Steering-by-example for Progressive Visual Analytics

MARIUS HOGRÄFER, Aarhus University
MARCO ANGELINI and GIUSEPPE SANTUCCI, Sapienza University of Rome
HANS-JÖRG SCHULZ, Aarhus University

Progressive visual analytics allows users to interact with early, partial results of long-running computations on large datasets. In this context, computational steering is often brought up as a means to prioritize the progressive computation. This is meant to focus computational resources on data subspaces of interest so as to ensure their computation is completed before all others. Yet, current approaches to select a region of the view space and then to prioritize its corresponding data subspace either require a one-to-one mapping between view and data space, or they need to establish and maintain computationally costly index structures to trace complex mappings between view and data space. We present steering-by-example, a novel interactive steering approach for progressive visual analytics, which allows prioritizing data subspaces for the progression by generating a relaxed query from a set of selected data items. Our approach works independently of the particular visualization technique and without additional index structures. First benchmark results show that steering-by-example considerably improves Precision and Recall for prioritizing unprocessed data for a selected view region, clearly outperforming random uniform sampling.

CCS Concepts: • **Human-centered computing** → **Visual analytics**; • **Computing methodologies** → *Classification and regression trees;*

Additional Key Words and Phrases: Computational steering, progressive computation, interactive data exploration

ACM Reference format:
Marius Hogräfer, Marco Angelini, Giuseppe Santucci, and Hans-Jörg Schulz. 2022. Steering-by-example for Progressive Visual Analytics. *ACM Trans. Intell. Syst. Technol.* 13, 6, Article 96 (September 2022), 26 pages.
https://doi.org/10.1145/3531229

1 INTRODUCTION

Progressive visual analytics (PVA) is a way to bring the user into the loop of long-running computations by visualizing intermediate results well before the final result is available [3]. This is particularly helpful when the dataset under analysis is very large, the computation run over that data is very complex, or, even worse, when both are true, in short, whenever running the entire analysis would take too long. PVA allows not only monitoring the running computation but also canceling it ahead of time once a good-enough result is shown [16], which user evaluations have

Marius Hogräfer and Marco Angelini contributed equally to this research.

Authors' addresses: M. Hogräfer and H.-J. Schulz, Aarhus University, ÃĚbogade 34, Aarhus, 8200, Denmark; emails: mhograefer@cs.au.dk, hjschulz@cs.au.dk; M. Angelini and G. Santucci, Sapienza University of Rome, Piazzale Aldo Moro 5, Rome, 00185, Italy; emails: angelini@diag.uniroma1.it, santucci@diag.uniroma1.it.

2157-6904/2022/09-ART96
https://doi.org/10.1145/3531229

shown to significantly outperform using "blocking," non-progressive systems in terms of insights gathered [34]. One of the most promising uses of PVA is for computational steering where the intermediate results are used by an analyst to identify subspaces of interest on which to focus the computational resources so as to prioritize their computation [19, 26].

From early on when computational steering was proposed, it was inherently tied to direct manipulation [27, 30]. Yet when wanting to use direct manipulation to steer a running computation in a PVA scenario, one quickly discovers a problem: Say, for example, we want to steer the computation by brushing a region in the still unfinished visualization of all computation results. Brushing that region would then mean to prioritize the data items inside so as to focus computational resources on that region and to complete the processing and rendering of the data items inside before all other parts of the visualization. Yet to give them this preferential treatment, we already would need to know which data items will in the end be mapped into that region, i.e., we would need the visualization already to be completed to determine those data items.

Existing computational steering approaches for PVA deal with this conundrum either by circumventing it using a one-to-one mapping between data space and view space, or they maintain a spatial index structure over the data to perform such a reverse-lookup from view space region to data subspace. A one-to-one mapping is used by the Sherpa system [9] that limits itself to chart types, which employ data attributes as axes, e.g., scatterplots and line charts, whereas the spatial index is used for progressive multidimensional scaling [29]. The index structure provides a binning of data items—those that are already rendered are binned based on their position in view space and those not yet rendered are binned based on their distance to the already rendered items in data space. Every now and then, a rebinning occurs that (1) updates the bins of newly rendered data items based on their now available position in view space, (2) subdivides overcrowded bins into smaller ones, and (3) recomputes the binning of the remaining unrendered data items based on these changes. As the bins are defined in view space, they can be overlaid as a gridlike structure on the projection, and the user can select individual bins to prioritize their associated, unrendered data items in the progression. Both of these approaches pose considerable limits: The one-to-one mapping greatly reduces the visualization possibilities to only those visual representations that use data dimensions as visual dimensions, whereas spatial indexing requires periodic updates to keep current with respect to any newly rendered data. As each of these updates incurs a full pass over the data, this limits the applicability of this approach to only moderately sized datasets.

In this article, we propose a third approach to this problem that can be utilized for progressive computations over multivariate, numerical data whenever other steering approaches fail, i.e., when no one-to-one mapping between view and data space exists and when the dataset is too large for periodic updates. This third approach relies on the idea of "query by example," where we use the data items already inside a selected region to find those that are similar and thus likely to be drawn in the selected region as well. To that end, we make use of decision tree classifiers, which have already proven useful for estimating user interest in data subspaces in non-progressive scenarios [10, 12]. The main idea is to train a decision tree that discerns between those data items already rendered inside the selected region and those outside of it. We then use the tree's decision rules to form SQL queries that return more, unrendered items of the "inside" case. To this end, we make the following contributions to the field of PVA:

- We introduce steering-by-example as an application of query-by-example for prioritizing data subspaces of interest based on selections in view space during incremental computations.

- We present a quantitative validation of steering-by-example in a series of benchmarks, showing that it significantly outperforms random uniform sampling in retrieving data for a given selected region in view space.
- We present ProSteer, an experimental visual environment for exploring steering-by-example, which is available as open source.

2 RELATED WORK

PVA divides long-running computations into small steps. As a result, analysts using PVA can interact with this ongoing process and adjust it while it is still running, facilitating progressive data exploration. According to Mühlbacher et al. [19], such interactions with an ongoing computation can be distinguished into two groups: *result control* (what the computation does) and *execution control* (how the computation does it).

Result control is defined as any "interaction with the ongoing computation to steer the final result" [19]. This encompasses, for example, the early validation of a result being computed and a potential reparametrization of the computation if the result does not meet the analyst's expectations. One type of result control is the *inner result control*, which is based on partial results being generated by an ongoing computation that can then be adjusted on the fly. The existing literature in PVA has particularly looked at the implications for UI design incurred by adjusting the parameters of a computation while it is running [5]. Complementary to this type of result control, there can also be *outer result control*, which is based on final results generated by multiple computations. Instances of this type are TPFlow [15] and the work by Xie et al. [31], which gradually lead analysts from a computation that produces an overview visualization toward computations that bring out increasingly more detailed patterns in data subspaces.

Execution control is defined as "any kind of control of the execution of the ongoing computation of the process as such" [19]. The first type of execution control and often cited benefit of PVA is the ability to cancel the computation early on, once a good-enough result has been obtained. Early cancellation has been an integral part of PVA from its inception [26] and has subsequently been shown to increase analysts' efficiency in the analysis [34]. Its main challenge is the potential to cancel a computation too early while larger changes to the result are still inbound. PVA research has thus looked at approaches to recover from such situations [13, 17]. A second type of execution control is prioritization, which refers to adjusting the order in which data are processed by a progressive computation. The idea is that data of higher interest to the user should be processed before data of lesser interest, so that the generated partial results will reflect data of interest as early as possible. This form of control is also called *interactive steering*. State-of-the-art steering approaches include Sherpa [9] and MDSteer [29].

Outside the field of PVA, there exist some approaches to facilitate prioritization. ForeCache [6] and IncVisage [23] are examples of interactive approaches for prioritizing data, while explore-by-example [10] is used to steer queries in a progressive context. In particular, the latter is of interest because of its general applicability: It extracts decision tree rules from a set of exemplars denoted by the user as being of interest. It then uses these rules to prioritize similar data elements.

Putting our approach of steering-by-example in this context, it is a method for execution control—specifically for prioritization of relevant data in the processing order. To do so in a generic way that does not depend on the type of visual mapping (as Sherpa [9] does) or on the ability to compute and maintain a spatial index (as MDSteer [29] does), our steering-by-example approach follows the idea put forth in the explore-by-example approach [10] and employs decision trees for prioritizing data items of interest.

3 THE STEERING-BY-EXAMPLE APPROACH

Next, we present steering-by-example as an approach for prioritizing subspaces of data in progressive visualization based on selections in view space. The following sections first introduce the scenario that steering-by-example addresses and then outline the algorithmic steps of our approach. Finally, some practical extensions to the general approach are discussed, which become relevant when implementing steering-by-example in a PVA system.

3.1 The Steering-by-Example Scenario

In the following, we denote the situations that benefit from our approach. To that end, we assume a function f that transforms a dataset D into D' with $D' \subset \mathbb{R}^2$. As we are specifically proposing steering-by-example for PVA, the computation of f should be complex enough to warrant the use of progression for a dataset of size $|D|$. Apart from this basic setup, steering-by-example makes no further assumptions about f. This makes steering-by-example a good fit for scenarios in which the details of f are either unknown due to the use of closed source software, or unspecified as would be the case when implementing a generic PVA library that is to work with any conceivable user-defined visualization technique. In its generality, the steering-by-example scenario specifically includes the following three cases:

(1) The dataset size $|D|$ is too large to iterate over D multiple times, effectively prohibiting its binning and re-binning into a spatial index, as it is done for the steerable, progressive MDS [29].
(2) The function $f^{-1} : D' \rightarrow D$ is unknown or does not exist at all (for instance when f's bijective property cannot be guaranteed due to dimensionality reduction), effectively prohibiting the direct lookup of all data items in a selected region of interest of D', as it is done for one-to-one mappings [9].
(3) Thefunction f is governed by a set of changing query parameters $\{p_1, \dots, p_k\}$, e.g., user location or date and time, that makes it impossible to precompute f for a wide range of possible data values from D and store the results for a table-based, reverse look-up of $D' \rightarrow D$.

Steering-by-example is able to handle these scenarios by (1) touching each data item at most once, (2) being agnostic about the used visualization, and (3) being computationally inexpensive enough to be used for highly context-dependent, ad hoc analysis scenarios. Or even more succinctly: When all existing methods fail, steering-by-example is still applicable.

3.2 Description of the Approach

Next, we describe the steering-by-example method along its four phases. Each phase corresponds to a distinct state of the underlying decision tree classifier that drives our approach. We use a decision tree as underlying classification model, which work by Dimitriadou et al. [10] has shown to perform well for data exploration, due to the following properties: (1) It can be trained quickly for interactive use, (2) it produces sufficiently powerful classification models from relatively small inputs, allowing its use early on during the progression, and most importantly, and (3) its model can be easily translated into SQL rules. Other classification techniques like SVM or deep neural networks do not fulfill these requirements, in particular the latter. A state diagram of the approach is depicted in Figure 1. The four phases are as follows:

(1) The *non-steering* phase is the default phase in which the user has either not yet selected a view region of interest or no further data items in the dataset D match the last steering query.

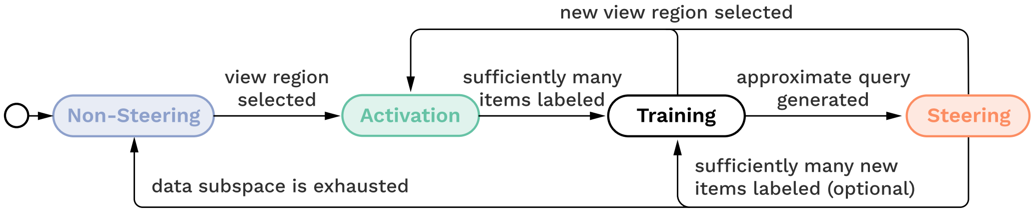

new view region selected
view region selected
sufficiently many items labeled
approximate query generated
Non-Steering
Activation
Training
Steering
data subspace is exhausted
sufficiently many new items labeled (optional)

Fig. 1. State diagram depicting the four phases of steering-by-example and the transitions between them.

(2) The *activation* phase is when data items are labeled for training based on whether they are located inside or outside a selected region of interest.
(3) The *training* phase is when the decision tree classifier is trained based on the labeled data items.
(4) The *steering* phase, in which a query approximating the characteristics of data items inside the selected view region is constructed from the decision tree rules and used to retrieve more data items likely to fall into that region.

We describe these phases in more detail below, explaining the main goal of each phase and the conditions for transitioning between them.

(1) Non-steering Phase: Initially, the PVA system does not have any indication by the user what data is of interest, as no selection has been made in the view space.

The goal of this initial phase is thus to give the users a first look at the data, allowing them to identify interesting regions that should be prioritized. Since the user interest is open at this point, this phase of steering-by-example uses a default sampling method of the dataset D based on the query parameters $\{p_1, \ldots, p_k\}$ to retrieve the next chunk of data, and the retrieved data items are not yet labeled as relevant or not. In our case, this default sampling method is random uniform sampling. In addition to being the first phase a PVA system runs after launch, the non-steering phase also functions as fallback for the steering phase, once all data items that match the approximate steering query have been retrieved. Then, the system will continue to retrieve further data items from D using default sampling. This “falling back” essentially resets the steering-by-example algorithm.

(2) Activation Phase: Once the user selects a region of interest, the system enters the activation phase. This selection can be made through any interaction in view space that identifies a region that the user is interested in. In our case, we use brushing directly on a two-dimensional rendering pane, notwithstanding that the approach could also work for other selection methods nor for higher dimensions, such as three-dimensional volumetric renderings provided a suitable brushing mechanism [14, 32]. As the user can perform the selection at any time during the progression, this phase can be reached from any other phase in the steering-by-example algorithm.

The goal of this phase is to gather a sufficient number of data the user is interested in, to guarantee an adequate quality of the classification model for steering later on. All newly retrieved data items are thus labeled as relevant or not, based on whether they lie inside or outside the indicated region of interest. In cases where data from previous iterations is still available, these data can also be considered for labeling, yet the steering-by-example algorithm does not require any caching. The system remains in this phase until sufficiently many data items are labeled as relevant. The training of the classification model has not yet been started, thus the system continues to use default sampling to retrieve the next chunk of data while in the activation phase.

(3) Training Phase: Once enough data items have been labeled as relevant, the training phase begins.

The goal of this phase is to produce a query representing the approximate inverse mapping from the selected data items of interest to data properties they and only they have in common. These properties are then to be used to query for more data having the same properties and prioritizing their computation—thus effectively steering the progression toward other relevant data. As these properties are still being established, the system continues to use its current query while training. All data items newly retrieved during this phase continue to be labeled as relevant or not. The system remains in this phase until the training of the classification model is completed. In practice, the training phase lasts only for a short period of time, depending on the complexity and size of the training dataset, due to the low computational complexity of training the decision trees.

Note that the system can also enter this phase from the steering phase if enough new data items are located inside the selection to trigger a refinement of the classification model based on that newly available data. Re-entering the training phase is however optional, as rebuilding the model often will lead to only marginal improvements of the steering quality once large portions of the dataset are already computed.

(4) Steering Phase: Once the used classification model is constructed, the system enters the steering phase.

The goal in this phase is to provide the user with all relevant data items from the dataset D that match the approximate query, by retrieving data items that will be likely plotted close to those in the view selection. In contrast to all previous phases, the system thus uses the steering query extracted from the decision tree to retrieve the next chunk of data, thus steering the progression toward interesting data subspaces. For decision trees, an SQL query for steering is constructed from the classification model as follows: Each path from the root to a leaf node in the decision tree model represents a set of decision rules that must be satisfied in order for the decision tree to classify it as relevant. Thus, the generated steering query is equal to the logical disjunction of conjunctions of these sets of rules. During the steering phase, all retrieved data items continue to be labeled as relevant.

The system exits the steering phase for two reasons: The first reason is that the data subspace indicated by the approximate query is exhausted, in which case it falls back into the non-steering phase. The second reason is that enough new data items from the steering query fell inside the region of interest to trigger further refinement of the query, in which case the system returns to the training phase.

3.3 Extension to the Basic Approach

The basic approach outlined above assumes that the user selection in view space (at some point in time) contains sufficiently many data items to train the decision tree classifier. Yet, this is not necessarily always the case. Specifically, there are two cases in which not enough relevant items can be collected: Either the region is temporarily too sparse, but sufficiently many data items of the data will land inside the selection in the future or the region will always be too sparse, even when waiting until the progression completes.

One strategy for addressing this challenge is that the threshold for what counts as "sufficient" could be a user-definable hyperparameter to the algorithm. Then, experienced users could lower this value based on their particular use case and thus increase the chance of engaging the steering. However, this is always a tradeoff with the quality of the classification model, which usually benefits from having a larger training dataset.

An additional, automated strategy is to artificially increase the size of the user selection each time no data item from the newest chunk of data falls into the selection, thus also considering data outside the original selection for training. Intuitively, this approach is intended to "broaden" the range of data that is considered interesting, thereby increasing the chance that sufficiently many data items can be collected. Conceptually, this approach resembles a query relaxation of the steering query [18]. The motivation here is that data inside this "broadened" selection remain at least somewhat interesting to the user, as it is rendered close to the region of interest. However, this approach will generally lead to a worse Precision during the steering phase compared to a model trained without increasing the selection but nevertheless remains more beneficial to the user's analysis than random uniform sampling. The degree to which the extent of the selection is increased is another hyperparameter to steering-by-example, for instance using a percentage-based increase in size. One could also make the growth proportional to the number of chunks that contained no data items located inside the selection. Then, the chance of "getting a hit" could increase with every chunk without a data item inside the selection, as the selection grows at a greater rate.

Another extension for this is to wait for a certain number of iterations before the size is increased, instead of increasing the size with every "fruitless" iteration. The idea here is to avoid reducing the steering quality for selections that only as an artifact of sampling remain empty for a single iteration but generally are densely populated. The number of chunks before the box increases would be an additional hyperparameter for steering-by-example.

4 BENCHMARKS

This section reports the results about the performance of steering-by-example for progressive visual analytics. To test the proposed solution, we defined a set of automatic test cases on which we collected evaluation metrics. We first describe the overall obtained results and then we detail the testing environment and the measures we collected with it.

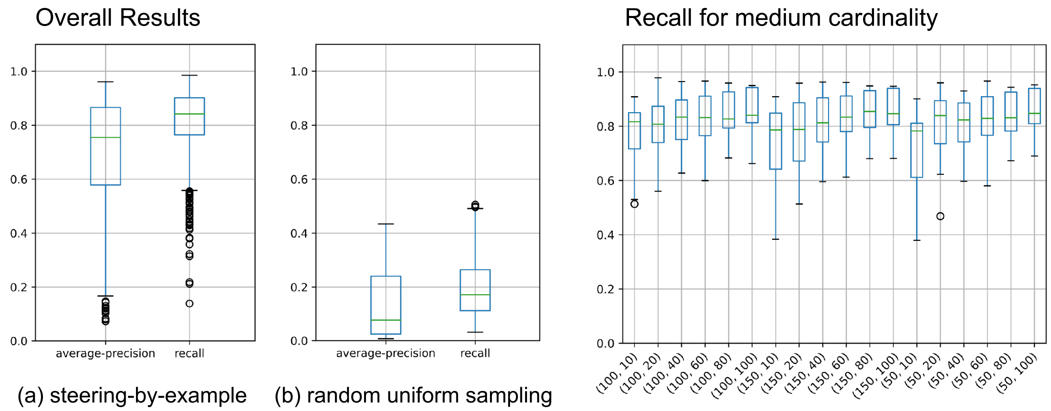
4.1 Test Results

We evaluated steering-by-example on a sample of the AirBnB dataset for Paris that consists of 64,216 data items for listings of housing options, with each listing being described by 47 dimensions, containing both numerical and categorical values. The dataset was obtained from InsideAirBnB.[1] While this dataset is relatively small, the additional computations we run on each chunk make the incremental use case worthwhile, as a full pass over the entire dataset could take up to 10 minutes.

We compared steering-by-example with random uniform sampling, in which the progression is not steered and the data space is uniformly sampled, serving as the average case. While more sophisticated sampling algorithms exist [33], we chose random uniform sampling as our baseline, as it still remains the de-facto standard sampling approach in PVA literature, since it is widely available across programming languages and frameworks, and performs reasonably well on large datasets. We tested the two approaches on more than 1,000 test cases, evaluating the performances with respect to Average-Precision and Recall metrics.

We measured a clear advantage of the steering-by-example approach over random uniform sampling, both in terms of average Precision (ca. 10 times higher) and Recall (ca. 4 times higher). Additionally, our results show independence of performance from chunk size and identify a good threshold for starting the activation phase and obtaining good performances in just 20 items. We show that the steering-by-example approach converges to the expected results much faster than

[1] http://insideairbnb.com/get-the-data.html.



(a) steering-by-example (b) random uniform sampling

Fig. 2. Left: Average Precision and Recall comparison for steering-by-example and random uniform sampling on the full set of test cases. The figure shows that steering-by-example clearly outperforms random uniform sampling for both metrics. Right: Recall results for medium cardinality group split by chunk size (100, 150, and 50 items, respectively) and activation threshold (10, 20, 40, 60, 80, and 100 items, respectively). The figure shows that chunk size has little to no effect on the Recall behavior. Conversely, activation threshold shows that from 20 items value the results tend to saturate, with only 10 values showing a consistent decrease of performance due to higher variability. Precision and Recall of the direct lookup approach are omitted, as these are—given that all the data for the one-to-one mapping is precomputed—consistently at the maximum values, performing better than steering-by-example.

random uniform sampling, similarly to the hypothetical, perfect performance of direct lookup, with only negligible overhead with respect to both per-chunk computation time and overall time. Finally, our benchmarks show how steering-by-example scales even to large datasets.

Figure 2 reports a summary of the whole experiment, showing the Average Precision and Recall box-plots for steering-by-example and random uniform sampling, clearly showing how steering-by-example outperforms random uniform sampling in both metrics.

4.2 Test Cases

To test the effectiveness of the steering-by-example solution in a systematic way, we instrumented a fully automated benchmark with the goal of producing a large and significant set of test cases. We use the following mapping function f for any given listing x in the AirBnB data: $f(x) = (priceSavings(x), walkingDistance(x))$. This function thus produces a tuple, containing the price difference to other listings and the walking distance to a fixed location of interest in the city. To reduce the precomputation overhead of the first part, we limit the benchmarks to a subset of the AirBnB data to include only listings within in a 60- to 90-Euro price range, limiting the total query size to 25,922 items.

We evaluated steering-by-example against two baseline conditions. Random uniform sampling forms the *lower baseline* condition, which retrieves the data in random order, regardless of the selection. As *upper baseline* we use the state-of-the-art approach of direct lookup that is also implemented by Sherpa [9] and that will behave like a perfect predictor: It always exclusively retrieves items inside the selection, until all data for the selection is processed. It should be noted that, because direct lookup requires a one-to-one mapping between data and view space that is not available for the mapping function we use in our scenario, we have to precompute all values needed to build a one-to-one mapping w.r.t. the user selections used in the test (latitude, longitude, min price, max price, and range of walking distance among alternative hotels). The time for precomputing these data for a single user test is about 15 minutes (on a quad core i7 and using indexed relational tables in MySQL). Obviously, it is not possible to precompute *all* data for *all* possible user selections of the five parameters of latitude, longitude, min price, max price, and walking distance. Even partitioning the user's selection domains in discrete intervals (e.g., selecting latitude and longitude in steps corresponding to 500 m, the price range in intervals of 5 US$ between 60 and 100, and the walking distance in a range of 100 m between 100 and 500) makes the computation time not feasible (about 70 months if limited to a 15 × 15 km square region in Paris). Moreover, the results of these precomputations must then be stored in a lookup table that allows to directly retrieve the data items inside a selected screen region. The size of this additional data is about 225,000 attributes per AirBnB listing.

We considered as parameters of this benchmark:

- *Chunk size*: the chunk size of the progressive process on three levels (50, 100, and 150 items per iteration)
- *Activation threshold*: the minimum number of items that must be inside the selected view region to trigger the training phase of steering-by-example (10, 20, 40, 60, 80, and 100 items each)
- *Query result cardinality*. We split the test cases into three groups based on the cardinality of each selected view region with respect to the full query:
  - *Low cardinality*: This set is formed by selected view regions containing a number of items ranging from 1% to 4.5% of the full query
  - *Medium cardinality*: This set is formed by selected view regions containing a number of items ranging from 4.5% to 22.5% of the full query
  - *High cardinality*: This set is formed by selected view regions containing a number of items ranging from 22.5% to 50% of the full query

  The rationale behind this choice is to characterize the performance of steering-by-example for different cardinalities. More in detail, we do not go over 50% of the query cardinality, because at that point the probability to correctly identify a point as part of the actually selected view region or not is equal (or higher) than chance, and results' validity would be affected. For this reason, even if the theoretical upper limit is 50%, the randomly generated selected view regions have a maximum cardinality of ~40%. We randomly generated 20 view selections per group without repetition. This results in 60 selected view regions tested that cover in a good way the variability of the region size and position on the screen (details can be found in the supplemental materials).

The combination of these benchmark parameters (3 chunk sizes, 6 thresholds) led to 18 runs per selected view region, and given the 60 query cardinalities resulted in 1,080 runs for steering-by-example and 1,080 for random uniform sampling. The data needed for the direct lookup approach were precomputed one time for each <*selected view region, chunk size*> combination, given its independence from any other parameter. We let each run execute until 100% Recall was obtained. These test cases included selected view regions of different sizes, aspect ratios, and density of the contained points to capture many different scenarios.

4.3 Detailed Measures

We tested the three approaches on all the test cases, evaluating the performances with respect to Precision, Average Precision, and Recall metrics. These metrics were computed from the standard binary classification variables with regards to whether an approach classifies or misclassifies a data item as inside or outside the selected view region.

Figure 2 reports the high-level comparison between steering-by-example and random uniform sampling for Average Precision (Precision computed on average for a single run) and Recall: Specifically, we collected those data at the iteration in which steering-by-example ends its effects (when the system switches from Steering phase to Non-Steering phase), which can vary depending on each run and on the cardinality of the tested query. For example, for the medium cardinality group, we collected the statistical median at iteration 32. For steering-by-example, the median value of the Recall is at 0.84 and the median value of Average Precision is at 0.77, both much higher than the respective values for random uniform sampling (Avg-Precision median = 0.08, Recall median = 0.18). While the Recall box for steering-by-example is very compact, showing that the effects of steering-by-example are valid for the majority of the test cases, the Average Precision box is more spread. This effect can be explained by runs in which the cardinality of items included in the selected view region is lower (specifically for the low cardinality group). Additional charts showing the performances split by query cardinality groups are present in the supplemental material. Even looking at the split group performances confirms that the steering-by-example solution achieves results much better than the random uniform sampling. Finally, the outliers present for the steering-by-example Recall are all relative to configurations in which the query result cardinality is very low (near 300 items), and the activation threshold is at the minimum (10).

Nonetheless, even for those "more difficult cases," steering-by-example obtains better performances than random uniform sampling in the same extent of the other cases, even if with overall lower values for Recall. For those reasons, we claim that, independently from the query cardinality, steering-by-example obtains much better results than random uniform sampling for both Average Precision and Recall.

We then inspected the obtained results with respect to the chunk size and activation threshold. Figure 2 shows Recall values for the medium cardinality group (the other groups' performances are reported in the supplemental material). We can observe that results do not show a significant effect of chunk size: In fact, the box-plots show a similar trend with respect to the three chunk sizes with which we experimented. This led us to discard this parameter for further analysis and consider it set up by default at 100 items. The chunk size only affects the speed of the progressive process and not its quality. However, the activation threshold shows a slight effect on both the median values and the compactness of the resulting box-plot, with higher thresholds yielding slightly more compact plots. Median values are meanwhile less affected, ranging (similarly) between values of 0.78 to 0.84, confirming what evidenced for the general case. On the lower end of the results, only test cases with an activation threshold set to 10 show degradation of results, with a minimum slightly below 0.4 for Recall. Even if those worst cases are still comparable to random uniform sampling best cases, we suggest setting up the steering-by-example with an activation threshold greater than 10 for maximizing the performances.

Having discussed all the parameters, we move on to comment on the temporal trends, considering the medium cardinality group and chunk size set at 100 items. For all the approaches, we report for all the experiments the trends of Precision and Recall metrics per iteration. Figure 3 shows statistically aggregated results on the Precision metric. The median values trend for each curve is reported with full color hue, while the alpha blended areas identify the variations between the upper and lower quartile values. In this way, the statistical variability is reported for all curves. Two vertical black dashed lines report, respectively, the median value of the starting iteration and the median value for the ending iteration for steering-by-example: Those lines identify the statistical extension of the steering phase for steering-by-example.

We can observe how Precision values are consistently high in the Steering phase, with median values just lower than 20% with respect to the flat line representing direct lookup. Additionally, it shows how steering-by-example is consistently better than random uniform sampling. In the

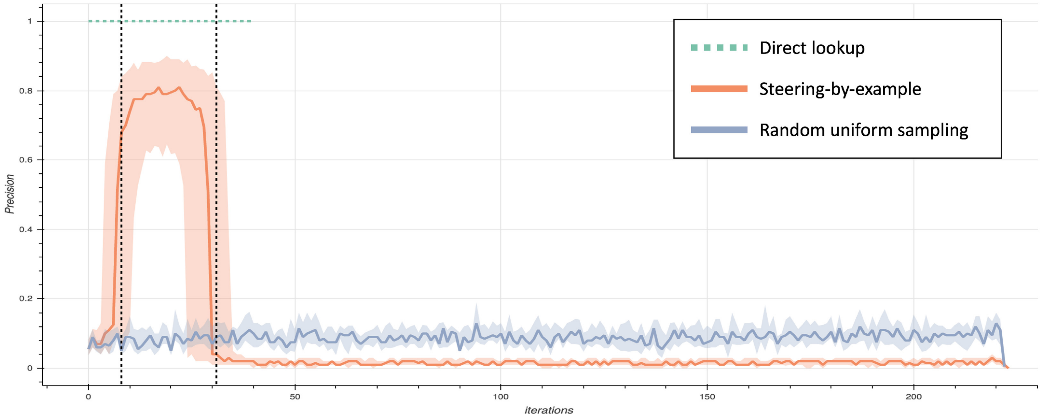


Fig. 3. Comparison of Precision trends for the direct lookup, steering-by-example, and the random uniform sampling cases. Comparison of median values shows that steering-by-example outperforms random uniform sampling for all the iterations belonging to the steering phase, with median at 0.8 less than 20% distant from the perfect predictor. In the Non-steering phase, the Precision values are very similar, with random uniform sampling slightly better due to a higher number of items left (resulting in a higher probability of finding the remaining data items). Precision of the direct lookup approach is consistently at 1, provided that the necessary one-to-one mapping has been precomputed.

Non-steering phase, the performance of steering-by-example becomes worse than random uniform sampling, even if at that moment both techniques are the same (random uniform sampling). We explain this behavior due to a lower residual probability of finding the (few) items left in the steering-by-example case.

The good performances of steering-by-example are confirmed by evaluating the Recall trend reported in Figure 4 for direct lookup. It is interesting to note how the steering-by-example Recall trend rises similarly to the perfect predictions of direct lookup, reaching very fast (a little more than 20 iterations, eventually lower if the chunk size is raised) the 0.8 level of Recall. After that point, the remaining 0.2 are achieved by reactivating random uniform sampling, which creates the long tail that at some point (slightly before the random uniform sampling) converges to the Recall 1.0. This slow convergence toward full Recall after the steering phase could be sped up by executing a new training and steering phases immediately after the end of the previous steering phase.

Overall, the benchmark demonstrated how steering-by-example outperforms the random uniform sampling with respect to Precision, Recall, and speed for all the tested cases. Full Benchmark results are available in the supplemental materials.

4.4 Implementation Details

Here we briefly outline our implementation underlying the benchmarks. More details, as well as source code of all components including the code used for benchmarking is publicly available as open source on Github.[2]

Our implementation of steering-by-example is written in Python 3.8 around the scikit-learn machine learning library [21] for its implementation of CART decision trees [8, ch. 2.3], as well as the Pandas and numpy libraries [28]. The datasets we use in our benchmarks are stored and

[2] https://vis-au.github.io/prosteer.

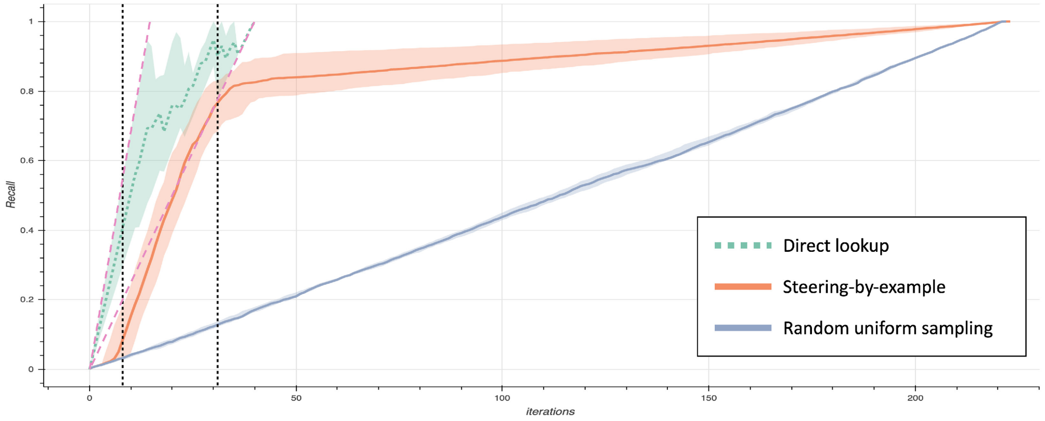


Fig. 4. Comparison of Recall trends for the direct lookup, steering-by-example, and the random uniform sampling cases. The figure shows how the steering-by-example Recall trends rise with similar speed to direct lookup during the steering-phase and clearly outperforms the random uniform sampling with *median* = 0.77 for steering-by-example and *median* = 0.12 for random uniform sampling at the median iteration in which the steering-phases end (iteration 32). The Recall of direct lookup (delimited by magenta dashed lines for reference) is consistently better than steering-by-example, given that a suitable one-to-one mapping has been precomputed.

accessed from a MySQL 8[3] database. All tests were run on an Intel Core i7 processor, running at 2.7 GHz, with 16 GByte of RAM. The data were stored on a 1-TByte SSD.

4.5 Threats to Validity of Benchmarks

Our benchmarks demonstrate the applicability of the steering-by-example approach under certain assumptions. Here we want to explicitly state the particularities of our evaluation that need to be taken into account when interpreting the results.

A first consideration is the *data type* used in the benchmarks. We have evaluated steering-by-example on a dataset containing numerical dimensions. While decision trees can generally also be trained on categorical data, we cannot make any conclusions about their performance for this data type.

Another consideration is that our evaluation relied on *rectangular selections* in view space. This design decision was made for consistency between benchmarks and to reduce the controlled variables in our testing, ensuring that all benchmarks use regularly shaped selection boundaries. Generally, steering-by-example does not rely on any particular selection mechanism. All the decision tree requires for training is a set of data items labeled as relevant. In a series of informal primary tests using the lasso selection tool in ProSteer, we also observed that our implementation can handle more complex selection shapes. Nevertheless, our benchmarks cannot guarantee that the performance of steering-by-example also applies to more complex selection boundary shapes.

A third consideration concerns the evaluation of *computation time* for the tested approaches. Under our settings, processing a single chunk requires 0.4 seconds to complete for all three approaches. The steering-by-example approach is the only one that requires additional time to train the decision tree classifier and extract from it the decision rules (training phase). To assess the impact on the overall retrieval time, we conducted a series of experiments in which we measured

[3] https://www.mysql.com.

items	training	rules	recall
100	2.1ms	73ms	0.85
150	2.5ms	110ms	0.82
200	3.3ms	163ms	0.85
250	4.3ms	224ms	0.84
300	4.5ms	248ms	0.83
350	5ms	342ms	0.87
400	5.8ms	395ms	0.87
450	6ms	426ms	0.86
500	6ms	494ms	0.86

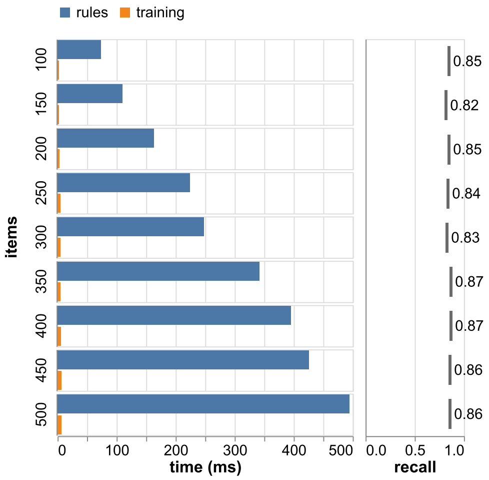


Fig. 5. Testing the impact of the number of items used for training the decision tree classifier on average training time and average time to extract the conditional rules, as well as the average Recall of the generated query, based on $N = 20$ runs. The concrete values are shown on the left, and a visualization of that data on the right. The results show that, while the time needed to extract an SQL query increases linearly with the number of items, the Recall does not increase noticeably and remains at around 0.85.

the impact of the number of items on the time it takes to train the decision tree classifier, the time it takes to extract the conditional rules in the training phase, and finally the Recall that the query produces. Figure 5 gives an overview of the results. Our observations are in line with previous work evaluating the impact of training dataset size on the performance of decision trees [20], in that the size of the training dataset did not benefit performance, while increasing the complexity of the tree structure. Overall, we conducted 20 runs for 9 different numbers of items, ranging from 100 to 500 in increments of 50. Average training times ranged from 2 ms for the 100 items cases to 6 ms for 500 items. This makes the training time not significant with respect to the overall time needed for both one iteration (400 ms) and the overall process length (220 iterations × 0.4 seconds = 1 minute and 28 seconds). Regarding the rule extraction time, we report 73 ms for 100 items. Again, this time represents 18.25% of the iteration time, as it is “paid” only once when the steering phase is activated, and 0.82% of the overall time. Therefore, the cost introduced by training the decision tree is negligible in the scope of a full computation, with a progression using steering-by-example terminating only imperceptibly later than a progression using random uniform sampling. Overall, rather than being limited by long training times from selections on very dense data, these findings suggest that we can keep training and extraction times of our approach consistently low, by drawing a fixed-size sample from those selections, and still provide a good steering performance.

For reproducibility, we removed any dependency from computing and network communication performance by precomputing the walking distances from the actual user-selected location in Paris to the relevant listings using Euclidean distance and simulating a call to the Google API introducing a delay of 0.04 seconds for each call, as well as the saving opportunity for each, by computing the difference between its price with listings within a radius of 300 m. The respective values were stored as the *Distance* and *Savings opportunity* attributes for each listing in the database. As the main design goal for our interactive environment was primarily benchmarking our approach, it thus does not allow producing listings for any arbitrary place on a map. Yet, computing the relevant listings for a single location on-demand can take hours, which was therefore not feasible

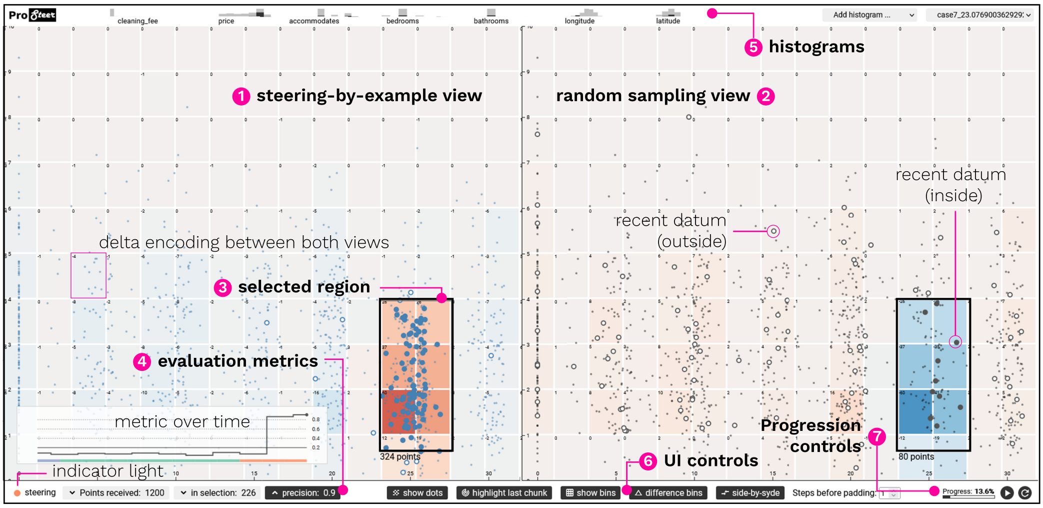


Fig. 6. Screenshot of ProSteer. The center of the interface is split in half, showing the progression on the same dataset in two views: The steering-by-example view ① shows the progression using our steering mechanism, while the random sampling view ② shows the baseline case. A heatmap in the background encodes the delta in the number of items that were retrieved in a grid cell of the view, and individual data points are rendered on top of that, with recent points rendered larger to highlight new data inside and outside of the selected region ③. Evaluation metrics ④ show how steering-by-example performs over time in a line chart based on the current phase, and histograms ⑤ show the distribution of data inside vs. outside the selection. Interface ⑥ and progression ⑦ can be controlled through widgets at the bottom.

to evaluate our method without precomputing the two measures. In addition to saving time, precomputation also isolates any potential fluctuations due to differing computation times from the results.

5 PROSTEER: AN EXPERIMENTAL VISUAL ENVIRONMENT FOR STEERING-BY-EXAMPLE

Here we present ProSteer, our interactive visual demonstrator for steering-by-example, which can be used to experiment and test-drive our approach. ProSteer is a client module to steering-by-example, in which the data are visualized and regions of interest can be defined in view space. ProSteer is not intended as a stand-alone, general-purpose visual analytics tool, but is instead designed for demonstrating steering-by-example. Thus, while the automated, "command-line" benchmarks reported in the next section provide a numerical perspective for assessing the performance of the approach, ProSteer can be used to illuminate the user perspective of utilizing steering-by-example in visual analytics. More specifically, ProSteer is designed to support the following tasks: (1) Compare the progression using steering-by-example with a progression using random uniform sampling, (2) make selections in view space, (3) explore the progression at one point in time, (4) compare the progression at different phases of the algorithm, (5) compare data inside the selection with the remaining dataset. ProSteer is implemented using the D3 visualization library [7], together with the React.[4] In the following, we will describe the interface of ProSteer along these requirements, using the labels ①–⑦ from the screenshot in Figure 6 to refer to its visual components.

[4] https://reactjs.org and TypeScript frameworks.

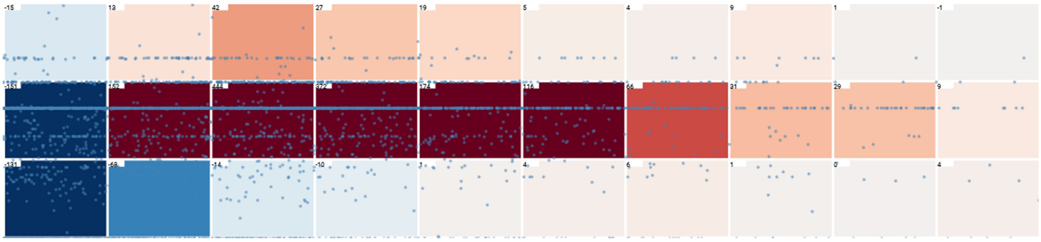


Fig. 7. Enlarged screenshot of the delta encoding in ProSteer (see ① and ② in Figure 6): Color values in a heatmap encode the difference in the number of items between the progression using steering-by-example and using random uniform sampling per grid cell. The darker the red tone, the more items lie in that cell in this view, and the darker blue, the fewer data in that cell. If the number of items is equal, then grid cells have a neutral gray tone. The number in the top left corner shows the absolute delta value per cell. On top of each cell, individual data items are encoded as dots along axes of a scatterplot (X: price saving in the neighborhood, Y: walking distance to a point of interest on the map), which are computed from the data after they are retrieved from the database and thus cannot be directly used for building an SQL query.

5.1 Comparing Steered with Non-steered Progressions

The first question we want to answer in ProSteer is whether and by how much the user gets to see interesting data faster using steering-by-example. ProSteer addresses this question through its central view that takes up most of the screen space. Data items retrieved by the steered progression (①) are visualized in a side-by-side view with a non-steered progression (②) over the same dataset. Through this juxtaposition, one can compare how steering-by-example affects the overall distribution of individual data items across regions of the view space. Other visual encodings of ProSteer rely on this side-by-side view to facilitate further comparisons.

For instance, as the changes in data layout become less apparent in later stages of the progression, when many data items are already plotted and new ones do not stick out as much, the latest chunk of items retrieved from the computation module is additionally highlighted as larger, fully opaque points. If a point is located inside the selection, then its fill color is either blue or black depending on the view, and it is white otherwise. This encoding allows to assess qualitatively, how the currently sampled region of the view space differs between the two progressions. In both views, the selection in view space of a region of interest is shown, indicating the number of data items contained in each. Additionally showing the numeric value of data items supports the quantitative assessment of steering on the one progression compared to the unsteered progression.

In addition to showing individual data items, ProSteer renders a grid-based heatmap in the background of the scatterplots (see Figure 7). Each cell of the heatmap encodes the difference in the number of data items that are rendered inside the particular region of the view space compared to the same position in the respective other view. A diverging color scale is used that encodes negative values as blue, positive values as red, and the neutral point where both progressions are equal as grey color hues. This encoding helps to get a quick impression about how much a region in view space differs in the steered and non-steered progressions, for instance during the steering phase. In addition to a qualitative assessment of the differences based on color, each cell in the heatmap also shows the numeric value that it encodes in its top-left corner.

Implicitly, encoding the difference in this way also carries uncertainty information, i.e., it informs the user that the fact that a region currently appears to be dense in the steered visualization may indeed be an artifact of the steering, that could in later iterations "even out," as more data lands in other regions. Uncertainty about observed patterns in regions targeted by the steering is closely related to the general challenge of uncertainty caused by the progression that constitutes a research challenge in its own right [13]. Our focus lies on testing algorithmic aspects of the steering-by-example, so future work is necessary to evaluate whether our encoding is an effective uncertainty encoding.

5.2 Make Selections in View Space

Another central design goal for ProSteer is the ability for user-driven selections of regions of interest, to support an interactive, customized evaluation of steering-by-example beyond the predefined test cases.

To this end, the interface supports direct brushing on the view space (③). ProSteer allows defining both rectangular selections like those used in the benchmarks as well as custom shapes using a lasso tool. Alternatively, the exact selections used in the benchmarks can be recreated using the dropdown menu in the top right. Moreover, ProSteer implements an extension to the basic steering-by-example approach, in that the size of that selection is increased with every chunk for which no data item was located inside the selection. As described in Section 3.3, one can define a custom number of chunks that should be waited, before the size is increased, using the text box in the bottom row of the interface.

5.3 Explore a Progression at One Point in Time

Another question for our benchmarks is how steering-by-example itself performs at a certain point in time. The ways in which ProSteer supports this task visualizing the state of the progression, visualizing evaluation metrics, providing widgets for customizing the UI, and controlling the progression.

The state of the progression is shown both in terms of the phase of the steering-by-example module and in terms of the progress of the computation. The phase for the latest chunk retrieved is shown as a small "indicator light" in the bottom-left corner of the interface. A progress bar (⑦) in the bottom-right corner of the interface in turn shows the percentage of data that has so far been processed by the progression. Next to that progress bar are widgets for temporarily pausing and resuming, and for fully resetting the progression.

To support exploration with steering-by-example, three evaluation metrics (④) are shown in small text boxes next to the indicator light: The number of retrieved items, the number of items that are located inside the view selection, and the resulting Precision of the latest chunk. These metrics give a quantitative view of the latest data chunk of the progression.

ProSteer also allows customizing the visual encoding for different analysis goals. Control widgets in the bottom row of the interface (⑥) for this purpose allow changing the encoding of the heatmap between absolute and delta values, toggling the side-by-side view on and off, and controlling how many data chunks without a data item from the selected region are permissible before the user selection is automatically grown in size. The top row allows setting the dimensions that are used for the visual encoding in the central view.

5.4 Comparing a Progression between Different Points in Time

Another question for our benchmarks is how the phases of steering-by-example affect performance over the duration of the progression.

For qualitative comparison, the locations of data items from the latest chunk are highlighted as opaque dots. In combination with the "indicator light," one can monitor how the distribution of the latest data items changes once the system goes in and out of the steering phase. Additionally, the heatmap encoding that visualizes the difference to a non-steered progression shows how during the steering phase, the sampling of the data gets skewed toward the selection in view space. As

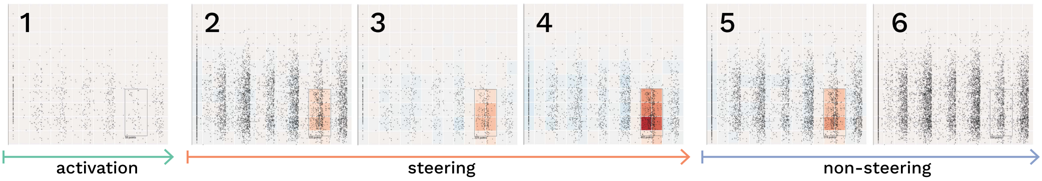


Fig. 8. Screenshot series showing how the heatmap that encodes the difference in regional data density goes from cells in gray during the activation phase (1) to red cells around the selected region during the steering phase (2–4). When entering the non-steering phase, the heatmap equalizes to gray again (5 and 6).

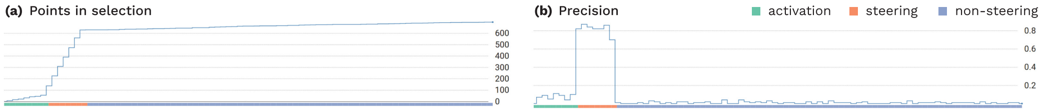


Fig. 9. Enlarged screenshots from ProSteer's evaluation metrics (see ④ in Figure 6) visualizing the items inside the user selection (a) and the Precision of retrieved items being located inside that selection (b). The colored line below the line charts indicates the state of the steering module at a certain time.

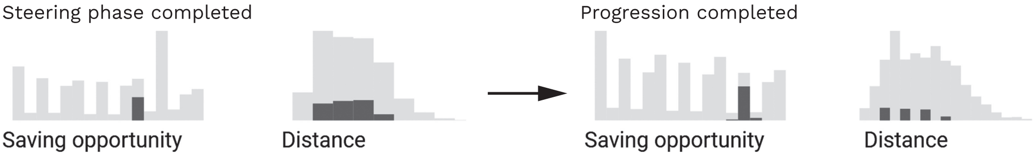


Fig. 10. Enlarged screenshot of histograms in ProSteer (see ⑤ in Figure 6) showing the overall distribution of the data compared to the data in the selected view region. The left pair shows the histograms after the steering phase is completed and the right pair shows the data when the entire progression is completed.

the progression continues, one can also observe how regional differences between the two progressions equalize (see Figure 8).

For quantitative comparisons, the system records the evaluation metrics over time as well as the phase of the algorithm that the system was in at each point. Line charts (④) that show the evolution of the metrics can be toggled when clicking on either of the text labels in the bottom left corner of the interface (see Figure 9). Below the line chart, colors indicate the phases of steering-by-example for each measurement. When hovering the mouse cursor over the line chart, individual values are shown in a tooltip.

5.5 Comparing Data Inside the Selection with the Rest of the Data

The last question we can address with ProSteer is how data items inside the selection differ from the rest of the dataset.

To qualitatively answer this question, the top row of the interface contains histograms (⑤) that can be created for each dimension of the data. Each bar in a histogram shows the percentage of data items that lie inside the selection. Based on these histograms, one can compare whether the selection is equally distributed across a dimension, which indicates a representative sample, or whether the selection is skewed toward certain values (see Figure 10). The latter case is often beneficial to a better performance of steering-by-example, as it allows the decision tree to better separate the class of relevant data items from irrelevant data.

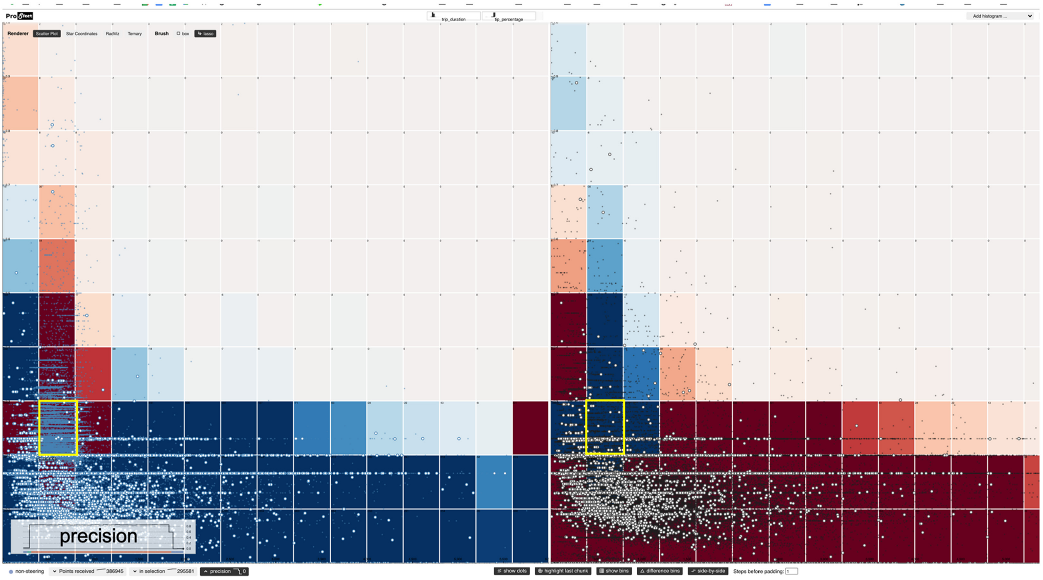


Fig. 11. Screenshot from ProSteer, applying steering-by-example to the NYC taxis dataset. The X axis shows the duration of the trip, the Y axis the ratio $tip/fare$. The steering phase handled about 350,000 taxi trips, showing a consistent Precision of about 0.82; the Recall is not available due to the time needed to precompute the ground truth. At the end of the steering phase the steered approach (left side) shows about 280,000 items in the user-selected area (yellow box) against the about 5,000 of the random approach (right side). Moreover, the heat-map of the random sampling shows that, according to the actual sample size (about 400,000 random items), the user-selected area has a very low density, making more evident the advantage of the steering phase.

For quantitative comparisons, the view space selection shows the number of data items that are currently located inside it. In combination with the "Points received" metric, one can compare whether the sample contains large or small portions of the dataset.

6 USE CASES

In this section, we assess the generality of steering-by-example for two use cases, applying the approach to a different, larger dataset and on a different visualization using dimensionality reduction.

6.1 Steering-by-Example on Large Datasets

To assess scalability toward large datasets, i.e., considering the case in which the complexity of the f function comes from the dataset size $|D|$ (see Section 3.1) and not by the computation, we have challenged ProSteer against the 112 million items dataset of the New York City 2018 Yellow Taxi Trip.[5] As a mapping function for each ride x, we used $f(x) = (tripDuration(x), tipRatio(x))$, investigating the relationship between the duration of the trip (i.e., $tripDuration(x) = x_{endTime} - x_{startTime}$) and the amount of the tip with respect to the fare (i.e., $tipRatio(x) = x_{tipAmount}/x_{totalAmount}$). The size of the dataset prohibits us from performing a full benchmark on it: Processing the whole dataset to compute the ground truth for a single use case

[5] https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq.

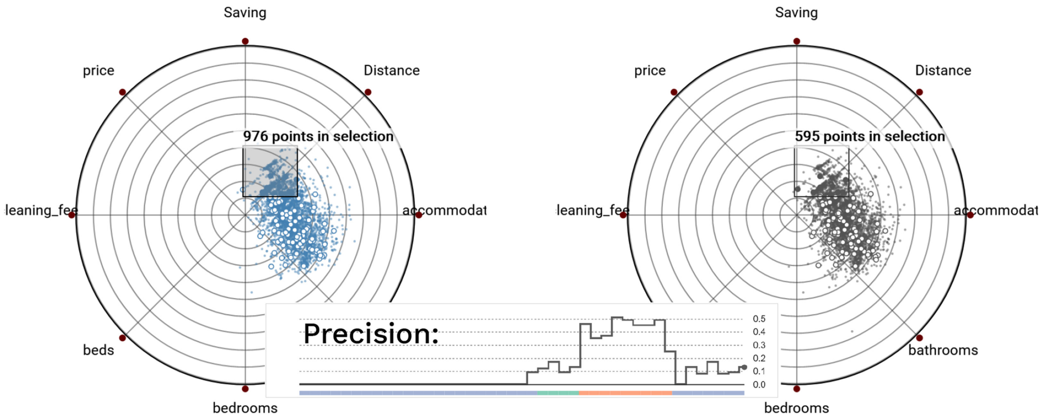


Fig. 12. Screenshot from ProSteer, using steering-by-example with the RadViz dimensionality reduction technique, compared to the same progression that uses random uniform sampling for retrieval. The selected region on the left contains 976 points after the steering phase, while the same region in the plot on the right contains only 595 points. The line chart in the foreground shows how the Precision metric increased during the steering phase from about 0.1 on average to about 0.45.

requires 15 hours and replicating the full analysis done for AirBnB (about 1,000 configurations explored) will require more than one year of computation. According to that, we just report on a qualitative test of the system, showing a comparison between steering and random sampling and terminating the analysis after the steering phase has been completed. Being the complexity of the plotting only associated with the size of the data, we have used a chunk size of 10,000 whose elaboration lasts about 5 seconds; processing the whole dataset requires about 15 hours. The results are quite similar to those observed for the AirBnB dataset: The steering outperforms the random sampling with a significantly higher Precision. Figure 11 shows an example of one of the qualitative experiments we have performed with the system, after having plotted about 380,000 trips.

6.2 Steering-by-Example on Dimensionality Reductions

We further assessed the generality of steering-by-example regarding the visual encoding. While the progressive visualization used for benchmarks placed the data along two computed axes, in dimensionality reduction the 2D position of a data item is computed from multiple dimensions, to make high-dimensional features of the data interpretable to the user [11]. In other words, the mapping function f is implemented as a complex algorithm. In particular, we used the RadViz dimensionality reduction method [1], which uses a spring-based model to compute the position of items along n dimension in a radial layout. Using the NYC taxis dataset we introduced above in Section 6.1 on a RadViz-viewer for ProSteer, we let the progression run under the same conditions as in the benchmarks and selected a region of interest. We found that in comparison with our benchmarks on the scatter plot, steering-by-example performed worse in terms of the average Precision, yet still produced a noticeable increase in data in the selected region from a (subjective) user perspective when compared to the baseline scenario. For instance, Figure 12 shows screenshots from an exemplar run after the steering phase ended, indicating how the progression using steering-by-example produces almost double the number of items in the selected regions, with an average Precision of around 0.45 during the steering phase. Nevertheless, these results are only a preliminary indication of the utility of steering-by-example when combined with dimensionality reduction techniques, and further evaluations are necessary to measure (and improve) its performance in that regard.

7 USER STUDY

To assess the usability of steering-by-example, we conducted a user study with VA experts to evaluate how users would apply the approach to address an analysis goal and what their experience is in doing so. Note that the purpose of this study was to collect expert feedback on steering-by-example as a conceptual approach rather than to evaluate ProSteer as a practical visual analytics tool. We first report on the setup and procedure used, followed by an evaluation of collected results.

7.1 Setup

We recruited five participants (three male and two female, aged between 24 and 33) from our departments, who actively work in the research field of visual analytics as Ph.D. students, PostDocs, and research assistants. On a 7-point Likert scale, participants reported strong expertise with PVA (median: 6), a strong familiarity with data science (median: 6), and with visualization (median: 6). Most were familiar with computational steering (median: 4). User studies lasted between 40 and 80 minutes and were conducted in a quiet environment, using either the participants' or experimenters' laptop computers.

7.2 Procedure

The experiment was split into four distinct phases:

Positioning questionnaire: In this phase, participants were tasked to read and sign a consent form, followed by a questionnaire concerning their expertise on the subjects of steering-by-example. The questionnaire consisted of three positioning questions (Age, Gender, and Role) and of five questions for self-assessing their expertise, based on a 7-point Likert scale *(Q4: "How familiar are you with data analysis?," Q5: "How familiar are you with visual analytics?," Q6: "How familiar are you with visualization?," Q7: "How familiar are you with what is called 'progressive visual analytics'?," and Q8: "How familiar are you with what is called 'computational steering'?")*. Participants had up to 15 minutes to complete this phase.

Introduction to ProSteer: In this phase, participants followed a live demonstration of ProSteer. We illustrated its main functionalities and the available user interactions. Participants were able to interject at any moment and ask questions about any part of the approach and the software prototype that remained unclear. The demonstration lasted from a minimum of 20 minutes up to the time to which participants explicitly confirmed to have understood everything and did not have additional questions.

Interactive usage of ProSteer: In this phase, we tasked participants to load ProSteer in their environment (participants were provided materials and assistance in setting up the ProSteer prior to the user study), load the NYC taxi and AirBnB datasets, and solve the following task: "Find all apartments that are close by, which in this neighborhood allow you to save as much money as possible." This task and its intentionally vague formulation was designed to maximize exploration of alternatives to find a potential solution, implicitly asking for steering support to explore these alternatives faster. There was no fixed time span allotted to this phase, but a suggested time of 20 minutes was communicated to the participants. Participants could end this phase prior to this limit if they found their analysis results to be sufficient.

Evaluation questionnaire: Finally, in this phase the participants were asked to fill out an evaluation questionnaire on their usage experience of ProSteer. The questionnaire was composed of eight open questions (Q9: "Do you understand what is going on in the interface? What questions arise?," Q10: "Change blindness of new points arriving in the interface during progression. Is that confusing to you?," Q11: "Does focusing the progression on a certain region help?," Q12: "Do you find it challenging that you do not fully understand the steering mechanism (i.e., that it is a black box)?," Q13: "What are your considerations with respect to the shape of the selection? Is it good? Is it enough?," Q14: "What are your considerations with respect to making multiple selections?," Q15: "Steering means biasing the progression toward a certain part of the data. What are your thoughts on that, having used ProSteer for a bit?," and Q16: "Can you see where this would fit into your data analysis workflow? Is this helpful to you?" A final question (Q19) was asking for any additional notes or comments from the participants. The time dedicated to this phase was 20 minutes.

7.3 Results

User Behavior: In terms of the usability of steering-by-example, all participants in our user study successfully used our approach to steer the progression toward interesting data and were able to complete the task. We generally observed an *iterative* interaction procedure: First, participants monitored data arriving in the progressive visualization, then they identified a region of interest, and afterward steered the progression toward that region by making a selection, returning to the observation step. This procedure matches well with the states of our steering-by-example approach, as discussed in Section 3. We did, however, also observe deviations from this general procedure. For instance, sometimes participants ($E1$, $E2$) began at the second stage, selecting a region of interest before any data had arrived and thereby skipping the initial non-steering phase in our approach. Essentially, this behavior likens the steering based on a one-to-one mapping as used by the state-of-the-art approaches we reported on in Section 2. This similarity suggests the need for future experiments, to determine the impact of different steering methods on the user experience, e.g., whether participants would actually notice that the steering mechanism differs between analysis scenarios. Other deviations we observed were participants ($E1$, $E2$) trying to make multiple selections corresponding to multiple regions of interest in the data. This behavior is characteristic for PVA, as analysts at any point in time can choose to adjust the ongoing computation to new insights gained from the latest data [16]. Thus, to further improve the usability of steering-by-example, future iterations on the approach should consider this flexibility as a design goal. A challenge that needs to be considered is whether to treat items from different selections together or separately during the training phase, i.e., whether to train one or multiple models. We also observed participants being unsure about the current phase of the steering. For instance, $E3$ at one point did not consider the steering phase indicator when looking at the interface, suggesting to make the current phase of the approach more prominent in the user interface. One adjustment to this end could be to adjust the visual style of the selection box, such that it changes color based on the current phase and its border thickness based on the current Precision.

User Experience: All participants reported that they found steering helpful in general, and could see the potential for long-running analyses that PVA is used for, in particular if users have a clear goal in mind. Expert $E2$ for example stated that "it is intuitive to support steering if the user has identified a region they are interested in" and $E1$ found it "helpful when you want to test a hypothesis on a very large dataset," while $E4$ noted that it might not be helpful for pure exploration of the data without a clear goal. When prompted, participants stated that highlighting the most recent points is useful for overcoming change blindness, and that the matrix in the background was useful in indicating that steering affected the sampling of the data in that region. All participants generally found ProSteer's interface helpful and appropriate for understanding the impact that steering has on the analysis. Some participants ($E3$, $E4$, $E5$) however noted that the encoding of recent points on the Taxi dataset could be improved, as points inside the selection became difficult to identify at later stages, and they reported on the issue of overplotting. They also suggested extending ProSteer to allow for adjusting the selected region during or after the steering phase to refine the selected region (*E*1, *E*2, *E*5).

Overall, participants agreed that the steering-by-example approach could potentially be of use for their own data analyses. These preliminary interviews provide a first impression of the utility and accessibility of steering-by-example. While the results are generally positive, our interviews can only serve as initial impressions and further evaluations are necessary to this end.

8 DISCUSSION

Steering-by-example allows prioritizing data subspaces for progressive computations for a variety of use cases. In this section, we want to take a step back and discuss the generality of the approach, looking at implicit assumptions and limitations.

8.1 Implicit Assumptions of Steering-by-Example

Selections are sensible: Steering-by-example requires a set of data items as input, which are assumed to be of interest to the user. The underlying assumption here is that these items share some characteristic in data space that makes them interesting compared to the rest of the data and that steering-by-example can identify these characteristics and translate them into an SQL query. Only if the selection of items is thus sensible can the query produced by steering-by-example retrieve other items sharing these interesting characteristics. A challenge is that if the selection is arbitrary, the decision tree would learn a similarly arbitrary set of decision rules that yields items that are potentially not interesting to the user. That assumption is based on the idea that an overarching goal of analyzing data through visualization is to use sensible mappings to encode that data, such that similar values in data space are represented close to each other in view space. Therefore, visualizations inherently support users when interactively creating sensible selections. An exception to this rule is bubble charts, in which the position is decided purely based on a circle packing algorithm, rather than the data. Beyond the visual encoding, we can also support the user through statistical approaches like active learning [25], automatically suggesting rendered items that may be of interest to the user based on data characteristics that distinguish them in the dataset.

Selections are permanent: In addition to selections being sensible, another implicit assumption to steering-by-example is that selections are permanent, i.e., an item of the data that lies inside the selection will remain inside the selection until the steering phase concludes. This assumption is based on the idea that users are expected to wait until the progressive visualization has stabilized (i.e., the “quality” of the visualization is high enough [2]), before they form an interest in the data. In our implementation and evaluation, this assumption manifests in that items do not change their position in the scatter plot, since axis extents were known upfront, and thus items are static, while the selection itself cannot be moved around by the user during steering. In some progressive scenarios, item locations are, however, not stable and thus items move into and out of the selection. For example, in progressive dimensionality reduction methods like incremental PCA discussed by Ross et al. [24], the location of rendered items needs to be updated as the model is trained on increments of the data, thus the computed axes change. Therefore, items that were previously inside the selection are potentially located outside that selection after updating positions. To address this, we in essence need to monitor the selection’s “fluctuation” and retrain the decision tree, once the number of items in the selection that changed reaches a threshold.

8.2 Limitations of Steering-by-Example in PVA

Steering may cause control bias: A general challenge in PVA is appropriately representing the uncertainty of the visualization data caused by the use of partial results. When also allowing users to steer the computation—which causes some regions of the data to be more "certain" than others—this representation becomes more complicated. When steering a progressive computation toward a region of data the user selected as interesting using any steering mechanism, the visualization is more "refined" in those regions, which can mislead users in their interpretation of the overall data distribution. The visualization is no longer "evenly incomplete," and so the visualized data needs to be interpreted under consideration of that unevenness, which users need to be (made) aware of. Micallef et al. call this phenomenon "control bias" [16]. Even though recent work by Procopio et al. has shown that the effect of control bias in PVA through steering usually does not affect users' performance, but in fact increases their certainty in the conclusions they draw [22], implementing steering mechanisms like steering-by-example into a visual analytics system nevertheless requires careful consideration of how completeness is encoded. In ProSteer, we use the heatmap in the background of the data to indicate the difference in sampling caused by steering, to immediately inform users about these effects. Furthermore, by highlighting the location of the latest data, changes in the data spread per chunk during steering become clearly noticeable.

Steering requires adequate data density: In order for steering-by-example to generate a query that yields appropriate results, our approach needs a certain number of interesting representatives as input. This number has both an upper and a lower bound. If the number of items in the selection is too low and will never reach the threshold of items needed for training even if the progression completes, then the decision tree cannot adequately capture the characteristics that make data interesting, as evident in some test cases of our benchmarks with sparse selections (see Section 4). Conversely, if the number of items is too high, then training the decision tree classifier can take too long. A potential solution to the former challenge that we discuss in Section 3.3, is to artificially increase the scope of the selection to also include neighboring regions. While this reduces the accuracy of the steering query, it can nevertheless increase the overall relevance of the retrieved items. For very sparse regions, however, even widening the scope of the selection in this way may not be sufficient, meaning that steering-by-example simply cannot support the user. For example, when users are interested in finding outliers and thus select a single item, extending that selection may select enough other data to trigger the training, yet it is exactly those data that the user was *not* interested in when selecting the outlier. Vice versa, a solution addressing selections with too many items in them is to train the decision tree on a smaller sample. This can potentially reduce the accuracy of the steering query, yet we can avoid training times becoming too long. Supporting this approach, prior work [20] as well as our own tests (see Figure 5) suggest that decision trees do not necessarily benefit from having more data in the training sets, and so sampling may be sufficient.

9 CONCLUSION AND FUTURE WORK

In this article, we presented steering-by-example, a novel approach to prioritizing data subspaces during progressive computations that are of interest to the user, based on a user selection in view space. We evaluated our approach with a series of benchmarks, which show steering-by-example significantly outperforming random uniform sampling in terms of Precision and Recall for relevant items in the selected view region. To demonstrate steering-by-example, we provide the open source visual benchmarking interface ProSteer.

Beyond extending the benchmarks toward more generalizable results (see Section 8), further questions arise for further developing the steering-by-example approach conceptually.

One field of future work is to use a continuous degree-of-interest function defined over the items in a selection rather than a strictly binary distinction between inside/relevant and outside/irrelevant. Steering-by-example currently expects that all data items inside the view selection are of equal interest to the user. However, view space selections are usually made in a fuzzy, approximate manner, rather than with surgical Precision. That means that the training data for the decision tree contains both items that are of high interest to the user as well as items that are of lesser interest. Yet, both contribute equally to the model. We want to address this in future work, by defining a degree-of-interest function over the data inside a selection, which assigns a continuous value to each data item instead of making a binary distinction. This function could for example consider how close to the center and how close to the boundary of the selection a data item lies. Then, we want to use regression trees [4] trained on these continuous values for building the model of the approximate inverse mapping function from view to data space. By increasing the expressivity of the model, the idea is that the performance of steering-by-example can be further improved.

We see a second potential field for future work in adapting steering-by-example for *iterative* progressions. In the introduction section, we have discussed PVA as a way for bringing the user into the loop of long-running computations through iterative or incremental approaches. We presented steering-by-example for *incremental* computations, in which the result of the computation progressively includes a larger subspace of a dataset, eventually ending up at the same result as were the computation run on the entire dataset. This naturally begs the question, how to adapt the approach for *iterative* computations such as node-link layouts or iterative clustering, where a computation instead progressively refines the result computed over the entire dataset, eventually converging toward a stable output. One potential steering-by-example approach could be described as focused refinement, i.e., instead of iterating over the entire dataset, the computation could prioritize subspaces similar to those selected by the user, producing a stable result first for these spaces before refining the rest of the data. Another way to involve the user in iterative computation is to let them provide examples of what they expect the final computation to look like. For instance, in the case of clustering, such an approach could be utilized as initialization of the computation: The user could manually define the clusters for a subset of the data based on their domain knowledge, and the system could then transfer this to the entire dataset, assigning similar data to similar clusters as in the user's selection. For both adaptions of steering-by-example, the research challenge lies in solving the inherent algorithmic and computational challenges.

Finally, while for practical reasons our benchmark used only a single box as user selection shape, it would be interesting to explore both multiple selections and other selection shapes like circles and lassos. Indeed, while our approach is generally independent of the way in which data items for the training phase are gathered, investigating not rectangular shapes or multiple selections in view space could both confirm the result presented in this article for the box-based selector and push for exploring different steering methods. That can lead to tailoring decision trees for specific selection shapes or using multiple decision tree models, each trained on one selection of the view space. Then, the steering query could be constructed by disjuncting the individual predicates extracted from each tree. Yet, the implications and side-effects of these solutions need to be further investigated.

ACKNOWLEDGMENTS

We are indebted to Heidrun Schumann, Georg Fuchs, Chris Weaver, and Davide Mottin with whom we discussed the steering-by-example problem in the inception phase of this work.

REFERENCES

[1] Marco Angelini, Graziano Blasilli, Simone Lenti, Alessia Palleschi, and Giuseppe Santucci. 2021. Effectiveness error: Measuring and improving radviz visual effectiveness. *IEEE Trans. Vis. Comput. Graph.* (2021). https://doi.org/10.1109/TVCG.2021.3104879

[2] Marco Angelini, Thorsten May, Giuseppe Santucci, and Hans-Jörg Schulz. 2019. On quality indicators for progressive visual analytics. In *Proceedings of the International EuroVis Workshop on Visual Analytics (EuroVA'19)*. Eurographics, Porto, Portugal, 25–29. https://doi.org/10.2312/eurova.20191120

[3] Marco Angelini, Giuseppe Santucci, Heidrun Schumann, and Hans-Jörg Schulz. 2018. A review and characterization of progressive visual analytics. *Informatics* 5, 3 (2018), 31:1–27. https://doi.org/10.3390/informatics5030031

[4] Chidanand Apté and Sholom Weiss. 1997. Data mining with decision trees and decision rules. *Fut. Gener. Comput. Syst.* 13, 2 (1997), 197–210. https://doi.org/10.1016/S0167-739X(97)00021-6

[5] Sriram Karthik Badam, Niklas Elmqvist, and Jean-Daniel Fekete. 2017. Steering the craft: UI elements and visualizations for supporting progressive visual analytics. *Comput. Graph. Forum* 36, 3 (2017), 491–502. https://doi.org/10.1111/cgf.13205

[6] Leilani Battle, Remco Chang, and Michael Stonebraker. 2016. Dynamic prefetching of data tiles for interactive visualization. In *Proceedings of the ACM Special Interest Group on Management of Data Conference (SIGMOD'16)*. ACM, New York, NY, 1363–1375. https://doi.org/10.1145/2882903.2882919

[7] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[8] Leo Breiman, Jerome Friedmann, and Charles J. Stone. 1984. *Classification and Regression Trees*. Chapman & Hall/CRC, Monterey, CA.

[9] Zhe Cui, Jayaram Kancherla, Hector Corrada Bravo, and Niklas Elmqvist. 2019. Sherpa: Leveraging user attention for computational steering in visual analytics. In *Proceedings of the Symposium on Visualization in Data Science (VDS'19)*. IEEE, 48–57. https://doi.org/10.1109/VDS48975.2019.8973384

[10] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. 2014. Explore-by-Example: An automatic query steering framework for interactive data exploration. In *Proceedings of the ACM Special Interest Group on Management of Data Conference (SIGMOD'14)*. ACM, New York, NY, 517–528. https://doi.org/10.1145/2588555.2610523

[11] Mateus Espadoto, Gabriel Appleby, Ashley Suh, Dylan Cashman, Mingwei Li, Carlos E. Scheidegger, Erik Wesley Anderson, Remco Chang, and Alexandru Cristian Telea. 2021. UnProjection: Leveraging inverse-projections for visual analytics of high-dimensional data. *IEEE Trans. Vis. Comput. Graph.* (2021). https://doi.org/10.1109/TVCG.2021.3125576 to appear.

[12] Kiran Gadhave, Jochen Görtler, Zach Cutler, Carolina Nobre, Oliver Deussen, Miriah Meyer, Jeff M. Phillips, and Alexander Lex. 2021. Predicting intent behind selections in scatterplot visualizations. *Inf. Vis.* 20, 4 (2021), 207–228. https://doi.org/10.1177/14738716211038604

[13] Jaemin Jo, Sehi L'Yi, Bongshin Lee, and Jinwook Seo. 2021. ProReveal: Progressive visual analytics with safeguards. *IEEE Trans. Vis. Comput. Graph.* 27, 7 (2021), 3109–3122. https://doi.org/10.1109/TVCG.2019.2962404

[14] Marcel Köster and Antonio Krüger. 2018. Screen space particle selection. In *Proceedings of the Computer Graphics & Visual Computing Conference (CGVC'18)*. Eurographics, 61–69. https://doi.org/10.2312/cgvc.20181208

[15] Dongyu Liu, Panpan Xu, and Liu Ren. 2019. TPFlow: Progressive partition and multidimensional pattern extraction for large-scale spatio-temporal data analysis. *IEEE Trans. Vis. Comput. Graph.* 25, 1 (2019), 1–11. https://doi.org/10.1109/TVCG.2018.2865018

[16] Luana Micallef, Hans-Jörg Schulz, Marco Angelini, Michaël Aupetit, Remco Chang, Jörn Kohlhammer, Adam Perer, and Giuseppe Santucci. 2019. The human user in progressive visual analytics. In *Proceedings of the EG Conference on Visualization (EuroVis'19)*. Eurographics Association, 19–23. https://doi.org/10.2312/evs.20191164

[17] Dominik Moritz, Danyel Fisher, Bolin Ding, and Chi Wang. 2017. Trust, but Verify: Optimistic visualizations of approximate queries for exploring big data. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'17)*. ACM, 2904–2915. https://doi.org/10.1145/3025453.3025456

[18] Davide Mottin, Alice Marascu, Senjuti Basu Roy, Gautam Das, Themis Palpanas, and Yannis Velegrakis. 2014. IQR: An interactive query relaxation system for the empty-answer problem. In *Proceedings of the ACM Special Interest Group on Management of Data Conference (SIGMOD'14)*. ACM, New York, NY, 1095–1098. https://doi.org/10.1145/2588555.2594512

[19] Thomas Mühlbacher, Harald Piringer, Samuel Gratzl, Michael Sedlmair, and Marc Streit. 2014. Opening the black box: Strategies for increased user involvement in existing algorithm implementations. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1643–1652. https://doi.org/10.1109/TVCG.2014.2346578

[20] Tim Oates and David Jensen. 1997. The effects of training set size on decision tree complexity. In *Proceedings of the International Conference on Machine Learning (ICML'97)*. Morgan Kaufmann, San Francisco, CA, 254–262. https://doi.org/10.5555/645526.657136

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html.

[22] Marianne Procopio, Ab Mosca, Carlos Scheidegger, Eugene Wu, and Remco Chang. 2021. Impact of cognitive bias on progressive visualization. *IEEE Trans. Vis. Comput. Graph.* (2021). https://doi.org/10.1109/TVCG.2021.3051013

[23] Sajjadur Rahman, Maryam Aliakbarpour, Ha Kyung Kong, Eric Blais, Karrie Karahalios, Aditya Parameswaran, and Ronitt Rubinfield. 2017. I've seen "enough": Incrementally improving visualizations to support rapid decision making. *Proc. VLDB Endow.* 10, 11 (2017), 1262–1273. https://doi.org/10.14778/3137628.3137637

[24] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. 2008. Incremental learning for robust visual tracking. *Int. J. Comput. Vis.* 77 (2008), 125–141. https://doi.org/10.1007/s11263-007-0075-7

[25] Burr Settles. 2009. *Active Learning Literature Survey.* Technical Report 1648. Department of Computer Sciences, University of Wisconsin–Madison, Madison.

[26] Charles D. Stolper, Adam Perer, and David Gotz. 2014. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Trans. Vis. Comput. Graph.* 20, 12 (2014), 1653–1662. https://doi.org/10.1109/TVCG.2014.2346574

[27] Robert van Liere, Jurriaan D. Mulder, and Jarke J. van Wijk. 1997. Computational steering. *Fut. Gener. Comput. Syst.* 12, 5 (1997), 441–450. https://doi.org/10.1016/S0167-739X(96)00029-5

[28] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[29] Matt Williams and Tamara Munzner. 2004. Steerable, progressive multidimensional scaling. In *Proceedings of the IEEE Information Visualization Conference (InfoVis'04).* IEEE, 57–64. https://doi.org/10.1109/INFVIS.2004.60

[30] Helen Wright, Robin H. Crompton, Sanjay R. Kharche, and Petra Wenisch. 2010. Steering and visualization: Enabling technologies for computational science. *Fut. Gener. Comput. Syst.* 26, 3 (2010), 506–513. https://doi.org/10.1016/j.future.2008.06.015

[31] Peng Xie, Wenyuan Tao, Jie Li, Wentao Huang, and Siming Chen. 2021. Exploring multi-dimensional data via subset embedding. *Comput. Graph. Forum* 40, 3 (2021), 75–86. https://doi.org/10.1111/cgf.14290

[32] Lingyun Yu, Konstantinos Efstathiou, Petra Isenberg, and Tobias Isenberg. 2016. CAST: Effective and efficient user interaction for context-aware selection in 3D particle clouds. *IEEE Trans. Vis. Comput. Graph.* 22, 1 (2016), 886–895. https://doi.org/10.1109/TVCG.2015.2467202

[33] Jun Yuan, Shouxing Xiang, Jiazhi Xia, Lingyun Yu, and Shixia Liu. 2021. Evaluation of sampling methods for scatterplots. *IEEE Trans. Vis. Comput. Graph.* 27, 2 (2021), 1720–1730. https://doi.org/10.1109/TVCG.2020.3030432

[34] Emanuel Zgraggen, Alex Galakatos, Andrew Crotty, Jean-Daniel Fekete, and Tim Kraska. 2017. How progressive visualizations affect exploratory analysis. *IEEE Trans. Vis. Comput. Graph.* 23, 8 (2017), 1977–1987. https://doi.org/10.1109/TVCG.2016.2607714

Received September 2021; revised March 2022; accepted April 2022