown. This makes it possible not only to interpret slope changes between two different time points, but also to relate changes across multiple or all time points. Although Tufte used the term slope graph only for visualizing time-dependent data, the technique can be applied equally to arbitrary multi-attribute data that uses the same scale.

Slope graphs that map ordered data to a scale using a unique spacing between items are referred to as *bump charts* [34, p.110]. Bump charts are specialized slope graphs that enable users to compare multiple rankings, fulfilling R X. The example in Figure 2(e) shows a bump chart where each column is sorted individually and the lines connect to the respective rank of each item. *Tableau*<sup>1</sup>, for example, uses bump charts to trace ranks over time. However, while bump charts show the rank of items (R I), they do not encode the cause of the rank (R II), which means the actual values of the attributes that define the ranking are lost.

A line-based multi-attribute visualization that mixes scales, semantics, and data types across the encoded attributes is a *parallel coordinates plot* [12], as shown in Figure 2(f). In contrast to a slope graph, a parallel coordinates plot uses the actual attribute values mapped to the axis instead of the rank of the attribute. While this is more general than slope graphs, users lose the ability to relate the slopes across different attributes. A thorough discussion of the differences between the aforementioned line-based techniques was published by Park [20, 21].

Fernstad et al. [6], for instance, propose a parallel coordinates plot that shows one axis for each attribute with an extra axis showing the overall rank. While this addresses R I - R III, adding the possibility to compare multiple rankings (R X) is difficult. In theory, we could create parallel coordinates showing multiple axes that map the rank and add further axes to show the attributes that influence each of the rankings. However, in this case it would be difficult to make it clear to the user which rank axis belongs to which set of attribute axes.

### 3.5 Ranking Visualization Techniques

Having discussed the design space of visualizing rankings, we now give some examples of specific ranking visualization techniques. The rank-by-feature approach by Seo and Shneiderman [29] uses ordered bars to present a ranked list together with a score. The ranking suggests potentially interesting features between two variables. The

scores are calculated according to criteria such as correlation and uniformity. In addition, the authors propose the rank-by-feature prism, a heat map that shows the feature scores between multiple variables. However, as the scores are based only on a single attribute, the rank-by-feature system does not address multi-attribute rankings and is therefore only tangentially relevant to our technique. The *RankExplorer* system by Shi et al. [30] uses stacked graphs [4] with augmented color bars and glyphs to compare rankings over time. While the system effectively addresses the ranking comparison requirement (R X), it can only incorporate the information about the cause of the rank based on multiple attributes (R II and R III) by showing details on demand in a coordinated view fashion. Sawant and Healey [27] visualize multi-dimensional query results in a space-filling spiral. Items from the query result are ordered by a single attribute and placed on a spiral. A glyph representation is used for encoding the attributes of the items. By using animation, the visualization can morph between different query results, highlighting the similarities and differences. Here, the ranking is based only on a single attribute. Recent work by Behrisch et al. [1] addresses the comparison of multiple rankings using a small-multiple approach in combination with a radial node-link representation; however, it is not designed to encode the cause of the rankings. The work by Kidwell et al. [14] focuses on the comparison of a large set of incomplete and partial rankings, for example, user-created movie rankings. The similarity of rankings is calculated and visualized using multi-dimensional scaling and heat maps. While their approach gives a good overview of similarities between a large number of rankings with many items, an in-depth comparison of rankings is not possible. As trees can be ranked, tree comparison techniques, such as *Tree Juxtaposer* [18], can also be used to compare rankings. However, encoding multiple attributes using trees is problematic.

## 4 MULTI-ATTRIBUTE RANKING VISUALIZATION TECHNIQUE

LineUp is an interactive technique designed to create, visualize, and explore rankings of items based on a set of heterogeneous attributes. The visualization uses bar charts in various configurations. By default, we use stacked bar charts where the length of each bar represents the overall score of a given item. The vertical position of the bar in the bar chart encodes the rank of the item, with the most highly ranked item at the top. The basic design of the LineUp technique, as shown in Figure 3, is introduced in detail in Section 4.1. The components of the stacked bars encode the scores of attributes, which can be weighted individually. Combined scores can be created for sets of attributes using two different operations (see Section 4.2). Either the sum of the attribute scores is computed and visualized as stacked bars – a *serial combination* – or the maximum of the attribute scores in the set is determined and visualized as bars placed next to each other – a *parallel combination*. Such combinations can be nested arbitrarily.

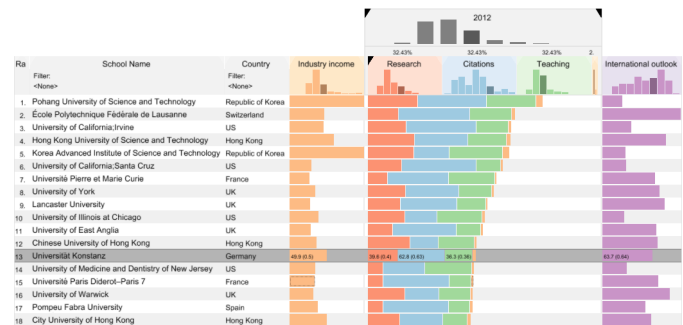


Fig. 3. A simple example demonstrating the basic design of the LineUp technique. The screenshot shows the top-ranked universities from the Times Higher Education Top 100 under 50 datasets (see Section 5 for data source). The first column shows the ranks of the universities, followed by their names and the categorical attribute *Country*. The list is sorted according to the combined attribute column containing four university performance attributes. Two numerical attribute columns which do not influence the ranking are also shown.

<sup>1</sup><http://www.tableausoftware.com>



Furthermore, LineUp can be used to create and compare multiple rankings of the same set of items (see Section 4.4). When comparing rankings, the individual rankings are lined up horizontally, and slope graphs are used to connect the items across rankings, as shown in Figure 1. The angle of the slope between two rankings represents the difference in ranks between two neighboring rankings.

Formally, rankings in LineUp are defined on a set of items  $x_i \in X = \{x_1, \dots, x_m\}$  and a heterogeneous set of attribute vectors  $\mathbf{a}_j \in A = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$  so that each item  $x_i$  is assigned a set of attribute values  $A_i = \{a_{i1}, \dots, a_{in}\}$ . Since the attributes are heterogeneous, for instance, numerical on different scales or categorical, the user first needs to normalize the attribute vectors  $\mathbf{a}_j$  by mapping them onto a common scale. To achieve this, the user provides a mapping function  $m_j(\mathbf{a}_j) = \mathbf{a}'_j \in A'$  with  $m_j : a_{ji} \rightarrow [m_{j\min}, m_{j\max}]$  for each attribute with  $0 \leq m_{j\min} \leq m_{j\max} \leq 1$ . The values of  $m_{j\min}$  and  $m_{j\max}$  can be defined by the user. Throughout the paper we refer to mapped attribute values  $a'_{ji}$  as *attribute scores* and to  $A'$  as the *mapped attributes*.

Additionally, the user may specify filters  $f_{j\min}$  and  $f_{j\max}$  on the original attribute values to remove items with attribute values  $a_{ji}$  outside the filter range  $[f_{j\min}, f_{j\max}]$  from the ranking. The visualizations and interactions provided for data mapping and filtering are described in detail in Section 4.6.

To assign an item score  $s_i \in S \subset \mathbb{R}_0^+$  to each item  $x_i$ , the user interactively defines a scoring function  $s$  over the mapped attributes in  $A'$  through the LineUp user interface. The user selects a list  $B = (\mathbf{a}'_q)$  of one or more attributes from  $A'$  with  $1 \leq q \leq n$ , where an attribute may be added more than once by cloning the attribute. The item score  $s_B(x_i)$  over a list of mapped attributes  $B$  from  $A'$  is defined as

$$s_B(x_i) = \sum_{\mathbf{a}'_q \in B} w_{\mathbf{a}'_q} a'_{qi} \quad | \quad 0 \leq w_{\mathbf{a}'_q} \leq 1 \wedge \sum w_{\mathbf{a}'_q} = 1,$$

where  $w_{\mathbf{a}'_q}$  are weights assigned to each instance of a mapped attribute  $\mathbf{a}'$ . Since the user may divide  $B$  into multiple lists  $B_l$  and combine, weight, and nest them arbitrarily, as discussed in Section 4.2, the final item score  $s_i$  is defined recursively over nested lists of mapped attributes as

$$s_i = s_B(x_i) = \begin{cases} \sum_{\mathbf{a}'_q \in B} w_{\mathbf{a}'_q} a'_{qi} & | \quad 0 \leq w_{\mathbf{a}'_q} \leq 1 \wedge \sum w_{\mathbf{a}'_q} = 1 \\ \sum_{B_l} w_{B_l} s_{B_l}(x_i) & | \quad 0 \leq w_{B_l} \leq 1 \wedge \sum w_{B_l} = 1 \\ \max_{B_l} s_{B_l}(x_i). & \end{cases}$$

The operators  $\sum$  and  $\max$  represent the sum (*serial combination*) and the maximum (*parallel combination*) of the item scores over a list of attribute scores, respectively. Users can interactively change the weights  $w \in \mathbb{R}_0^+$  for each list of attributes by changing the width of the corresponding column header in LineUp.

LineUp determines the rank  $r_i \in \mathbb{N}^+$  of an item  $x_i$  based on its item score  $s_i$  (which is equivalent to  $a'_{ji}$  for cases in which ranks are based on a single attribute  $\mathbf{a}_j$ ), with  $\max(S) = 1$  so that  $r_j - r_i = d \in \mathbb{N}^+$  if  $s_j < s_i$ , and there are exactly  $d - 1$  other scores  $s_k \in S$  with  $s_j < s_k \leq s_i$ . Ties are allowed since two or more items may have the same score. To resolve ties, the scoring method described above can be applied recursively to a tied set of items, for instance, using different attributes.

#### 4.1 Basic Design and Interaction

LineUp is a multi-column representation where users can arbitrarily combine various types of columns into a single table visualization, as shown in Figure 3. The following column types are supported:

- **Rank columns** showing the ranks of items.
- **Textual attribute columns** for labels of items or nominal attributes. Text columns provide contextual information on the basis of which users can also search or filter items.
- **Categorical attribute columns** can be used in a similar fashion as textual attribute columns.
- **Numerical attribute columns** encoding numerical attribute scores as bars. In addition to the name of the attribute, the header

can show the distribution of attribute scores on demand. When the user selects a particular item, the corresponding bin in the histogram will be highlighted.

- **Combined attribute columns** representing combinations of sets of numerical attributes. The process and visual encoding of combined columns is explained in Section 4.2.

In its simplest form, LineUp presents each attribute as a separate column where numerical columns use bars to represent their values. The ranking can be driven by any column, using sorting based on scores or on lexicographic order. Figure 3 also shows a combined attribute column labeled “2012” with four nested attributes. The decreasing lengths of the stacked bar charts indicate that this combined column drives the ranking, which is also shown using black corners at the top of its column header. In this example, the item labeled “Universität Konstanz” is selected. For selected items we show the original and the normalized attribute value (score) inside the bars if sufficient space is available. To simplify tracking of items across columns, rows are given alternating background colors.

#### 4.2 Combining Attributes

A fundamental feature of our ranking visualization is the ability to flexibly combine multiple attributes, as described in requirement R III. LineUp supports the combination of attributes either in series or in parallel, as formally introduced earlier in this section. Both types of combinations are created by dragging the header of a (combined) attribute column onto the header of another (combined) column. Removing an attribute from a combined column is possible by using drag-and-drop or by triggering an explode operation that splits the combined column into its individual parts.

##### 4.2.1 Serial Combination

In a serial combination the combined score is computed by a weighted sum of its individual attributes scores. The column header of such a combined column contains the histograms of all individual attributes that contribute to the combined score as well as a histogram showing the distribution of the combination result. While the combined scores are represented as stacked bars, the weight of an attribute is directly reflected by the width of its column header and histogram. Altering weights can be realized either by changing the width of a column using drag-and-drop or by double-clicking on the percentages shown above the histograms to specify the exact distribution of weights. While the former approach is particularly valuable for experimenting with different weights because the ranking will be updated interactively, the latter is useful to reproduce an exactly specified distribution, as demonstrated in the university ranking use case presented in Section 6.2.

Stacked bars allow users to perceive both the combined attribute score and the contribution of individual attribute scores to it. However, stacked bars complicate the comparison of individual attribute scores across multiple items, as only the first one is aligned to the baseline. Therefore, LineUp realizes four alignment strategies, which are shown

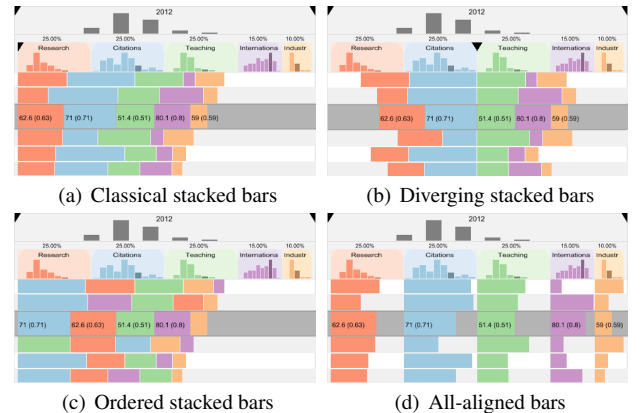


Fig. 4. Strategies for aligning serial combinations of attribute scores.

in Figure 4. Besides classical stacked bars, diverging stacked bars, where the baseline can be set to an arbitrary attribute, are provided. The third strategy sorts the bars of each row in decreasing order, highlighting the attributes that contribute the most to the score of an item. The last strategy aligns every attribute by its own baseline, resulting in a regular table with embedded bars. These strategies can be toggled dynamically and use animated transitions for state changes.

#### 4.2.2 Parallel Combination

In contrast to the serial combination that computes a combined score by adding up multiple weighted attribute values, the parallel combination is defined as the maximum of a set of attribute scores. Due to the limited vertical space, only the attribute with the largest score is shown as the bar for a given item. The attribute scores that do not contribute to the rank of the item are only shown when the item is selected. The corresponding bars are drawn on top of each other above the largest bar, as illustrated in Figure 5. In order to avoid small values overlapping bigger ones, the bars are sorted according to length.

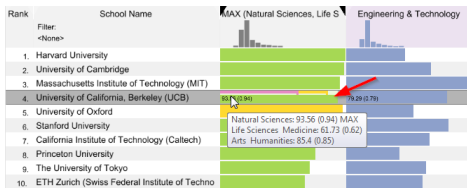


Fig. 5. Parallel combination of three attributes. Only the bar for the attribute with the largest score is shown for unselected items.

#### 4.3 Rank Change Encoding

One of the major strengths of the proposed approach is that users receive immediate feedback when they interactively change weights, set filters, or create and refine attribute combinations. LineUp supports users in keeping track of rank changes by using a combination of animated transitions [10] and color coding. Animated transitions are an effective way to encode a small number of changes; however, when the number of changing items is very large and the trajectories of the animation paths cross each other, it is hard to follow the changes. Therefore, we additionally use color to encode rank changes, as demonstrated in Figure 6 (right). Items that move up in the ranking are visualized in green, whereas those that move down are shown in red. The rank change encoding is shown only for a limited time after a change has been triggered and is then faded out. The animation time and color intensity depend on the absolute rank change, which means the more ranks an item moves up or down, the longer and the more intense the highlighting. By providing interactive rank change and refinement capabilities combined with immediate visual feedback, LineUp is able to address requirement R VIII.

#### 4.4 Comparison of Rankings

Encoding rank changes within a ranking visually is an essential feature to help users track individual items. However, animated changes and color coding are of limited assistance when analyzing differences between rankings. In order to address this problem, a more persistent visual representation is needed that allows users to evaluate the changes in detail. In fact, we want to enable users to compare different rankings of the same set of items, as formulated in R X. However, comparing rankings is not only important to support users in answering “What if?” questions when underlying attribute values change, but are also highly relevant for analyzing how multiple different attribute combinations or weight configurations compare to each other. An example, presented in the use case of university rankings (see Section 6.2), is the comparison of rankings over time.

We realize the comparison of rankings by lining up multiple rankings horizontally next to each other, each having its own item order, and connect identical items across the rankings with lines as in a slope graph. An example with two rankings and one with multiple rankings are given in Figures 6 and 1, respectively. The slope graph acts as a

rank separator. This means that every attribute column that is separated by a slope graph has its own ranking. Also, changes in weights, mappings, filters, or attribute combinations only influence the order of items between the rank separators.

Due to limited vertical space, it can happen that items connected by lines in the slope graph end up outside the visible area of the ranking. To reduce clutter, we replace connection lines that have an invisible target with arrows pointing in the direction of the invisible item. In addition to the slope graphs and arrows, we also show the absolute difference in ranks.

In order to allow users to track changes of actions they trigger, they can create a *snapshot* of individual or combined attribute columns. Snapshots are fully functional clones of columns that are inserted as the rightmost column of the table. A new snapshot is assigned its own rank column, and a slope graph is used to connect it to the column immediately to its left. Combined with the visual encodings for rank changes, snapshots are an effective way to answer “What if?” questions, for instance, “What happens if I change the weights in my ranking?” In Figure 6, for example, the user creates a snapshot which duplicates the current attribute combination. The user can then modify the weights of the attributes in the snapshot and compare the changes to the original attribute combination that holds the original ranking.

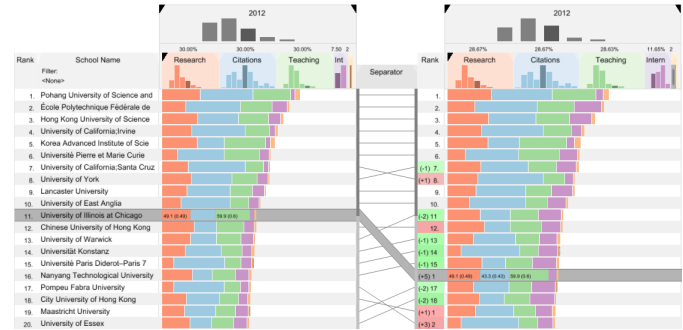


Fig. 6. Comparison between rankings. The concept of rank separators between score attribute columns makes it possible to use different orders on both sides and to relate them to each other by following the lines connecting them. Changes in either one of the rankings are immediately reflected in the visualization. This is visually supported with animations and also indicated by color changes of the rank label: green = item moved up, red = item moved down. The more intense the color, the more ranks were gained or lost.

#### 4.5 Scalability

A powerful ranking visualization needs to scale in the number of attributes and the number of items it can handle effectively, as formulated in the scalability requirement (R VI). Here we discuss our approaches for both.

##### Many Attributes

In order to handle dozens of attributes, in addition to providing scroll-bars, we allow users to reduce the width of attribute columns by collapsing or compressing them on demand. *Collapsing* a column reduces its width to only a few pixels. As bars cannot effectively encode data in such a small space, we switch the visualization from a bar chart to a grayscale heat map representation (darker values indicate higher scores). Examples of collapsed columns are the three rightmost in Figure 1.

While collapsing can be applied to any column, *compressing* is applicable only to serial combinations of attribute columns. To save space, users can change the level of detail for the combined column to replace the stacked bar showing all individual scores with a single summary bar. An example of this is shown in the “2011” and “2010” columns in Figure 1.

To further increase scalability with respect to the number of attributes, we provide a *memo pad* [31], as shown at the bottom of Figure 1. The memo pad is an additional area for storing and managing

attributes that are currently not of immediate interest but might become relevant later in the analysis. It can hold any kind of column that a user removes from the table, including full snapshots. Attributes can be removed completely from the dataset by dragging them to the trash can icon of the memo pad.

We assign colors to each attribute based on a carefully chosen qualitative color scheme and repeat colors when we exceed seven [9]. However, this approach becomes increasingly problematic with a growing number of attributes. One option to address this problem is to use the same color for semantically related attributes, as illustrated in the use case in Section 6. For instance, when the goal is to rank food products, we use the same color for all vitamin-related attributes. However, whether this approach is useful depends on the specific scenario and the task. Therefore, the mapping between attributes and colors can be refined by the users.

## Many Items

In order to make our technique useful in real-world scenarios, we need to cope with thousands of items. We use two strategies to achieve this: filtering and optimizing the visual representation. While filtering is straightforward, we also allow users to choose between uniform line spacing and a fish-eye selection mode [26]. Most users will be more familiar with uniform line spacing; which is, however, limited to showing only up to about 100 items at a time on a typical display. The fish-eye, in contrast, scales much better. The disadvantage is that changes in slope for comparison are less reliable due to distortion.

## 4.6 Data Mapping

Data mapping is the process of transforming numerical or ordered categorical attribute values to a normalized range between 0 and 1 (R V) that can then be used to determine the length of the attribute score bars (1 corresponds to a full bar). By default, LineUp assumes a linear mapping and infers the bounds from the dataset so that no user interaction is required. To create more complex mappings, LineUp provides three approaches: choosing from a set of essential mapping functions (e.g., logarithmic or inversion), a visual data mapping editor that enables users to interactively create mappings, and a scripting interface to create sophisticated mappings. As non-linear mappings, inversions, etc. can have a profound impact on the ranking, we use a hatching pattern for all non-linear bars to communicate this.

In order to let users create mappings with different levels of complexity, the visual data mapping editor provides two options to interactively define them, as illustrated in Figure 7. All interactive changes in the mapping functions are immediately reflected in the visualization.

In the *parallel mapping editor* we show the histogram of the normalized attribute values above the original histogram of the raw data. Connection lines between the two histograms, drawn for every item, help the user to assess which attribute value maps to which score in the normalized range. By default, we apply a linear mapping that results in parallel lines between the histograms, as shown in Figure 7(a). By dragging the minimum or maximum value markers in the histograms, users can filter items above or below a certain threshold, as shown in Figure 7(b). To flexibly create arbitrary mappings, mapping markers can be added, moved, and removed. Figure 7(c) shows a mapping scenario where the attribute values range from -1 to 1 but the scores are based on the absolute values. In addition, items with an attribute score of less than 0.2 are filtered.

The *orthogonal mapping editor* is an alternative view that uses a horizontal histogram of raw attribute values and a perpendicular vertical histogram of the normalized scores to visualize the mapping. This layout has the advantage that it can be interpreted just like a regular mapping function. Users can flexibly add, move, and remove support points to define the shape of the mapping function. We use linear interpolation to calculate the mappings between the user-defined support points. Figures 7(d) to 7(f) show the same examples that were used above to illustrate the parallel layout. Users can hover over any part of the mapping function to see the raw and normalized value pairs.

The visual data mapping editor also shows a formal representation of the mapping function that is always in sync with the interac-

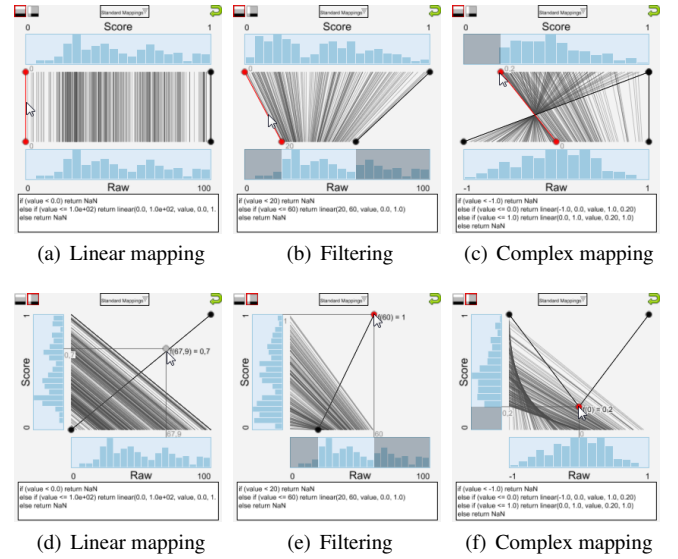


Fig. 7. Visual mapping editor for mapping attribute values to normalized scores. The parallel mapping editor used in (a)-(c) shows the distribution of values as well as the normalized scores as histograms on top of each other. Connection lines between the histograms make it easy to interpret the mapping. In (d)-(f) the layout of the mapping editor is changed to an orthogonal arrangement that resembles an actual mapping function. We show three mapping examples defined using the two different mapping editor layouts. (a) and (d) show the default case, where raw values are linearly mapped. In (b) and (e) raw values below 20 and above 60 are filtered. The remaining value range is spread between 0 and 1. In (c) and (f) the mapping is driven by three markers, to produce a mapping that emphasizes high and low values and at the same time filters low scores.

tive mapping editor. Clicking this representation opens the *JavaScript function editor* that can be used to define complex mapping functions, such as polynomial and logarithmic functions, which cannot easily be defined in the visual editors.

## 4.7 Missing Values

As real-world datasets are seldom complete, we need to deal with missing attribute values and encode them in the visualization. The only way to obtain a meaningful combined score based on multiple attributes where at least one has a missing value is to infer them. However, there is no general solution for inferring missing values that works for every situation. We currently apply standard methods such as calculating the mean and median; however, the integration of more complex algorithms is conceivable [28]. Besides the computation of missing value replacements, their visualization is crucial. As inference of missing values introduces artificial data, it is important to make the user aware of this fact. In LineUp we encode inferred data with a dashed border inside the bars.

## 5 IMPLEMENTATION

The LineUp visualization technique is part of *Caleydo*, an open-source data visualization framework [15]. Caleydo is implemented in Java and uses OpenGL/JOGL for rendering. A demo version of LineUp is freely available at <http://lineup.caleydo.org> for Windows, Linux, and Mac OS X.

In the examples discussed throughout the paper we used three datasets: the *Times Higher Education 100 Under 50 University Ranking* [32], the *QS World University Ranking* [22], and a subset of the *USDA National Nutrient Database* [19].

## 6 USE CASES

We demonstrate the technique in two use cases: the nutrition content of food products and ranking of universities.



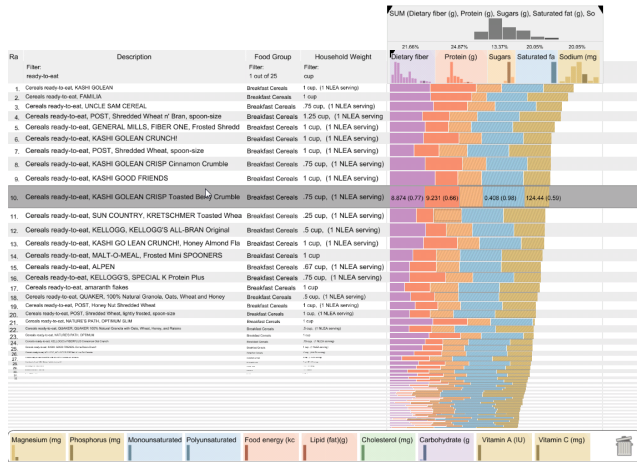


Fig. 8. Example of a customized food nutrition ranking to identify healthier choices of breakfast cereals.

### 6.1 Food Nutrition Data

The first use case demonstrates how users can interactively create complex attribute combinations using the LineUp technique. Let us assume that John receives bad news from his doctor during his annual physical exam. In addition to his pre-existing high blood pressure, his cholesterol and blood sugar levels are elevated. The doctor advises John to increase his physical activity and improve his diet. John decides to systematically review his eating and drinking habits and begins with evaluating his usual breakfast. He loads a subset of the comprehensive food nutrition dataset from the *USDA National Nutrient Database* containing 19 nutrition facts (attributes) for each of about 8,200 food products (items) into LineUp. After loading the dataset, every attribute ends up in a separate column. As a first task, he filters the list to only include products in the category *breakfast cereals*. When he looks up his favorite breakfast cereals, he is shocked to find that they contain very high amounts of sugar and saturated fat. In order to find healthier choices, John searches for products that are high in *dietary fiber* and *protein* but low in *sugar*, *saturated fat*, and *sodium*. To rank the products according to these criteria, he creates a new serial attribute combination that assigns all attributes equal weight. In addition, as he is interested only in products that have “ready-to-eat” in their description, he applies another filter. Since he wants low values of *sugar*, *saturated fat*, and *sodium* to receive a high score while high values should receive a low score, he uses the parallel mapping editor to invert the mapping function for these attributes. After looking at the top 20 items in the ranking, he realizes that none of the products matches his taste. He starts to slowly decrease the weight assigned to the *sugar* attribute, which means he reduces its impact on the overall ranking, and tracks the changes by observing the rank change color encoding and animations. He also uses the fish-eye to handle the large number of products when browsing the list until he finally finds his new breakfast cereal. Figure 8 shows the result of his analysis.

### 6.2 University Ranking

In the second use case we demonstrate how Jane, a prospective undergraduate student, utilizes LineUp to find the best institutions to apply to. As a basis for the selection process, Jane chooses the well established *QS World University Ranking*. She starts by loading the annual rankings published from 2007 through 2012. As Jane does not want to leave the US, she adds a filter to the categorical *country* attribute to remove universities outside the US, and reviews the rankings for US institutions. By looking at the bar charts, she is able to see what factors contribute to the ranks and how they are weighted. The *QS World University Ranking* is based on six attributes: *academic reputation* (weighted 40%), *employer reputation* (10%), *faculty/student ratio* (20%), *citations* (20%), *international faculty ratio* (5%), and *international student ratio* (5%). Additionally, the authors publish performance data on five broad subject areas, such as *arts & humanities* and *life sciences*, which, however, do not influence the ranking. While

university rankings try to capture the overall standing of institutions, Jane is a prospective undergraduate student and does not care much about research and citations but rather wants to emphasize teaching. Additionally, she wants to go to a university that has a renowned arts faculty and a strong international orientation, as documented by many exchange students and staff from abroad. To obtain the ranking that reflects her preferences, Jane wants to combine these attributes and adjust the weights accordingly. In order not to lose the original ranking, she takes a snapshot of the original attribute combination. In the new snapshot she removes *employer reputation* and *citations* from the combined score and adds *arts & humanities* to the weighted attributes. Next, she adjusts the weights by interactively resizing the width of some of the columns. The immediate feedback through the animated transitions and the changed color coding help her to get a feeling of how sensitive the ranking is to changes in the attribute weights. She then refines the weights according to her preferences. The slope graph between the original ranking (stored in the snapshot) and the ranking based on the copied attribute combination clearly indicates that there are significant differences between the rankings. Jane realizes that she actually wants to find a university that not only matches her criteria but also has a high rank in the *QS World University Ranking*. To do this, she nests the original combined *QS World University Ranking* score, stored in the snapshot, within the customized combination. The result of this nested combination is shown in Figure 1. As a final step, she wants to make sure to only apply to universities that do not show a downward trend over the last 3 years. By following the slope graphs over time, she then picks the five universities that best fit her preferences and to which she will apply.

## 7 EVALUATION

LineUp is tailored to allow novice users to solve tasks related to the creation, analysis, and comparison of multiple rankings. As discussed in the Related Work section, there is no single technique or software that fulfills all requirements described in Section 2. However, it is indeed possible to successfully solve most, if not all, ranking-related tasks with off-the-shelf tools such as Microsoft Excel or Tableau Desktop. To confirm this, we ran a pre-study where we asked an expert Excel user and an expert Tableau user to complete the ranking tasks described in Section 7.2. This informal test confirmed that solving these tasks with generic analysis and visualization tools is possible but tedious and time-consuming.

Also, tools such as Tableau and Excel require considerable scripting skills and experience to solve complex ranking tasks. In contrast, our technique aims to empower novice users to achieve the same results with very little training. A formal comparative study with a between-subject design where experts use Excel or Tableau and novices use LineUp would be unable to confirm this, as it would be impossible to tell whether the observed effects were caused by the difference in the subjects’ backgrounds or between the tools. Also, a within-subject design that uses either experts or novices would be highly problematic. The first option would be to use Excel and Tableau experts and compare their performance using each tool. However, the experts would not only be biased because of their previous training in the respective tool, but, even more importantly, they are not the target audience of our technique. The second option, a within-subject design that tests how novices would complete the tasks using the different tools is not possible either because of the level of experience and knowledge necessary to perform the tasks in the other two tools. Consequently, we believe that it is more meaningful to show the effectiveness of the LineUp technique in a qualitative study.

### 7.1 Study Design

For the qualitative study we recruited eight participants (6 male, 2 female) between 26 and 34 years old. They are all researchers or students with a background in computer science, bioinformatics, or public health. Half of them indicated that they have some experience with visualization, one of them had considerable experience. In a pilot study with an additional participant, we ensured that the overall process runs smoothly and that the tasks are easy to understand. Prior



to the actual study, we checked the participants for color blindness. We then introduced the LineUp technique and the study datasets. After the introduction, the participants had the opportunity to familiarize themselves with the software (using a different dataset than those used during the actual study) and to ask questions concerning the concept and interactions. Overall, the introduction and warm-up phase took about 25 minutes per subject. We advised the participants to “think aloud” during the study. In addition to measuring task completion time for the answers, we took notes on the participants’ approaches to the tasks and what problems they encountered. After each task, participants had to fill out the standardized NASA-TLX questionnaire for workload assessment [8]. After they had finished all tasks, we gave the subjects a questionnaire with 23 questions, which evaluated the tool on a 7-point Likert scale. It included task-specific and general questions about the LineUp technique and questions making comparisons with Excel and Tableau (which they were asked to answer only if they were sufficiently experienced in using one of the tools). Additionally, we concluded every session by asking open questions to collect detailed feedback and suggestions for improvements.

## 7.2 Results and Discussion

We designed 12 tasks that the participants had to perform using LineUp. The tasks covered all important aspects of the technique concerning the creation, analysis, and comparison of rankings. Detailed task descriptions, study results, and questionnaires can be found in the supplementary material of this paper.

Although we aimed to formulate the tasks as atomically as possible, they intentionally had different levels of complexity. This is also apparent in the results of the NASA-TLX workload assessment questionnaires. We measured task completion times (see supplementary material) to approximate the potential for improvements in the interface and to identify disruptions in the workflow. In general, participants were able to fulfill the tasks successfully and in a short period of time. Two outliers, where users needed more time, were noticeable: Task 7, in which participants were asked to filter data and evaluate the change, and Task 12, a comprehensive example with subtasks. As Task 12 comprised multiple aspects, we expected it to take longer. The long task completion times for Task 7, on the other hand, were unexpected. While most users solved the task in reasonable time, some needed longer because they tried to evaluate the changes in-place in the table, while we had assumed that they would use the snapshot feature, which would make the task significantly easier.

In the questionnaire, which is based on a 7-point Likert scale ranging from strongly agree (1) to strongly disagree (7), the majority of participants stated that the technique is visually pleasing (mean 1.6), potentially helpful for many different application scenarios (1.3), and generally easy to understand (2.4).

In the questionnaire, participants were asked about their experience level in Excel and Tableau. Although the participants used only LineUp in the study, we wanted to know (if they had any experience in Excel or Tableau) whether (a) the task could be solved in one of the other tools and (b) whether this could have yielded more insight. The average level of experience in Excel was 3.8 on a 7-point scale ranging from novice (1) to expert (7). None of the participants were familiar with Tableau. Participants rated the expected difficulty for doing the same tasks with Excel 4.4 on average. Most of them were convinced that LineUp would save time (1.6) and allow them to gather more insights (1.6).

In addition to evaluating the general effectiveness of our solution, we also wanted to find out if users are able to understand the mapping editor and which layout they prefer. The study showed that the participants found the parallel editor easy to understand (1.8) but that they were skeptical about the orthogonal layout (4.4).

In the open-ended feedback session, participants particularly valued the interactive approach combined with immediate feedback. Some of them stated that using drag-and-drop to create complex rankings is much more intuitive than typing formulas in Excel. Also, snapshots for comparing rankings were positively mentioned several times. In addition to the positive aspects, participants provided suggestions for im-

provements and reported minor complications in their workflow. They mentioned, for instance, that the button for creating a combined score is hard to find and suggested introducing a mode in which users can hold a modifier key and then select the attributes they want to combine. Also, some participants said that the rank change encoding disappears too fast to keep track of the changes. However, after reviewing the notes taken during the study, it was obvious that the participants who mentioned this did not use the snapshot feature, which provides better support for tracking rank changes than the transient change indicators.

The Tableau and Excel experts from the pre-study were asked to complete the same tasks as the regular participants. As previously mentioned, both were able to perform most of the tasks correctly in the pre-study. However, their completion times suggest that novice users are considerably faster in solving the tasks with LineUp than the experts using Tableau or Excel. We did not formally measure the task completion time in the pre-study, as the goal of the pre-study was not to collect performance data that can be used for comparison, but to get an impression if the tasks are possible in general and how difficult it is to solve them. Simple tasks that required users to filter, search, or sort to a certain attribute had about the same performance in all tools. However, the pre-study revealed that the experts in both tools had problems to solve tasks that involved “What if?” questions. For instance, it was difficult for them to change weights or mappings and evaluate changes, as these tasks benefit significantly from interactive refinement with immediate visual feedback. The Tableau expert even mentioned that “there is only trial-and-error or manual calculation. I do not know an elegant way of doing that”.

## 8 CONCLUSION

In this paper we introduced LineUp, a technique for creating, analyzing, and comparing multi-attribute rankings. Initially, our goal for the technique was to enable domain experts to formulate complex biological questions in an intuitive and visual way. When we realized that the technique can be applied in many scenarios, we chose to generalize it to a domain-independent universal solution for rankings of items.

Our evaluation shows that major strengths of LineUp are the interactive refinement of weights and mappings and the ability to easily track changes. While this is valuable in many cases, it still requires users to actually perform the changes in order to see their result. In the future, we plan to provide means to optimize rankings, for instance, by calculating and communicating how much one or multiple attributes need to be changed to achieve a given rank (see R IX). In addition, we want to investigate how statistical techniques can be used to help the user to effectively deal with a large number of attributes.

The integrated slope graphs in LineUp support the task of comparing multiple ranked lists. However, large differences in the rankings result in many steep slopes that are hard to interpret. One interesting solution would be to rank the list according to the rank delta of the comparison, making it trivial to identify winners and losers between large lists of items. These derived rankings could even be used as new attributes in attribute combinations.

Finally, we validated the *algorithm design* and *encoding/interaction design* aspects of LineUp according to Munzner’s model [17] in a comprehensive evaluation. What remains to be done is to observe actual users applying our tool in real-world analyses and to observe adoption rates. We plan to study this in two scenarios: First, we intend to create a web-based implementation to make the tool available to a general audience for popular tasks. Second, we plan to apply our tool to its original purpose: ranking of genes, clusters, and pathways in analyses of genomic data.

## ACKNOWLEDGMENTS

The authors wish to thank Blake T. Walsh, Günter Öller and the anonymous reviewers for their input. This work was supported in part by the Austrian Research Promotion Agency (840232), the Austrian Science Fund (J 3437-N15), the Air Force Research Laboratory and DARPA grant FA8750-12-C-0300, and the United States National Cancer Institute (U24 CA143867).

## REFERENCES

- [1] 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.
- [2] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. ESRI Press, 2010. First published in French in 1967.
- [3] W. C. Brinton. *Graphic presentation*. Brinton Associates, 1939.
- [4] L. Byron and M. Wattenberg. Stacked graphs - geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.
- [5] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.
- [6] S. J. Fernstad, J. Shaw, and J. Johansson. Quality-based guidance for exploratory dimensionality reduction. *Information Visualization*, 12(1):44–64, 2013.
- [7] S. Few. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press, 2nd edition, 2012.
- [8] S. G. Hart. NASA-Task load index (NASA-TLX); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.
- [9] 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.
- [10] J. Heer and G. G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, 13(6):1240–1247, 2007.
- [11] C. Holzhüter, A. Lex, D. Schmalstieg, H.-J. Schulz, H. Schumann, and M. Streit. Visualizing uncertainty in biological expression data. In *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '12)*, volume 8294, page 82940O. IS&T/SPIE, 2012.
- [12] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, 1985.
- [13] M. John, C. Tominski, and H. Schumann. Visual and analytical extensions for the table lens. In *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '08)*, 2008.
- [14] P. Kidwell, G. Lebanon, and W. S. Cleveland. Visualizing incomplete and partially ranked data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1356–1363, 2008.
- [15] A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg. Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In *Proceeding of the IEEE Symposium on Pacific Visualization (PacificVis '10)*, pages 57–64, 2010.
- [16] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.
- [17] T. Munzner. A nested process model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, 15(6):921–928, 2009.
- [18] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. In *Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '03)*, pages 453–462. ACM Press, 2003.
- [19] Nutrient Data Laboratory. USDA National Nutrient Database for Standard Reference, release 25. <http://www.ars.usda.gov/ba/bhnrc/ndl>, 2013.
- [20] C. Park. Edward Tufte's "Slopegraphs". <http://charliepark.org/slopegraphs/>, 2011.
- [21] C. Park. A slopegraph update. <http://charliepark.org/a-slopegraph-update/>, 2011.
- [22] Quacquarelli Symonds. QS world university ranking. <http://www.iu.qs.com/university-rankings/world-university-rankings/>, 2013.
- [23] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*, pages 318–322. ACM Press, 1994.
- [24] N. B. Robbins and R. M. Heiberger. Plotting Likert and other rating scales. In *Proceedings of the 2011 Joint Statistical Meeting*, 2011.
- [25] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Proceedings of the Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV '07)*, pages 61–71. IEEE Computer Society Press, 2007.
- [26] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '93)*, pages 81–91. ACM Press, 1993.
- [27] A. P. Sawant and C. G. Healey. Visualizing multidimensional query results using animation. In *Electronic Imaging 2008*, page 680904, 2008.
- [28] J. Scheffer. Dealing with missing data. *Research letters in the information and mathematical sciences*, 3(1):153–160, 2002.
- [29] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, 2005.
- [30] C. Shi, W. Cui, S. Liu, P. Xu, W. Chen, and H. Qu. RankExplorer: visualization of ranking changes in large time series data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2669–2678, 2012.
- [31] M. Streit, M. Kalkusch, K. Kashofer, and D. Schmalstieg. Navigation and exploration of interconnected pathways. *Computer Graphics Forum (EuroVis '08)*, 27(3):951–958, 2008.
- [32] Times Higher Education. Times higher education 100 under 50. <http://www.timeshighereducation.co.uk/world-university-rankings/2012/one-hundred-under-fifty>, 2012.
- [33] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2nd edition, 1983.
- [34] E. Tufte. *Envisioning information*. Graphics Press, Cheshire Conn., 5th edition, 1995.
- [35] M. Ward, G. Grinstein, and D. A. Keim. *Interactive Data Visualization: Foundations, Techniques, and Application*. A.K. Peters, 2010.