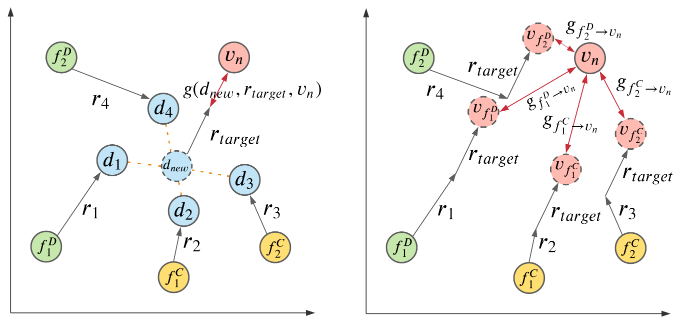
A Detailed Data Features

In this section, we introduce the data features used in our paper in detail. Following VizML [19], we extract 81 data features in total for each data column. All 81 features are grouped into 3 classes, *Types*, *Values* and *Names*. The features under *Types* mainly reflect the data type of the data column, for example, temporal and decimal. The features under *Values* describe the characteristic of values in the data column, for example, the existence of outliers and the standard deviation. The features under *Names* are extracted from the column name and aim to represent the semantic meaning of the column. Among all the features, we have 50 continuous features and 31 categorical features. The detailed lists of continuous and categorical features and the corresponding feature explanations are shown in Table 8 and Table 9, respectively.

B Alternative Inference Method

In Section 4.4, we describe an inference method based on the aggregation of rules. To facilitate a convenient reference, we denote the inference method in Section 4.4 as *FeaToVis*, since it extracts explicit rules which map from data features to visualization design choices. In this section, we briefly introduce another alternative inference method which first estimates the embeddings of the new data column and then infers how the new column is visually encoded (denoted as *DataToVis*). A comparison between *FeaToVis* and *DataToVis* is shown in Fig. 9.



(a) *DataToVis* (b) *FeaToVis*

Fig. 9. Our two methods of inference, *DataToVis* and *FeaToVis* are shown in (a) and (b) respectively. Each point is a 2-D embedding of an entity. f_1^D - f_1^D and f_1^C - f_1^C represent two discretized continuous data features and two categorical data features of a new data column. d_1 - d_4 denote 4 data columns in the training set. r_1 - r_4 are relations connecting corresponding data features to data columns. v_n is a visualization design choice connected by r_{target}. d_{new} in (a) is the estimated embedding of new data column and v_fs in (b) are the estimated embeddings of visualization design choices. Red arrows with two heads in (a) and (b) denote the distances between embeddings. g represents the score function.

In *DataToVis*, we first estimate the embedding of a new column d_{new} by using the embeddings of all its data features F_{new} and the corresponding relations from $f_i \in F_{new}$ to d_{new} by the following equation:

$$\mathbf{d}_{new} = \frac{1}{|F_{new}|} \sum_{f_i \in F_{new}} (\mathbf{f}_i + \mathbf{r}_i), \tag{8}$$

where $\mathbf{d}_{new}$ is the approximate embedding of this new data column d_{new}, $\mathbf{f}_i$ is the embedding of a data feature entity $f_i \in \mathbb{E}_{DF} \cup \mathbb{E}_{CF}$ and $\mathbf{r}_i$ is the embedding of relation $r_i \in \mathbb{R}_{DF \to D} \cup \mathbb{R}_{CF \to D}$, which connects f_i to the existing data columns in the training dataset-visualization pairs. As shown in Fig. 9(a), this method of estimating the embedding of d_{new} can be viewed as computing the average embedding of all the existing data columns that share some common data features with d_{new}. With $\mathbf{d}_{new}$, we calculate the score that a visualization design choice v_n which is connected to d_{new} with relation r_{target} as follows:

$$\begin{aligned} g(d_{new}, r_{target}, v_n) &= -||\mathbf{d}_{new}, \mathbf{r}_{target}, \mathbf{v}_n|| \\ &= -|| \frac{1}{|F_{new}|} \sum_{f_i \in F_{new}} (\mathbf{f}_i + \mathbf{r}_i + \mathbf{r}_{target} - \mathbf{v}_n)||, \end{aligned} \tag{9}$$

Based on the score function, we calculate the score of all possible design choices and recommend the choice with the highest score. We compare the two inference methods in detail, which shows that they are comparable with each other (see Table 4). However, *FeaToVis* can generate interpretable rules to illustrate the reasons for the recommendation results. Thus, we finally adopt *FeaToVis* as our inference method.

Table 4. This table shows the result of our quantitative evaluation on our inference method. The best result is in bold. Overall, *DataToVis* and *FeaToVis* are comparable. Among all metrics, a smaller MR indicates better performance while larger accuracy and Hits@2 are better.

		Axis	Visualization Type	
		Accuracy	MR	Hits@2
TransE-adv	*DataToVis*	**0.7386**	1.9630	0.7467
	FeaToVis	0.7350	**1.9567**	**0.7489**
TransE	*DataToVis*	0.7213	1.9847	0.7397
	FeaToVis	0.7214	1.9718	0.7445
RotatE	*DataToVis*	0.7137	1.9816	0.7448
	FeaToVis	0.7193	1.9608	0.7458

C Statistics of Visualization Corpus in Evaluation

Following VizML, we construct a visualization corpus by randomly keeping a dataset-visualization pair for each user. The detailed statistics are shown in Table 5.

Table 5. Statistics of our visualization corpus.

Number of datasets		88,548
Number of columns		309,335
Number of columns of visualization types	Bar	57,425
	Box	19,959
	Heatmap	1,571
	Histogram	8,179
	Line	109,782
	Scatter	112,419
Number of columns on x- or y- axis	x	114,587
	y	188,443

D Preliminary Experiments on Parameters of MDLP

As introduced in Section 5.1, two key parameters of MDLP are used to control the minimum proportion of samples to split an interval (*min_proportion_split*) and the minimum proportion of samples in an interval (*min_proportion_interval*), respectively. To further study how these parameters may affect the final performance, we conducted some preliminary experiments following the same settings described in Section 5.1 and Section 5.2. The results of our experiments are in Table 6. In Table 6, we can observe that our method has a consistent performance using all sets of parameters and the results are slightly better when both parameters are small. A potential reason is that when intervals are more fine-grained, the generated rules can be more accurate. However, as discussed in Section 5.1, to balance the intuitiveness and the performance, we prefer to control the number of intervals for each data feature. Specifically, we set *min_proportion_split* and *min_proportion_interval* to 0.1 and 0.05 respectively in experiments.

© 2021 IEEE. This is the author's version of the article that has been published in IEEE Transactions on Visualization and Computer Graphics. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

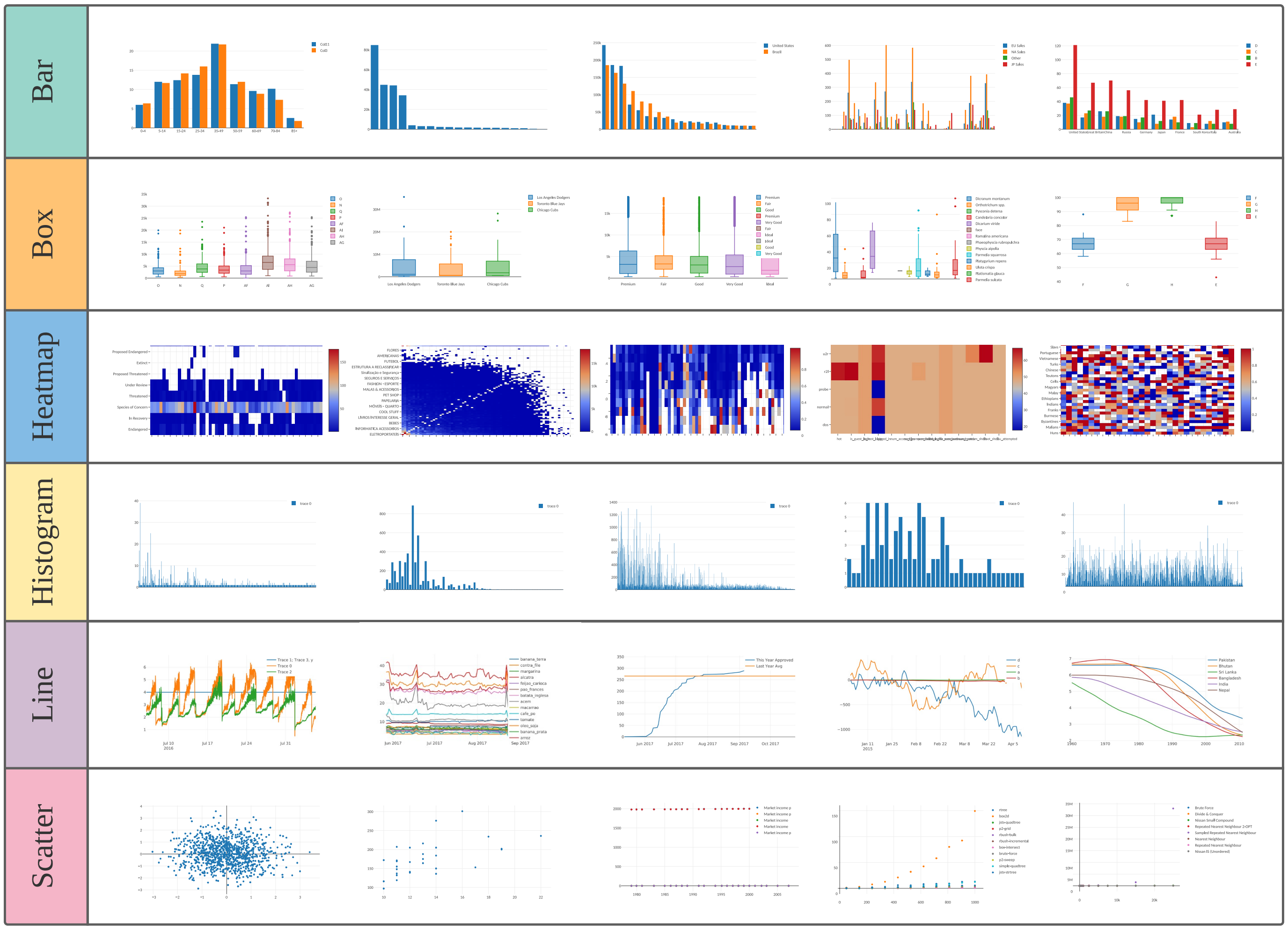


Fig. 10. A gallery of recommended visualizations.

Table 6. This table shows the result of our experiments on parameters of MDLP. *Min_proportion_split* denotes the minimum proportion of samples to split an interval while *min_proportion_interval* denotes the minimum proportion of samples in an interval after splitting. The best results are in bold. Overall, the performance is consistent among different sets of parameters and the results are better when both parameters are small. Among all metrics, a smaller MR indicates better performance while larger accuracy and Hits@2 are better.

Min_proportion_split	Min_proportion_interval	Axis	Visualization Type	
		Accuracy	MR	Hits@2
0.05	0.025	**0.7437**	**1.9556**	0.7478
0.1	0.025	0.7360	1.9565	0.7468
0.1	0.05	0.7350	1.9567	**0.7489**
0.2	0.05	0.7225	1.9704	0.7437
0.2	0.1	0.7242	1.9796	0.7433

Table 7. This table shows the top five rules on the axes where a data column should be encoded.

Feature	Axis
The name of the data column contains the character "x".	x
The name of the data column contains the character "y".	y
Values in the data column are sorted.	x
Values in the data column are not sorted.	y
The general data type of the data column is quantitative.	y

E Example of Rules on Axes

Table 7 presents the top five rules on axes where a data column should be encoded. In our paper, "x-axis" is the horizontal axis of the chart, while "y-axis" is the vertical axis. The first two rules in Table 7 are obviously correct, since the characters in column names explicitly describe on which axis the user wants to encode the column. The next two rules regarding whether the values are sorted match our visualization practice well. For example, temporal values are often sorted and encoded on the x-axis. In real applications, quantitative values can be encoded on both axes. However, quantitative values are more often encoded on the y-axis if the data type of the other column is categorical. Thus, the last rule also makes sense. Furthermore, users' preference on y-axis when encoding quantitative data is also reflected in the learnt weights of rules in Draco [30]. In general, these rules about axes are appropriate, which confirms the effectiveness of our method.

F Gallery of Recommended Visualizations

Fig. 10 shows a few examples of the recommended visualizations by *KG4Vis*.

Table 8. This table shows the detailed continuous data feature list used in our approach.

	Feature Name	Explanation
Values	num_unique_elements	This feature counts the number of unique values.
	sortedness	This feature measures the how sorted the values are.
	%_outliers_15iqr %_outliers_3iqr %_outliers_3std %_outliers_1_99	These features measure the proportion of outliers according to 1.5IQR/3IQR/3Std/(1%, 99%) rule.
	unique_percent	This feature measures the proportion of unique values in a data column.
	entropy gini	These features measure how disordered the values are.
	skewness kurtosis moment_5 moment_6 moment_7 moment_8 moment_9 moment_10	These features reflect properties of values' distribution.
	log_space_seq_coef lin_space_seq_coef	These features measure how much the values are in log/linear space.
	quant_coeff_disp med_abs_dev avg_abs_dev coeff_var std var	These features measure the variation of values in a data column.
	normality_p normality_statistic	These features reflect how much the values are distributed normally.
	normalized_range range	These features reflect the range of values in a data column.
	q25 q75 normalized_median normalized_mean min max mean median length	These features are some basic statistical measurements of a data column.
	mode_percent	This feature measures the proportion of mode values in a data column.
	#_missing_elements missing_percent	These features reflect the number/ proportion of missing values in a data column.
	mean_value_length median_value_length	They measure the length of mean/ median values in a data column.
	min_length_of_value std_length_of_value max_length_of_value	They measure the minimum/ standard deviation/maximum length of values in a data column.
Names	#_of_words_in_name	This feature measures how many words in the column name.
	name_length	This feature measures the length of the column name.
	#_uppercase_char	This feature measures how many upper case in the column name.
	field_name_length	This feature measures how many characters in the column name.

Table 9. This table shows the detailed categorical data feature list used in our approach.

	Feature Name	Explanation
Types	specific_string specific_integer specific_decimal specific_datetime	The specific data type of this data column is string/integer/decimal/datetime.
	general_temporal general_quantitative general_categorial	The general data type of this data column is temporal/quantitative/categorial.
Values	has_none	The data column contains missing value.
	is_monotonic	Values are monotonic.
	is_lin_space	Values are in linear space.
	is_log_space	Values are in log space.
	is_unique	All values are unique.
	is_sorted	All values are sorted.
	has_outliers_15iqr has_outliers_3iqr has_outliers_3std has_outliers_1_99	Outlier exists according to 1.5IQR/3IQR/ 3Std/(1%, 99%) rule.
	is_normal_1 is_normal_5	Values are normally distributed with $p < 0.01/p < 0.05$.
	is_only_field	The data column is the only one within the dataset.
Names	1st_uppercase	The column name starts with an upper case.
	x_in_name y_in_name id_in_name time_in_name digit_in_name whitespace_in_name dollar_in_name pounds_in_name euro_in_name yen_in_name	A word or symbol "x", "y","id", "time", digit, whitespace, "$","€", "£", "¥" is in the column name.