VizML: A Machine Learning Approach to Visualization Recommendation

Kevin Hu, Michiel A Bakker, Stephen Li, Tim Kraska, César A. Hidalgo

VizML: A Machine Learning Approach to Visualization Recommendation

Kevin Hu
MIT Media Lab
kzh@mit.edu

Michiel A. Bakker
MIT Media Lab
bakker@mit.edu

Stephen Li
MIT Media Lab
sli2014@mit.edu

Tim Kraska
MIT CSAIL
kraska@mit.edu

César Hidalgo
MIT Media Lab
hidalgo@mit.edu

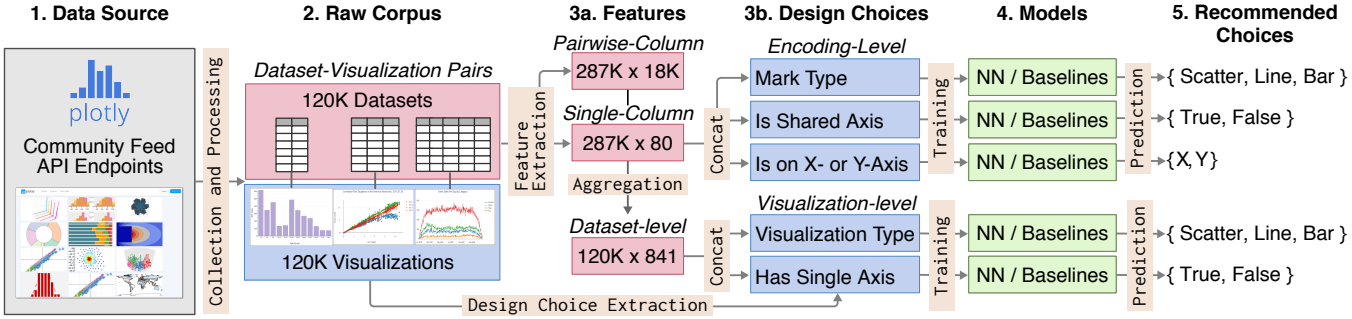


Figure 1: Diagram of data processing and analysis flow in VizML, starting from (1) the original Plotly Community Feed API endpoints, proceeding to (2) the deduplicated dataset-visualization pairs, (3a) features describing each individual column, pair of columns, and dataset, (3b) design choices extracted from visualizations, (4) task-specific models trained on these features, and (5) potential recommended design choices.

ABSTRACT

Visualization recommender systems aim to lower the barrier to exploring basic visualizations by automatically generating results for analysts to search and select, rather than manually specify. Here, we demonstrate a novel machine learning-based approach to visualization recommendation that learns visualization design choices from a large corpus of datasets and associated visualizations. First, we identify five key design choices made by analysts while creating visualizations, such as selecting a visualization type and choosing to encode a column along the X- or Y-axis. We train models to predict these design choices using one million dataset-visualization pairs collected from a popular online visualization platform. Neural networks predict these design choices with high accuracy compared to baseline models. We report and interpret feature importances from one of these baseline models. To evaluate the generalizability and uncertainty of our approach, we benchmark with a crowdsourced test set, and show that the performance of our model is comparable to human performance when predicting consensus visualization type, and exceeds that of other visualization recommender systems.

CCS CONCEPTS

• **Human-centered computing** → **Visualization design and evaluation methods**; **Visualization theory, concepts and paradigms**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Automated visualization, machine learning, crowdsourcing

ACM Reference Format:
Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo. 2019. VizML: A Machine Learning Approach to Visualization Recommendation. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3290605.3300358

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300358

1 INTRODUCTION

Knowledge workers across domains – from business to journalism to scientific research – increasingly use data visualization to generate insights, communicate findings, and make decisions [9, 26, 58]. Yet, many visualization tools have

steep learning curves due to a reliance on manual specification through code [7, 68] or clicks [2, 62]. As a result, data visualization is often inaccessible to the growing number of domain experts who lack the time or background to learn sophisticated tools.

While required to create bespoke visualizations, manual specification is unnecessary for many common use cases such as preliminary data exploration and the creation of basic visualizations. To support these use cases in which speed and breadth of exploration are more important than customizability [63], systems can leverage the finding that *the properties of a dataset influence how it can and should be visualized.* For example, prior research has shown that the accuracy with which visual channels (*e.g.* position and color) encode data depends on the type [5, 15, 67] and distribution [28] of data values.

Most recommender systems encode these visualization guidelines as collection of "if-then" statements, or *rules* [21], to automatically generate visualizations for analysts to search and select, rather than manually specify [64]. For example, APT [35], BOZ [13], and SAGE [52] generate and rank visualizations using rules informed by perceptual principles. Recent systems such as Voyager [72, 73], Show Me [34], and DIVE [23] extend these approaches with support for column selection. While effective for certain use cases [72], these *rule-based* approaches face limitations such as costly rule creation and the combinatorial explosion of possible results [1].

In contrast, *machine learning (ML)-based* systems directly learn the relationship between data and visualizations by training models on analyst interaction. While recent systems like DeepEye [33], Data2Vis [17], and Draco-Learn [37] are exciting, they do not learn to make visualization design choices as an analyst would, which impacts interpretability and ease of integration into existing systems. Furthermore, because these systems are trained with annotations on rule-generated visualizations in controlled settings, they are limited by the quantity and quality of data.

We introduce **VizML**, a ML-based approach to visualization recommendation using a large corpus of datasets and associated visualizations. To begin, we describe visualization as a process of making design choices that maximize effectiveness, which depends on dataset, task, and context. Then, we formulate visualization recommendation as a problem of developing models that learn to make design choices.

We train and test machine learning models using one million unique dataset-visualization pairs from the Plotly Community Feed [46]. We describe our process of collecting and cleaning this corpus, extracting features from each dataset, and extracting five key design choices from corresponding visualizations. Our learning task is to optimize models that use features of datasets to predict these choices.

Neural networks trained on 60% of the corpus achieve $\sim 70-95\%$ accuracy at predicting design choices in a separate 20% test set. This performance exceeds that of four simpler baseline models, which themselves out-perform random chance. We report feature importances from one of these baseline models, interpret the contribution of features to a given task, and relate them to existing research.

We evaluate the generalizability and uncertainty of our model by benchmarking against a crowdsourced test set. We construct this test set by randomly selecting datasets from Plotly, visualizing each as a bar, line, and scatter plot, and measuring the consensus of Mechanical Turk workers. Using a scoring metric that adjusts for the degree of consensus, we find that VizML performs comparably to Plotly users and Mechanical Turkers, and outperforms two rule-based and two ML-based visualization recommendation systems.

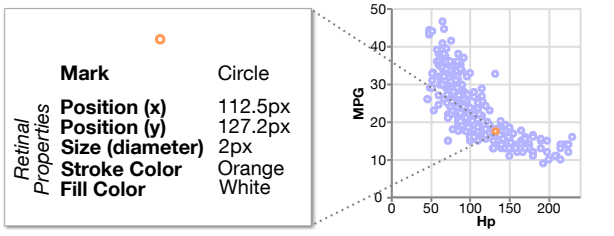
To conclude, we discuss interpretations, applications, and limitations of our initial machine learning approach to visualization recommendation. We also suggest directions for future research, such as aggregating public training and benchmarking corpora, integrating separate recommender models into an end-to-end system, and refining definitions of visualization effectiveness.

2 PROBLEM FORMULATION

Data visualization communicates information by representing data with visual elements. These representations are specified using *encodings* that map from data to the *retinal properties* (*e.g.* position, length, or color) of *graphical marks* (*e.g.* points, lines, or rectangles) [5, 12].

Concretely, consider a dataset that describes 406 automobiles (rows) with eight attributes (columns) such as miles per gallon (`MPG`), horsepower (`Hp`), and weight in pounds (`Wgt`) [50]. To create a scatterplot showing the relationship between `MPG` and `Hp`, an analyst encodes each pair of data points with the position of a circle on a 2D plane, while also specifying other retinal properties such as size and color:

Model	MPG	Cyl	Disp	Hp	Wgt	Acc	Year	Origin
chevrolet chevelle	18	8	307	130	3504	12	70	US
buick skylark 320	15	8	350	165	3693	11.5	70	US
			...					
chevy s-10	31	4	119	82	2720	19.4	82	US



To create bespoke visualizations, analysts may need to exhaustively specify encodings in detail using expressive

tools. But a scatterplot is specified with the Vega-lite [55] grammar by selecting a mark type and fields to be encoded along the x- and y-axes, and in Tableau [62] by placing the two columns onto the respective column and row shelves.

That is, to create basic visualizations in many grammars or tools, an analyst specifies higher-level *design choices*, which we define as statements that compactly and uniquely specify a bundle of lower-level encodings. Equivalently, each grammar or tool affords a design space of visualizations, which a user constrains by making choices.

We formulate basic visualization of a dataset d as a set of interrelated design choices $C = \{c\}$. However, not all design choices result in valid visualizations – some choices are incompatible with each other. For instance, encoding a categorical column with the Y position of a line mark is invalid. Therefore, the set of choices that result in valid visualizations is a subset of the space of all possible choices.

The effectiveness of a visualization can be defined by informational measures such as efficiency, accuracy, and memorability [6, 74], or emotive measures like engagement [19, 27]. Prior research also shows that effectiveness is informed by low-level perceptual principles [15, 22, 31, 51] and dataset properties [28, 54], in addition to contextual factors such as task [3, 28, 53], aesthetics [14], domain [24], audience [60], and medium [36, 57]. In other words, an analyst makes design choices C_{max} that maximize visualization effectiveness given a dataset and contextual factors.

But making design choices can be expensive. A goal of visualization recommendation is to reduce the cost of creating visualizations by automatically suggesting a subset of design choices $C_{rec} \subseteq C$ that maximize effectiveness. Trained with a corpus of datasets $\{d\}$ and corresponding design choices $\{C\}$, ML-based recommender systems treat recommendation as an optimization problem, such that predicted $C_{rec} \sim C_{max}$. A more detailed formulation of the learning task is included in the Supplementary Material (SM) Section S1.

3 RELATED WORK

We relate and compare our work to existing *Rule-based Visualization Recommender Systems* and *ML-based Visualization Recommender Systems*.

Rule-based Visualization Recommender Systems

Visualization recommender systems either suggest data queries (selecting *what* data to visualize) or visual encodings (*how* to visualize selected data) [71]. Data query recommenders vary widely in their approaches [59, 69], with recent systems optimizing statistical "utility" functions [18, 65]. Though specifying data queries is crucial to visualization, it is a distinct task from design choice recommendation.

Most visual encoding recommenders implement guidelines informed the seminal work of Bertin [5] and Cleveland and McGill [15]. This approach is exemplified by Mackinlay's **APT** [35] – the *ur*-recommender system – which enumerates, filters, and scores visualizations using *expressiveness* and perceptual *effectiveness* criteria. The closely related **SAGE** [52], **BOZ** [13], and **Show Me** [34] support more data, encoding, and task types. Recently, hybrid systems such as **Voyager** [71–73], **Explore in Google Sheets** [20, 66], **VizDeck** [43], and **DIVE** [23] combine visual encoding rules with the recommendation of visualizations that include non-selected columns.

Though effective for many use cases, these systems suffer from three major limitations. First, visualization is a complex process that may require modelling non-linear relationships that are difficult to capture with simple rules. Second, crafting rule sets is a costly process that relies on expert judgment. Lastly, as the dimension of input data increases, the combinatorial nature of rules result in an explosion of possible recommendations.

ML-based Visualization Recommender Systems

The guidelines encoded by rule-based systems often derive from experimental findings and expert experience. Therefore, an indirect manner, heuristics distill best practices learned from another analyst's experience of creating and consuming visualizations. Instead of aggregating best practices learned from data and representing them in a system with rules, ML-based systems propose to train models that learn directly from data and can be embedded into systems *as-is*. A schematic comparison of ML-based visualization recommender systems can be found in the SM Section S2.

DeepEye [33] combines rule-based visualization generation with models trained to 1) classify a visualization as "good" or "bad" and 2) rank lists of visualizations. The DeepEye corpus consists of 33,412 bivariate visualizations of columns drawn from 42 public datasets. 100 students annotated these visualizations as good/bad, and compared 285,236 pairs. These annotations, combined with 14 features for each column pair, train a decision tree for classification and a ranking neural network [10] for the "learning to rank" task.

Data2Vis [17] uses a neural machine translation approach to create a sequence-to-sequence model that maps JSON-encoded datasets to Vega-lite visualization specifications. The model is trained using 4,300 automatically generated Vega-Lite examples, consisting of 1-3 variables, generated from 11 distinct datasets. Model predictions are qualitatively validated by examining the visualizations generated from 24 common datasets.

Draco-Learn [37] learns trade-offs between constraints in Draco, a formal model that represents 1) visualizations as logical facts and 2) design guidelines as hard and soft constraints. Constraint weights are learned using a ranking

support vector machine trained on ranked pairs of visualizations harvested from graphical perception studies [28, 53]. Draco then recommends visualizations that satisfy these constraints by solving a combinatorial optimization problem.

VizML differs from these systems in three major respects. In terms of the *learning task*, DeepEye learns to classify and rank visualizations, Data2Vis learns an end-to-end generation model, and Draco-Learn learns soft constraints weights. By learning to predict design choices, VizML models are easier to quantitatively validate, provide interpretable measures of feature importance, and can be more easily integrated into visualization systems.

In terms of *data quantity*, the VizML training corpus is orders of magnitude larger than that of DeepEye and Data2Vis. The size of our corpus permits the use of 1) large feature sets that capture many aspects of a dataset and 2) high-capacity models such as deep neural networks.

The third major difference is one of *data quality*. In contrast to the few datasets used to train the three existing systems, the datasets used to train VizML models are extremely diverse in shape, structure, and distribution. Furthermore, the visualizations used by other ML-based recommender systems are generated by rule-based systems and evaluated under controlled settings. The corpus used by VizML is the result of real visual analysis by analysts on their own datasets.

However, VizML faces two major limitations. First, these three ML-based systems recommend both data queries and visual encodings, while VizML only recommends the latter. Second, in this paper, we do not create an application that employs our visualization model. Design considerations for user-facing systems that productively and properly employ ML-based visualization recommendation are important, but beyond the scope of this paper.

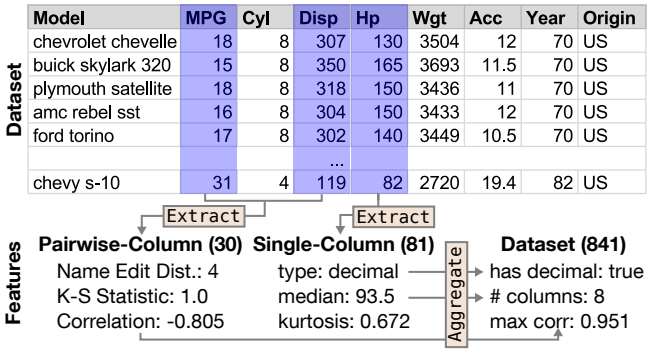


Figure 2: Extracting features from the Automobile MPG dataset [50].

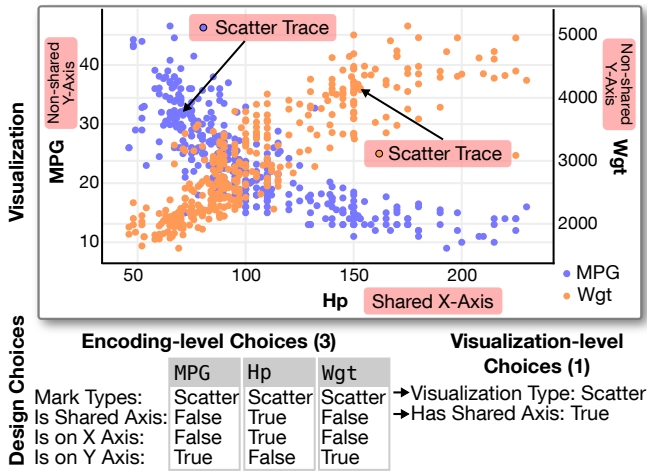


Figure 3: Extracting design choices from a dual-axis scatterplot visualizing three columns of the MPG dataset.

4 DATA

We describe our process for extracting features and design choices from the processed Plotly data. These are steps **1**, **2** and **3** in Figure 1. In the SM Section S3, we describe our process for collecting and cleaning the corpus of 2.3 million dataset-visualization pairs from the Plotly Community Feed [44, 46] and provide a description of the data. This paper is the first time the *Plotly corpus*, generated by 143,007 unique users, is used to train visualization recommender systems. The corpus along with analysis scripts is publicly available at `https://vizml.media.mit.edu`.

Feature Extraction

We map each dataset to 841 features, mapped from 81 single-column features and 30 pairwise-column features using 16 aggregation functions. Detail on each of the features is found in Table S2 in the SM Section S4.

Each column is described by **81 single-column features** across four categories. The **Dimensions (D)** feature is the number of rows in a column. **Types (T)** features capture whether a column is categorical, temporal, or quantitative. **Values (V)** features describe the statistical and structural properties of the values within a column. **Names (N)** features describe the column name. We distinguish between these feature categories for three reasons. First, these categories let us organize how we create and interpret features. Second, we can observe the contribution of different types of features. Third, some categories of features may be less generalizable than others. We order these categories (**D** → **T** → **V** → **N**) by how biased we expect those features to be towards the Plotly corpus.

We describe each pair of columns with **30 pairwise-column features**. These features fall into two categories: **Values** and **Names**. Note that many pairwise-column features depend

on the individual column types determined through single-column feature extraction. For instance, the Pearson correlation coefficient requires two numeric columns, and the "number of shared values" feature requires two categorical columns.

We create **841 dataset-level features** by aggregating these single- and pairwise-column features using the **16 aggregation functions** shown in Table S2c in SM Section S4. These aggregation functions convert single-column features (across all columns) and pairwise-column features (across all pairs of columns) into scalar values. For example, given a dataset, we can count the number of columns, describe the percent of columns that are categorical, and compute the mean correlation between all pairs of quantitative columns. Two other approaches to incorporating single-column features are to train separate models per number of columns, or to include column features with padding. Neither approach yielded a significant improvement over the results reported in Section 6.

Design Choice Extraction

Each visualization in Plotly consists of traces that associate collections of data with visual elements. Therefore, we extract an analyst's design choices by parsing these traces. Examples of **encoding-level design choices** include *mark type*, such as scatter, line, bar; and *X or Y column encoding*, which specifies which column is represented on which axis; and whether or not an X or Y column is the single column represented along that axis. For example, the visualization in Figure 3 consists of two scatter traces, both of which have the same column encoded on the X axis (`Hp`), and two distinct columns encoded on the Y axis (`MPG` and `Wgt`).

By aggregating these encoding-level design choices, we can characterize **visualization-level design choices** of a chart. Within our corpus, over 90% of the visualizations consist of homogeneous mark types. Therefore, we use *visualization type* to describe the type shared among all traces, and also determined whether the visualization *has a shared axis*. The example in Figure 3 has a scatter visualization type and a single shared axis (X).

5 METHODS

We describe our feature processing pipeline, the machine learning models we use, how we train those models, and how we evaluate performance. These are steps **4** and **5** of the workflow in Figure 1.

Feature Processing

We converted raw features into a form suitable for modeling using a five-stage pipeline. First, we apply one-hot encoding to categorical features. Second, we set numeric values above the 99th percentile or below the 1st percentile to those respective cut-offs. Third, we imputed missing categorical values using the mode of non-missing values, and missing numeric values with the mean of non-missing values. Fourth, we removed the mean of numeric fields and scaled to unit variance.

Lastly, we randomly removed datasets that were exact deduplicates of each other, resulting in unique 1,066,443 datasets and 2,884,437 columns. However, many datasets are slight modifications of each other, uploaded by the same user. Therefore, we removed all but one randomly selected dataset per user, which also removed bias towards more prolific Plotly users. This aggressive deduplication resulted in a final corpus of **119,815 datasets** and **287,416 columns**. Results from only exact deduplication result in significantly higher within-corpus test accuracies, while a soft threshold-based deduplication results in similar test accuracies.

Prediction Tasks

Our task is to train models that use the features described in Section 4 to predict the design choices also described in Section 4. **Two visualization-level prediction tasks** use dataset-level features to predict visualization-level design choices:

(1) **Visualization Type [VT]**: *2-, 3-, and 6-class*
Given all traces are the same type, what type is it?

Scatter	*Line*	*Bar*	*Box*	*Histogram*	*Pie*
44829	26209	16002	4981	4091	3144

(2) **Has Shared Axis [HSA]**: *2-class*
Do the traces all share one axis (either X or Y)?

False	*True*
95723	24092

The **three encoding-level prediction tasks** use features about individual columns to predict how they are visually encoded. These prediction tasks consider each column independently, instead of alongside other columns in the same dataset, which accounts for the effect of column order.

(1) **Mark Type [MT]**: *2-, 3-, and 6-class*
What mark type is used to represent this column?

Scatter	*Line*	*Bar*	*Box*	*Histo*	*Heatmap*
68931	64726	30023	13125	5163	1032

(2) **Is Shared X-axis or Y-axis [ISA]**: *2-class*
Is this column the only column encoded on its axis?

False	*True*
275886	11530

(3) **Is on X-axis or Y-axis [XY]**: *2-class*
Is this column encoded on the X-axis or the Y-axis?

False	*True*
144364	142814

For the **Visualization Type** and **Mark Type** tasks, the 2-class task predicts line vs. bar, and the 3-class predicts scatter vs. line vs. bar. Though Plotly supports over twenty mark types, we limited prediction outcomes to the few types that comprise the majority of visualizations within our corpus. This heterogeneity of visualization types is consistent with the findings of [4, 38].

Neural Network and Baseline Models

Our primary model is a fully-connected feedforward neural network (NN) with 3 hidden layers, each consisting of 1,000 neurons with ReLU activation functions and implemented using PyTorch [41]. For comparison, we chose four simpler baseline models, all implemented using scikit-learn [42] with default parameters: naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR) and random forest (RF). Randomized parameter search for each model did not result in a significant performance increase over the reported results.

For all models, we split the data into 60/20/20 train/validation/test sets and train and test each model five times using 5-fold cross-validation. The reported results are thus test results averaged across the five test sets. We oversample the train, validation, and test sets to the size of the majority class while ensuring no overlap between the three sets. We oversample because of the heterogeneous outcomes, naive classifiers guessing the base rates would have high accuracies. Balanced classes also allow us to report standard accuracies (fraction of correct predictions), ideal for interpretability and generalizing results to multi-class cases $C > 2$, in contrast to measures such as the F_1 score.

The neural network was trained with the Adam optimizer and a mini-batch size of 200. The learning rate was initialized at 5×10^{-4}, and followed a learning rate schedule that reduces the learning rate by a factor of 10 upon encountering a plateau, defined as 10 epochs during which validation accuracy does not increase beyond a threshold of 10^{-3}. Training ended after the third decrease in the learning rate, or at 100 epochs. Weight decay, dropout and batch normalization did not significantly improve performances.

In terms of features, we constructed four different feature sets by incrementally adding the **Dimensions (D)**, **Types (T)**, **Values (V)**, and **Names (N)** categories of features, in that order. We refer to these feature sets as **D**, **D+T**, **D+T+V**, and **D+T+V+N=All**. The neural network was trained and tested using all four feature sets independently. The four baseline models only used the full feature set (D+T+V+N=All).

6 EVALUATING PERFORMANCE

We report performance of each model on the five prediction tasks in the barplot in Figure **??** and in Table 2 in the SM. The neural network consistently outperforms the baseline models and model performance generally progressed as NB < KNN < LR ≈ RF < NN. That said, the performance of both RF and LR is not significantly lower than that of the NN in some cases. Simpler classifiers may be desirable, depending on the need for optimized accuracy, and the trade-off with other factors such as interpretability and training cost.

Because the four feature sets are a sequence of supersets (D ⊂ D+T ⊂ D+T+V ⊂ D+T+V+N), we consider the accuracy of each feature set above and beyond the previous. For instance, the increase in accuracy of a model trained on D+T+V over a model trained on D+T is a measure of the contribution of value-based (V) features. These marginal accuracies are visualized alongside baseline model accuracies in Figure **??** in the SM.

We note that the value-based feature set (*e.g.* the statistical properties of a column) contribute more to performance than the type-based feature set (*e.g.* whether a column is categorical), potentially because there are many more value-based features than type-based features. Or, because many value-based features are dependent on column type, there may be overlapping information between value- and type-based features.

Interpreting Feature Importances

Feature importances help relate our results to prior literature and inform design guidelines for rule-based systems. Here, we determine feature importances for our top performing random forest models using the standard mean decrease impurity (MDI) measure [8, 32]. We choose this method for its interpretability and its stability across runs. The top ten features for five different tasks are shown in Table 2a and for all other tasks in the SM Table S3.

We first note the importance of **dimensionality** ( ), like the length of columns (*i.e.* the number of rows) or the number of columns. For example, the length of a column is the second most important feature for predicting whether that column is visualized as a line or bar trace. The dependence of mark type on number of visual elements is consistent with heuristics like "keep the total number of bars under 12" for showing individual differences in a bar chart [61], and not creating pie charts with more "more than five to seven" slices [30]. The dependence on number of columns is related to the heuristics described by Bertin [5] and encoded in Show Me [34].

Features related to **column type** ( ) are consistently important for each prediction task. For example, whether a dataset contains a string type column is the fifth most important feature for determining two-class visualization type. The dependence of visualization type choice on column data type is consistent with the type-dependency of the perceptual properties of visual encodings described by Mackinlay [35] and Cleveland and McGill [15].

(a) Prediction accuracies for two visualization-level tasks.

Model	Features	d	Visualization Type C=2	Visualization Type C=3	Visualization Type C=6	HSA C=2
NN	D	15	66.3	50.4	51.3	84.1
	D+T	52	75.7	59.6	60.8	86.7
	D+T+V	717	84.5	77.2	87.7	95.4
	All	841	**86.0**	**79.4**	**89.4**	**97.3**
NB	All	841	63.4	49.5	46.2	72.9
KNN	All	841	76.5	59.9	53.8	81.5
LR	All	841	**81.8**	64.9	**69.0**	90.2
RF	All	841	81.2	**65.1**	66.6	**90.4**
N_{raw} (in 1000s)			42.2	87.0	99.3	119

(b) Prediction accuracies for three encoding-level tasks.

Model	Features	d	Mark Type C=2	Mark Type C=3	Mark Type C=6	ISA C=2	XY C=2
NN	D	1	65.2	44.3	30.5	52.1	49.9
	D+T	9	68.5	46.8	35.0	70.3	57.3
	D+T+V	66	79.4	59.4	76.0	95.5	67.4
	All	81	**84.9**	**67.8**	**82.9**	**98.3**	**83.1**
NB	All	81	57.6	41.1	27.4	81.2	70.0
KNN	All	81	72.4	51.9	37.8	72.0	65.6
LR	All	81	73.6	52.6	43.7	**84.8**	79.1
RF	All	81	**78.3**	**60.1**	**46.7**	74.2	**83.4**
N_{raw} (in 1000s)			94.7	163	183	287	287

Table 1: Design choice prediction accuracies for five models, averaged over 5-fold cross-validation. The standard error of the mean was $< 0.1\%$ for all results. Results are reported for the neural network (NN) and four baseline models: naive Bayes (NB), K-nearest neighbors (KNN), logistic regression (LR), and random forest (RF). Features are separated into four categories: dimensions (D), types (T), values (V), and names (N). N_{raw} is the size of the training set before resampling, d is the number of features, and *C* is the number of outcome classes. HSA = Has Shared Axis, ISA = Is Shared X-axis or Y-Axis, and XY = Is on X-axis or Y-axis.

(a) Feature importances for two visualization-level tasks.

#	Visualization Type (C=2)		Has Shared Axis (C=2)	
1	% Values are Mode	std	Number of Cols	
2	Min Value Length	max	Is Monotonic	%
3	Entropy	var	Field Name Length	AAD
4	Entropy	std	# Words In Name	NR
5	String Type	has	X In Name	#
6	Median Length	max	# Words In Name	range
7	Mean Value Length	AAD	Edit Distance	mean
8	Entropy	mean	Edit Distance	max
9	Entropy	max	Length	std
10	Min Value Length	AAD	Edit Distance	NR

(b) Feature importances for three encoding-level tasks.

#	Mark Type (C=2)	Is Shared Axis (C=2)	Is X or Y Axis (C=2)
1	Entropy	# Words In Name	Y In Name
2	Length	Unique Percent	X In Name
3	Sortedness	Field Name Length	Field Name Length
4	% Outliers (1.5IQR)	Is Sorted	Sortedness
5	Field Name Length	Sortedness	Length
6	Lin Space Seq Coeff	X In Name	Entropy
7	% Outliers (3IQR)	Y In Name	Lin Space Seq Coeff
8	Norm. Mean	Lin Space Seq Coeff	Kurtosis
9	Skewness	Min	# Uppercase Chars
10	Norm. Range	Length	Skewness

Table 2: Top-10 feature importances determined by mean decrease impurity for the top performing random forest models. The second column in the visualization-level importances table describes how each feature was aggregated, using the abbreviations in Table S2c. Colors represent different feature groupings: dimensions ( ), type ( ), statistical [Q] ( ), statistical [C] ( ), sequence ( ), scale of variation ( ), outlier ( ), unique ( ), name ( ), and pairwise-relationship ( ).

Statistical features (quantitative: , categorical:) such as Gini, entropy, skewness and kurtosis are important across the board. The presence of these higher order moments is striking because lower-order moments such as mean and variance are low in importance. The importance of these moments highlight the potential importance of capturing high-level characteristics of distributional shape. These observations support the use of statistical properties in visualization recommendation, like in [59, 70], but also the use of higher-order properties such as skewness, kurtosis, and entropy in systems such as Foresight [16], VizDeck [43], and Draco [37].

Measures of orderedness ( ), specifically sortedness and monotonicity, are important for many tasks. Sortedness is defined as the element-wise correlation between the sorted and unsorted values of a column, that is $|corr(\boldsymbol{X}_{raw}, \boldsymbol{X}_{sorted})|$, which lies in the range $[0, 1]$. Monotonicity is determined by strictly increasing or decreasing values in $\boldsymbol{X}_{raw}$. The importance of these features could be due to pre-sorting of a dataset by an analyst, which may reveal which column is considered to be the independent or explanatory column, which is typically visualized along the X-axis. While intuitive, we have not seen orderedness factor into existing systems.

We also note the importance of the linear or logarithmic space sequence coefficients, which are heuristic-based features that roughly capture the **scale of variation** ( ). Specifically, the linear space sequence coefficient is determined by $std(Y)/mean(Y)$, where $Y = \{X_i - X_{i-1}\}$ with

$i = (1 + 1)..N$ for the linear space sequence coefficient, and $Y = \{X_i/X_{i-1}\}$ with $i = (1 + 1)..N$ for the logarithmic space sequence coefficient. A column "is" linear or logarithmic if its coefficient $\leq 10^{-3}$. Both coefficients are important in all four selected encoding-level prediction tasks. We have not seen similar measures of scale used in prior systems.

In sum, the diversity of the features in Table 2a and Table S3 in the SM suggest that rule-based recommender systems should include more features than the current type based features most systems rely on (*e.g.* [34, 73]). Furthermore, the task-specific ranking of features, as well as the non-linear dependencies in the models, make it even harder for rule-based systems to perform well across tasks and domains and thus further emphasize the need for ML-based recommender systems

7 BENCHMARKING WITH CROWDSOURCED EFFECTIVENESS

We expand our definition of effectiveness from a binary to a continuous function that can be determined through crowdsourced consensus. Then, we describe our experimental procedure for gathering visualization type evaluations from Mechanical Turk workers. We compare different models at predicting these evaluations using a consensus-based effectiveness score.

Modeling and Measuring Effectiveness

As discussed in Section 2, we model data visualization as a process of making a set of design choices $C = \{c\}$ that maximize an effectiveness criteria *Eff* that depends on dataset d, task, and context. In Section 6, we predict these design choices by training a machine learning model on a corpus of dataset-design choice pairs $[(d, c_d)]$. But because each dataset was visualized only once by each user, we consider the user choices c_d to be effective, and each other choice as ineffective. That is, we consider effectiveness to be binary.

But prior research suggests that effectiveness is continuous. For example, Saket et al. use time and accuracy preference to measure task performance [53], Borkin et al. use a normalized memorability score [6], and Cleveland and McGill use absolute error rates to measure performance on elementary perceptual tasks [15]. Discussions by visualization experts [25, 29] also suggest that multiple visualizations can be equally effective at displaying the same data.

Our effectiveness metric should be continuous and reflect the ambiguous nature of data visualization, which leads to multiple choices receiving a non-zero or even maximal score for the same dataset. This is in agreement with measures of performance for other machine learning tasks such as the BLEU score in language translation [40] and the ROUGE metric in text summarization [11], where multiple results can be (partly) correct.

To estimate this effectiveness function, we need to observe a dataset d visualized by multiple potential users. Assume that a design choice c can take on multiple discrete values $\{v\}$. For instance, we consider c the choice of **Visualization Type**, which can take on the values $\{bar, line, scatter\}$. Using n_v to denote the number of times v was chosen, we compute the probability of making choice v as $\hat{P}_c(v) = n_v/N$, and use $\{\hat{P}_c\}$ to denote the collection of probabilities across all v. We normalize the probability of choice v by the maximum probability to define an effectiveness score $\hat{Eff}_c(v) = \hat{P}_c(v) / max(\{\hat{P}_c\})$. Now, if all N users make the same choice v, only $c = v$ will get the maximimum score while every other choice $c \neq v$ will receive a zero score. However, if two choices are chosen with an equal probability and are thus both equally effective, the normalization will ensure that both receive a maximum score.

Developing this crowdsourced score that reflects the ambiguous nature of making data visualization choices serves three main purposes. First, it lets us establish uncertainty around our models – in this case, by bootstrap. Second, it lets us test whether models trained on the Plotly corpus can generalize and if Plotly users are actually making optimal choices. Lastly, it lets us benchmark against performance of the Plotly users as well as other predictors.

To generate the crowdsourced evaluation data, we recruited and successfully pre-screened 300 participants through Amazon Mechanical Turk. The data preparation and crowdsourced evaluation procedures is described in more detail in SM Section S6.

Benchmarking Procedure

We use four types of predictors in our benchmark: human, rule-based model, ML-based model, and baseline. The two human predictors are the **Plotly** predictor, which is the visualization type of the original plot created by the Plotly user, and the **MTurk** predictor is the choice of a single random Mechanical Turk participant. When evaluating the performance of individual Mechanical Turkers, that individual's vote was excluded from the set of votes used in the mode estimation.

The two rule-based predictors include one commercial system and another research system. The first, Tableau's **Show Me** feature [34], is based on the expressiveness and effectiveness criteria of Mackinlay's APT [35]. The second, the **CompassQL** recommender engine [71], powers the Voyager and Voyager 2 systems [72, 73].

The two learning-based predictors are **DeepEye** and **Data2Vis**. In all cases, we tried to make choices that maximize prediction performance, within reason. We uploaded datasets to Show Me, DeepEye, and CompassQL as comma-separated values (CSV) files, and to Data2Vis as JSON objects. Unlike

VizML and Data2Vis, DeepEye supports pie, bar, and scatter visualization types. We marked both pie and bar recommendations were both bar predictions, and scatter recommendations as line predictions in the two-type case.

For all tools, we modified the data within reason to maximize the number of valid results. For the remaining errors (4 for Data2Vis, 14 for DeepEye), and cases without returned results (12 for DeepEye and 33 for CompassQL) we assigned a random chart prediction.

Predictor performance is evaluated as the total sum of normalized effectiveness scores. This *Consensus-Adjusted Recommendation Score* (CARS) of a predictor is defined as:

$$CARS_{predictor} = \frac{1}{|D|} \sum_{d \in D} \frac{\hat{P}_c\left(\hat{c}_{predictor,\,d}\right)}{max(\{\hat{P}_c\})} \times 100 \quad (1)$$

where $|D|$ is the number of datasets (66 for two-class and 99 for three-class), $\hat{c}_{predictor,\,d}$ is the predicted visualization type for dataset d, and $\hat{P}_c$ returns the fraction of Mechanical Turker votes for a given visualization type. Note that the minimum CARS > 0%. We establish 95% confidence intervals around these scores by comparing against 10^5 bootstrap samples of the votes, which can be thought of as synthetic votes drawn from the observed probability distribution.

Benchmarking Results

We first measure the degree of consensus using the Gini coefficient, the distribution of which is shown in Figure 4. If a strong consensus was reached for all visualizations, then the Gini distributions would be strongly skewed towards the maximum, which is 1/2 for the two-class case, and 2/3 for the three-class case. Conversely, a lower Gini implies a weaker consensus, indicating an ambiguous ideal visualization type. The Gini distributions are not skewed towards either extreme, which supports the use of a soft scoring metric such as CARS over a hard measure like accuracy.

The Consensus-Adjusted Recommendation Scores for each model and task are visualized as a bar chart in Figure 5. We first compare the CARS of VizML (88.96 ± 1.66) against that of Mechanical Turkers (86.66 ± 5.38) and Plotly users (90.35 ± 1.85) for the two-class case, as shown in Figure 5a. It is surprising that VizML performs comparably to the original Plotly users, who possess domain knowledge and invested time into visualizing their own data. VizML significantly out-performs Data2Vis (75.61 ± 2.44) and DeepEye (79.12 ± 4.33). Show Me achieves a CARS of (81.70 ± 2.05), which is similar to that of CompassQL (80.98 ± 4.32). While the other recommenders were not trained to perform visualization type prediction, all perform slightly better than the random

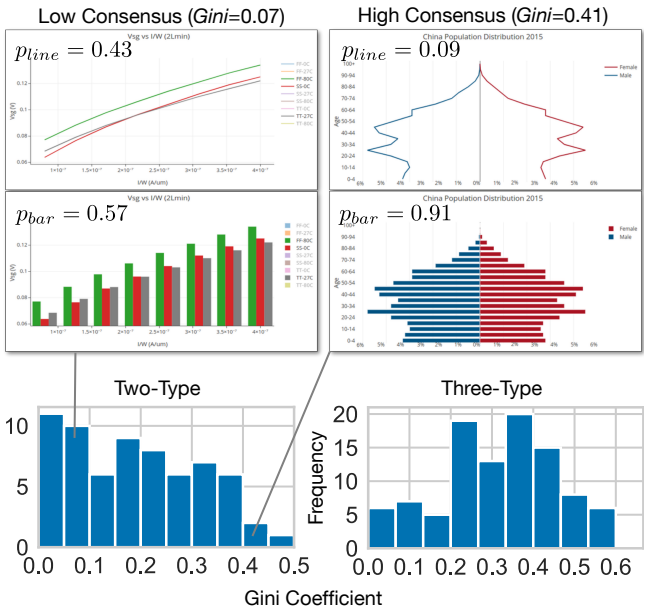


Figure 4: Distribution of Gini coefficients

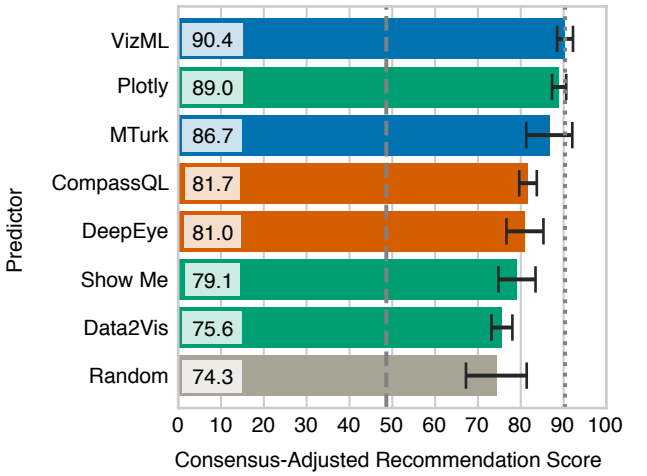
classifier (74.30 ± 7.09). For this task, the absolute minimum score was (48.61 ± 2.95).

The same results hold for the three-class case shown in Figure 5b, in which the CARS of VizML (81.18 ± 2.39) is slightly higher, but within error bars, than that of Mechanical Turkers (79.28±4.66), and Plotly users (79.58±2.44). Data2Vis (64.75 ± 3.13) and DeepEye (68.09 ± 4.11) outperform the Random (60.37 ± 6.98) with a larger margin, but still within error. CompassQL (68.95 ± 4.48) slightly surpasses Show Me (65.37 ± 2.98), also within error. The minimum score was (26.93 ± 3.46).

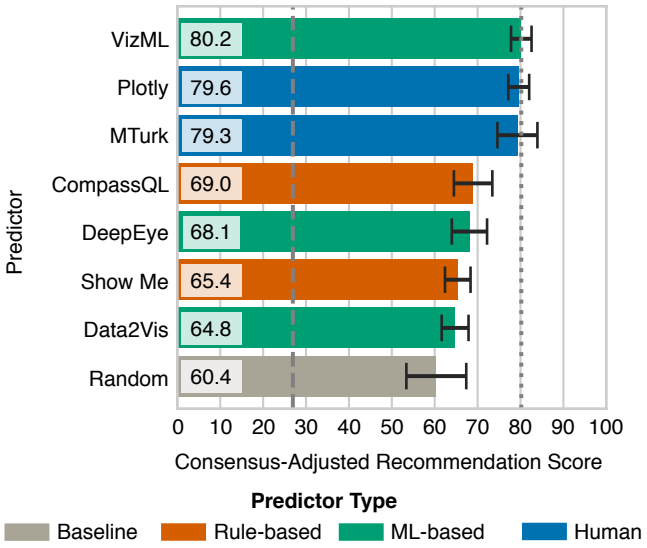
8 DISCUSSION

In this paper, we introduce VizML, a machine learning approach to visualization recommendation using a large corpus of datasets and corresponding visualizations. We identify five key prediction tasks and show that neural network classifiers attain high test accuracies on these tasks, relative to both random guessing and simpler classifiers. We also benchmark with a test set established through crowdsourced consensus, and show that the performance of neural networks is comparable that of individual humans.

Visualization system developers have multiple paths towards incorporating ML-based recommenders such as VizML into authoring workflows. Partial specification recommenders on top of existing manual specification tools, such as the Show Me [34] feature in Tableau [62], rely on design choice suggestions that could be provided by a learned model. Code-based authoring environments such as the Draco [37] and



(a) Two-type (bar/line) visualization type CARS.



(b) Three-type (bar/line/scatter) visualization type CARS.

Figure 5: Consensus-Adjusted Recommendation Score of three ML-based, two rule-based, and two human predictors when predicting consensus visualization type. Error bars show 95% bootstrapped confidence intervals, with 10^5 bootstraps. The mean minimum achievable score is the lower dashed line, while the highest achieved CARS is the upper dotted line.

Vega-Lite [55] editors, could use partial specification recommenders to power visualization "autocomplete" features, which suggest design choices in response to user interaction, in real time. Mixed-initiative systems such as Voyager [73] and DIVE [23] could leverage Top-N recommendations to present a gallery of visualizations for users to search and drill-down. Designing interactions with ML-based recommenders is an important area of future work.

In order to develop ML-based recommenders for their own systems, developers could begin by identifying user design choices and extracting simple features from data. Given sufficient volume, those features and design choices can be used to train models as we have demonstrated in this paper. Alternatively, developers can overcome the cold-start problem by using pre-trained models such as VizML. With models in hand, developers can progress further by collecting the usage analytics (*e.g.* measures of engagement such as clicks and shares) to establish customized measures of visualization effectiveness.

We acknowledge the limitations of the Plotly corpus and our approach. First, despite aggressive deduplication, our model is certainly biased towards the Plotly dataset. As a web-based platform, Plotly could draw a certain cohort of analysts, encourage certain types of plots by interface design or defaults, or be more appropriate for specific types and sizes of data. Second, neither the Plotly user nor the Mechanical Turker is an expert in data visualization. Thirdly, we acknowledge that this paper was only focused on a subset of the tasks usually considered in a visualization recommendation pipeline.

Promising avenues for future work lie in both data collection and modelling directions. On the data side, there is a need for more diverse training data from other tools (*e.g.* Many Eyes and Tableau) and pertaining to adjacent data science tasks such as feature selection and data transformation. Richer training data allows researchers to investigate the previous bias concerns, optimize visualization recommenders with a task-based (or generally multi-objective) effectiveness metric, recommend multiple views of a dataset, study complementary approaches to feature engineering, and integrate distinct design choice recommendations using a probabilistic graphical model.

Underlying each ML-based recommender model is a measure of visualization effectiveness. Determining the parameters that inform effectiveness is an open question for the visualization community. Machine learning tasks such as image annotation or medical diagnosis are often objective, in that there exists a clear human-annotated ground truth. Other tasks are subjective, such as language translation or text summarization tasks, and are benchmarked against human evaluation or against human-generated results.

Questions of objective visualization quality point towards the role of experts in visualization assessment. Visualization experts provide evaluations that are informed by experience and knowledge of perceptual studies. But if laypeople are the target audience of visualizations, the consensus opinion of crowdsourced agents may be a good measure of visualization quality. By providing a large training corpus, initial machine learning models, and a crowdsourced benchmark, VizML is a step forward in addressing these questions.

REFERENCES

[1] C. C. Aggarwal. *Recommender Systems: The Textbook*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[2] C. Ahlberg. Spotfire: An Information Exploration Environment. *SIGMOD Rec.*, 25(4):25–29, Dec. 1996.

[3] R. Amar, J. Eagan, and J. Stasko. Low-Level Components of Analytic Activity in Information Visualization. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 15–, Washington, DC, USA, 2005. IEEE Computer Society.

[4] L. Battle, P. Duan, Z. Miranda, D. Mukusheva, R. Chang, and M. Stonebraker. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 594:1–594:8, New York, NY, USA, 2018. ACM.

[5] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[6] M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva, and H. Pfister. What Makes a Visualization Memorable? *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2306–2315, Dec 2013.

[7] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[8] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[9] E. Brynjolfsson and K. McElheran. The Rapid Adoption of Data-Driven Decision-Making. *American Economic Review*, 106(5):133–39, May 2016.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to Rank Using Gradient Descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[11] C. Lin. ROUGE: a package for automatic evaluation of summaries. pages 25–26, 2004.

[12] S. K. Card, J. D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[13] S. M. Casner. Task-analytic Approach to the Automated Design of Graphic Presentations. *ACM Trans. Graph.*, 10(2):111–151, Apr. 1991.

[14] N. Cawthon and A. V. Moere. The Effect of Aesthetic on the Usability of Data Visualization. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 637–648, July 2007.

[15] 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.

[16] Ç. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati. Foresight: Rapid Data Exploration Through Guideposts. *CoRR*, abs/1709.10513, 2017.

[17] V. Dibia and Ç. Demiralp. Data2Vis: Automatic Generation of Data Visualizations Using Sequence to Sequence Recurrent Neural Networks. *CoRR*, abs/1804.03126, 2018.

[18] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis. MuVE: Efficient Multi-Objective View Recommendation for Visual Data Exploration. *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 731–742, 2016.

[19] S. Few. Data Visualization Effectiveness Profile. https://www.perceptualedge.com/articles/visual_business_intelligence/data_visualization_effectiveness_profile.pdf, 2017.

[20] Google. Explore in Google Sheets. https://www.youtube.com/watch?v=9TiXR5wwqPs, 2015.

[21] F. Hayes-Roth. Rule-based Systems. *Commun. ACM*, 28(9):921–932, Sept. 1985.

[22] J. Heer, N. Kong, and M. Agrawala. Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1303–1312, New York, NY, USA, 2009. ACM.

[23] K. Hu, D. Orghian, and C. Hidalgo. DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows. In *ACM SIGMOD Workshop on Human-in-the-Loop Data Analytics (HILDA)*. ACM, 2018.

[24] E. M. Jonathan Meddes. Improving visualization by capturing domain knowledge. volume 3960, pages 3960 – 3960 – 10, 2000.

[25] B. Jones. Data Dialogues: To Optimize or to Satisfice When Visualizing Data? https://www.tableau.com/about/blog/2016/1/data-dialogues-optimize-or-satisfice-data-visualization-48685, 2016.

[26] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, Dec. 2012.

[27] H. Kennedy, R. L. Hill, W. Allen, , and A. Kirk. In *Engaging with (big) data visualizations: Factors that affect engagement and resulting new definitions of effectiveness*, volume 21, USA, 2016. First Monday.

[28] Y. Kim and J. Heer. Assessing Effects of Task and Data Distribution on the Effectiveness of Visual Encodings. *Computer Graphics Forum (Proc. EuroVis)*, 2018.

[29] C. N. Knaflic. Is there a single right answer? http://www.storytellingwithdata.com/blog/2016/1/12/is-there-a-single-right-answer, 2016.

[30] R. Kosara. Understanding Pie Charts. https://eagereyes.org/techniques/pie-charts, 2010.

[31] Y. Liu and J. Heer. Somewhere Over the Rainbow: An Empirical Assessment of Quantitative Colormaps. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 598:1–598:12, New York, NY, USA, 2018. ACM.

[32] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding Variable Importances in Forests of Randomized Trees. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 431–439, USA, 2013. Curran Associates Inc.

[33] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards Automatic Data Visualization. *The 34th IEEE International Conference on Data Engineering (ICDE)*, 2018.

[34] J. Mackinlay, P. Hanrahan, and C. Stolte. Show Me: Automatic Presentation for Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, Nov. 2007.

[35] J. D. Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Trans. Graphics*, 5(2):110–141, 1986.

[36] P. Millais, S. L. Jones, and R. Kelly. Exploring Data in Virtual Reality: Comparisons with 2D Data Visualizations. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, pages LBW007:1–LBW007:6, New York, NY, USA, 2018. ACM.

[37] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2018.

[38] K. Morton, M. Balazinska, D. Grossman, R. Kosara, and J. Mackinlay. Public data and visualizations: How are many eyes and tableau public used for collaborative analytics? *SIGMOD Record*, 43(2):17–22, 6 2014.

[39] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari. Learning with Noisy Labels. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 1196–1204, USA, 2013. Curran Associates Inc.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, Nov. 2011.

[43] D. B. Perry, B. Howe, A. M. Key, and C. Aragon. VizDeck: Streamlining exploratory visual analytics of scientific data. In *iConference*, 2013.

[44] Plotly. Plotly. https://plot.ly, 2018.

[45] Plotly. Plot.ly Chart Studio. https://plot.ly/online-chart-maker/, 2018.

[46] Plotly. Plotly Community Feed. https://plot.ly/feed, 2018.

[47] Plotly. Plotly for Python. https://plot.ly/d3-js-for-python-and-pandas-charts/, 2018.

[48] Plotly. Plotly REST API. https://api.plot.ly/v2, 2018.

[49] Plotly. Plotly.js Open-Source Announcement. https://plot.ly/javascript/open-source-announcement, 2018.

[50] E. Ramos and D. Donoho. ASA Data Exposition Dataset. http://stat-computing.org/dataexpo/1983.html, 1983.

[51] K. Reda, P. Nalawade, and K. Ansah-Koi. Graphical Perception of Continuous Quantitative Maps: The Effects of Spatial Frequency and Colormap Design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 272:1–272:12, New York, NY, USA, 2018. ACM.

[52] S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive Graphic Design Using Automatic Presentation Knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 112–117, New York, NY, USA, 1994. ACM.

[53] B. Saket, A. Endert, and C. Demiralp. Task-Based Effectiveness of Basic Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[54] B. Santos. Evaluating visualization techniques and tools: What are the main issues. In *The AVI Workshop on Beyond Time and Errors: Novel Evaluation Methods For information Visualization (BELIV '08)*, 2008.

[55] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, Jan. 2017.

[56] A. Satyanarayan, K. Wongsuphasawat, and J. Heer. Declarative Interaction Design for Data Visualization. In *ACM User Interface Software & Technology (UIST)*, 2014.

[57] M. M. Sebrechts, J. V. Cugini, S. J. Laskowski, J. Vasilakis, and M. S. Miller. Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 3–10, New York, NY, USA, 1999. ACM.

[58] E. Segel and J. Heer. Narrative Visualization: Telling Stories with Data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, Nov. 2010.

[59] J. Seo and B. Shneiderman. A Rank-by-Feature Framework for Interactive Exploration of Multidimensional Data. 4:96–113, 2005.

[60] S. Silva, B. S. Santos, and J. Madeira. Using color in visualization: A survey. *Computers & Graphics*, 35(2):320 – 333, 2011. Virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.

[61] D. Skau. Best Practices: Maximum Elements For Different Visualization Types. https://visual.ly/blog/maximum-elements-for-visualization-types/, 2012.

[62] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional databases. *Commun. ACM*, 51(11):75–84, 2008.

[63] J. Tukey. *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company, 1977.

[64] M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. Parameswaran. Towards Visualization Recommendation Systems. *SIGMOD Rec.*, 45(4):34–39, May 2017.

[65] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. SeeDB: Automatically Generating Query Visualizations. *Proceedings of the VLDB Endowment*, 7(13):1581–1584, 2014.

[66] F. Viégas, M. Wattenberg, D. Smilkov, J. Wexler, and D. Gundrum. Generating charts from data in a data table. US 20180088753 A1., 2018.

[67] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[68] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer Publishing Company, Incorporated, 2nd edition, 2009.

[69] L. Wilkinson, A. Anand, and R. Grossman. Graph-Theoretic Scagnostics. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, Washington, DC, USA, 2005. IEEE Computer Society.

[70] G. Wills and L. Wilkinson. AutoVis: Automatic Visualization. *Information Visualization*, 9:47–6927, 2010.

[71] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards A General-Purpose Query Language for Visualization Recommendation. In *ACM SIGMOD Workshop on Human-in-the-Loop Data Analytics (HILDA)*, 2016.

[72] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2016.

[73] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *ACM Human Factors in Computing Systems (CHI)*, 2017.

[74] Y. Zhu. Measuring Effective Data Visualization . In G. Bebis, R. Boyle, B. Parvin, D. Koracin, N. Paragios, S.-M. Tanveer, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Müller, and T. Malzbender, editors, *Advances in Visual Computing*, pages 652–661, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.