DOI: 10.1111/cgf.14528
Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

Reusing Interactive Analysis Workflows

K. Gadhave[1], Z. Cutler[1], A. Lex[1]

[1]University of Utah, USA

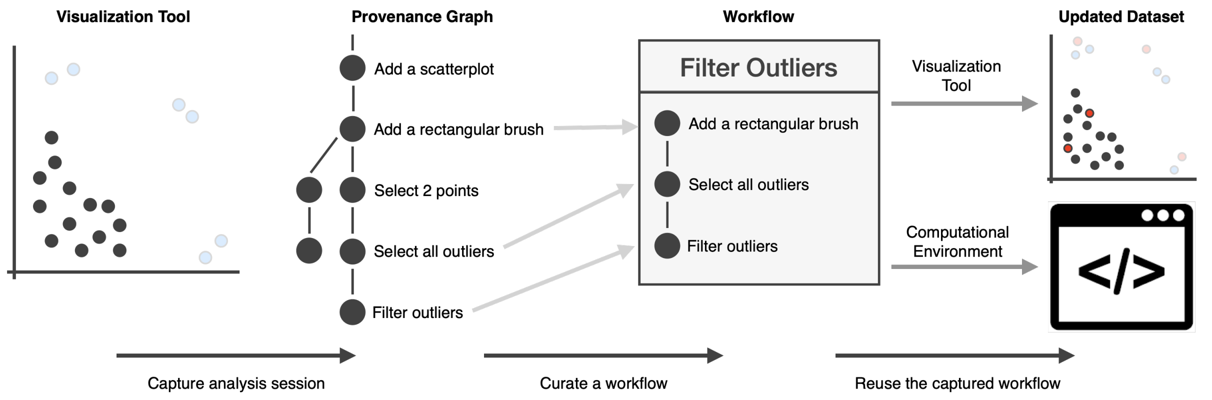


Figure 1: *The process of reusing workflows created in interactive visualizations. Interactions, such as brushes, filters, or selections based on higher level patterns (outliers in this example), are applied on a dataset. A series of actions can be extracted into a workflow. This workflow can then be applied to an updated dataset either in an interactive visualization system or in a computational environment.*

Abstract

Interactive visual analysis has many advantages, but an important disadvantage is that analysis processes and workflows cannot be easily stored and reused. This is in contrast to code-based analysis workflows, which can simply be run on updated datasets, and adapted when necessary. In this paper, we introduce methods to capture workflows in interactive visualization systems for different interactions such as selections, filters, categorizing/grouping, labeling, and aggregation. These workflows can then be applied to updated datasets, making interactive visualization sessions reusable. We demonstrate this specification using an interactive visualization system that tracks interaction provenance, and allows generating workflows from the recorded actions. The system can then be used to compare different versions of datasets and apply workflows to them. Finally, we introduce a Python library that can load workflows and apply it to updated datasets directly in a computational notebook, providing a seamless bridge between computational workflows and interactive visualization tools.

1. Introduction

Data visualization enables analysts to leverage the powerful human visual system to identify patterns and draw conclusions. When data visualizations are made interactive with selections and data transformations such as filters, labels, and aggregation, they can be used for a variety of analysis, cleanup, and processing tasks. One significant drawback of interactive visual data analysis, however, is that analysis processes remain ad hoc: when a dataset is updated or changed, the analysis has to be redone. Updating datasets, however, is very common. For example, businesses add new sales data regularly, scientists expand or correct their datasets as errors are discovered or new samples come in, and economists get updated data about various countries' indicators every year. Data visualization tools typically do not have the ability to reapply actions, such as filters, to new versions of a dataset. This lack of reusability in an interactive visual analysis is in sharp contrast to computational analysis workflows: A function that filters a dataset based on parameters can be reapplied to an updated dataset. This application comes with the usual drawbacks of computational approaches: analysts should know how to program, they are hard to write, and they cannot leverage the benefits of graphical perception.

In this paper, we propose methods to capture and reuse workflows in an interactive visualization system. Our workflows are based on series of interactions made in interactive data visualizations, such as choosing data dimensions and selecting, filtering, labelling, categorizing, or aggregating items. We introduce methods to capture these workflows in a semantically meaningful way, making them robust to changes in the datasets, as shown in Figure 1.

When reapplying a workflow, human review and potentially updates are required to ensure that the actions in a workflow still achieve the analysis goal. To address this, we introduce a review interface that visualizes changes in the dataset, the consequences of the actions in a workflow, and enables corrections, if necessary.

Finally, in addition to making workflows available for reuse within our interactive prototype, we also expose the workflows so that they can be used directly in code. This approach makes it possible to bridge between interactive visualization systems and scripted data analysis processes. For example, an analyst could do some preprocessing in a Jupyter notebook, launch an interactive visualization system from the notebook to execute a series of complex selections and data transformations that are more easily achieved in a visualization system, and then return to the notebook to apply, e.g., an algorithm to the transformed dataset. Because we now have reusable visualization workflows, all parts of such an analysis can then be reapplied to an updated dataset.

We demonstrate these capabilities in a prototype visualization system that captures interaction provenance, from which analysts can extract workflows. We show that these workflows can be reapplied to updating or changing datasets with some examples. We also introduce a Python library to bridge between the visualization system and Python code, and provide examples for these workflows.

In summary, our contribution is a method to capture workflows in interactive visualization systems that can then be reapplied to a new dataset. To ensure the accuracy of the reapply process, we introduce visualizations of changes, and review and update capabilities for these workflows. Finally, we introduce methods to use these workflows as part of computational workflows. We believe that our methods will make it possible to use interactive visualizations even for analyzing datasets that are being updated. We also push the limits of what is possible with regard to integration between computational and interactive workflows, thereby enabling analysts to leverage the best tool for each part of a job.

2. Related Work

Our work is related to creating reusable workflows in interactive systems and integration between interactive visual analysis and computational analysis, which we discuss in the following section.

2.1. Workflows

We define workflows in the context of data analysis as a sequence of steps that are executed to achieve some data transformation or analysis goal. We distinguish between two types of workflows: those that are based on a series of interactions in an interactive system, and those that are explicitly modeled.

Explicit modeling of workflows is common in scientific data analysis [DGST09]. Representative examples are systems such as Galaxy [GNTT10] for biomolecular data, SCIRun [PJ95] and Kepler [ABJ*04] for scientific/simulation data, and KNIME [BCD*09] in a machine learning context. Workflow approaches are also common for scientific visualizations applications such as volume rendering. Here, VisTrails [BCS*05] is a prolific example. Notable workflow-based systems for abstract data visualization include GraphTrail [DHRL*12], where each node in the workflow shows an aspect of a multivariate network, and VisFlow [YS17], which is tailored to tabular data. GEM-NI [ZSN*15] is a system that presents a workflow-based approach for generative design. The GEM-NI approach demonstrates the use of explicit workflows for parallel exploration of alternative designs. Explicitly modeled workflows are designed to be easily reused. At the same time, the definition of these workflows is similar to explicit code-based specification of visualizations, and thus the associated interaction cost [Lam08] is high, and the spontaneity and rapid exploration that is associated with interaction patterns such as direct manipulation [Shn83] is lost. They are easier to learn than writing code, but they have a steeper learning curve compared to interactive systems.

An alternative approach to explicit workflow modeling is tracking user actions provenance [NCE*11, XOW*20] and using this information to later extract workflows. Although several visualization systems track provenance [HMSA08,KNS04,Sv08,vdEvW13] and a few dedicated libraries to making tracking provenance easier to implement exist [CCSK19, CGL20], most tools do not explicitly curate workflows based on provenance. A notable exception is the Vistories tool [GLG*16], which enables analysts to curate interaction steps into data stories. However, these data stories cannot be reused on different datasets. Chen et al. proposed a parametric symbolic approach to support analytic provenance in their CZSaw system [CQW*14]. CZSaw enables analysts to reuse parts of the analysis process based on a previously created parametric model. The system does not support autodetection and application of patterns, and the analysis has to be done in the same system.

2.2. Interactive Visualization in Computational Environments

Computational notebook-based environments are better for narrative data analysis, combining data visualizations, narration, figures, etc. with analysis code, thereby fulfilling Knuth's vision of literate programming [Knu84]. However, a limitation of notebooks is that **interactive visualizations can typically not be used to manipulate data**. Schmidt and Ortner [SO20] discuss reasons for the lack of interactive data analysis in notebook-style environments and cite, among others, limited interaction capabilities native to the environment. Native visualization libraries, such as Matplotlib [Hun07], have only basic interactive capabilities, and cannot feed back actions from visualizations to code. Complex visualizations, such as custom tools or Tableau views, can be embedded in Jupyter notebooks, but typically cannot manipulate data in the notebooks. Libraries such as Altair [VGH*18] support interactive visualizations, but the interactions primarily serve the purpose of coordinating between multiple views, not for data transformation. Observable notebooks provide a platform for implementing complex interactive visualizations, including native manipulation of data by visualizations, however, they lack the ability to create workflows based on user interactions and reuse them.

An interesting approach for preserving workflows generated through an interactive visualization is B2 [WHS20], which is a set of techniques that treat the data queries as a shared abstraction between code and visualizations. The shared abstraction in B2 uses the notebook cells to track the interactions as query predicates, which depend on expressiveness of the language. We discuss B2 and other related techniques in more detail in Section 5.

3. Capturing and Reusing Workflows

We propose an approach by which analysts curate their workflow from the provenance of their analysis sessions, rather than explicitly modelling the analysis workflow either in a graphical workflow editor or in code. This way, analysts can freely work and explore until they have achieved a result that they think is worth saving in a workflow. Only when an interesting state is reached, can they extract the relevant steps to preserve it as a workflow, thereby minimizing the required overhead to the analyst and encouraging open exploration. In the rest of this section, we describe how we can capture provenance in a reusable manner and use it to curate reusable workflows.

3.1. Capturing Interaction Provenance

Usually, capturing analytic provenance involves tracking the low-level mouse, pointer, or keyboard events by the analyst. Such events are well-suited for implementing features like undo/redo or logging the analysts' activity. However, such low-level events lack the information necessary to recreate the interaction. We propose capturing the interactions as an abstraction rather than as events. The abstraction captures all the information that is necessary to recreate the interaction. In our specification, we capture the interactions that are relevant for data manipulation, namely *view specification*, *selections*, and various types of *data transformations* [HS12].

View specification interactions are concerned with choosing the subset of dimensions. An example is to show two dimensions of a dataset in a scatterplot: here, the view specification entails both the choice of dimensions and the choice of visualization technique.

Selections are a basic but very important interaction available in visualizations. Selections cannot only be used to highlight items of interest, but also form the basis for further data transformation on selected subsets of the data. For our specification, we break down types of selections by the level of semantics they capture:

The simplest form of selection is *ID-based selection*, which directly stores the IDs of the selected items. ID-based selection has the lowest level of semantics and is the least useful when reusing a selection. When items are added in a new version of a dataset, they are not considered, even if they clearly fall into a selected pattern.

Next, we specify the *range selection* that stores ranges over dimensions, capturing a set of rules for a selection, similar to e.g., an SQL query predicate. Range selections are usually specified using rectangular brushes [BC87, MW95] in scatterplots, or a series of brushes along an axis in parallel coordinates. They are reusable for updates in the data as long as the updates happen within the extent of the range selection. For example, if an updated version of the dataset has three new points within a rectangular brush area, these points will be selected automatically.

The next level of selection is *semantic selection* as introduced in our previous work [GGC*21]. This approach captures higher level semantics behind the selection that are apparent when the data is visualized. A pattern-based selection recognizes, for example, that an analyst selected all outliers, or a cluster centered at a specific location. By selecting based on higher level patterns in the data, such selections are robust to changes in the data. For example, when outliers are selected in a dataset, similar outliers can be selected in an updated dataset, even if the outliers appear in new locations. We could also use other approaches, such as query relaxation proposed by Heer et al. [HAW08]. The aim of semantic selection is to capture complex selection patterns rather than a list of IDs or simple rules. Such selections are the most reusable selections, as they are robust to fairly complex updates to the dataset.

Our specification supports four **data transformation** actions: *Filter* actions track whether certain items are filtered-in or filtered-out of the dataset. Filters are useful for focusing on a select group of items by filtering them in or for removing irrelevant data items by filtering them out. *Labeling* sets of items in a dataset is useful for annotating or tagging items with metadata or observations. *Categorize* actions are used to classify items, or assign them to categories, from a set of dynamically defined categories. Usually, each point is assigned to a unique category. Categorization is useful for dividing the dataset into distinct subsets that can then be compared or used separately in subsequent analysis steps. Deriving new data items by *aggregating* groups of existing ones is an important data transformation, as is evident from the popularity of pivot tables in Excel. An aggregate item can replace the items it was derived from, simplifying the data. Aggregation also allows analysts to compare groups with shared characteristics effectively. Aggregation is done by grouping multiple data items into a new item. Attributes are aggregated based on a mathematical function (often called an "apply" function in programming libraries). These functions usually summarize multiple values into a single representative value like the *mean*, *median*, *sum*, *min*, or *max* of the item's attribute, but custom functions are also common. In a dataset on metrics about countries, for example, it might be useful to aggregate all European countries into a single "Europe" item, and compare it to an "Asia" item. For this aggregation, we have to define "the population" of Europe as the sum of the population of all the countries in it. A column like "life expectancy", however, would need a different, more sophisticated function for meaningful aggregation.

We have chosen these interactions as they allow us to demonstrate our approach, yet other operations, such as deriving attributes, or manipulation, such as moving around items, could be supported by our methods. Together, these view specification, selection, and data transformation actions make up a powerful set of tools that benefit strongly from being available in an interactive visualization interface and are commonly used in data science tasks.

3.2. Capturing Analysis Process

We propose capturing the analysis session in a provenance graph where each node in the graph represents an interaction. A provenance graph records the interactions in the sequence they were performed, allowing us to infer the dependency of the interactions on each other. Provenance graphs are typically directed acyclic graphs. Downstream interactions can depend only on upstream interactions. Data transformations are derived from selections: to cate-

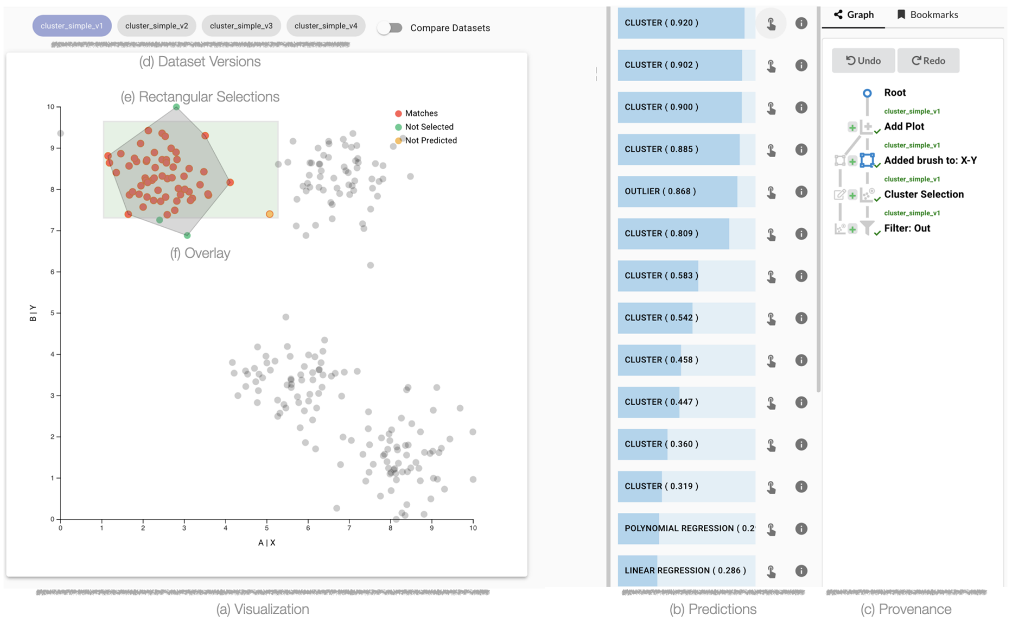


Figure 2: *A dataset exhibiting clusters is shown in a scatterplot (a). (e) A rectangular brush selection is used to compute predictions for patterns to capture the semantics of the selection. (b) A ranked list of these predictions is shown to the right of the scatterplot. The analyst selects the top prediction, which is a cluster, and the system shows an overlay (f) to visualize the boundary of the cluster. (c) The provenance graph on the right shows the captured interactions.*

gorize a group of points, for example, they are first selected and then assigned to a category. The robustness of the transformation depends on the type of selections used, as described previously. For example, if an analyst selects 15 items individually and assigns them to a category, and the category is associated with just those items. However, if the items were selected using a range selection, the category is associated with the range rather than individual items. When the dataset is updated, items appearing in the region of the range selection are categorized as well. Using semantic selections further improves the robustness of subsequent actions. An analyst can use the pattern-based selection to refine the initial selection of 15 items as a cluster. The category is then associated with the cluster rather than the original selection. Updating the dataset by adding or removing items automatically updates the categorization as well. If the groups of items were to move, the semantic selection would still accurately track the items, in contrast to a range selection, which would loose items that move outside the range.

Analysis sessions are rarely linear, and usually involve trial and error. Multiple attempts can be achieved in the form of analyst going back a few steps in the analysis provenance by undoing actions and starting a different analysis flow. Alternatively, an analyst might decide to clear all their current interactions and go back to an initial state, before trying something else. A graph representation allows us to capture the former type of iterations as parallel branches in the provenance, whereas the latter are in a sequence.

An analyst can also add metadata to each node in the provenance graph. The metadata can be be in form of annotations, where the analyst tries to externalize their knowledge such as assumptions about the data, known errors in the data, etc. which are typically not possible to capture automatically. In large analysis provenances, especially ones with long analysis sequences and multiple branching analysis, the analyst can also bookmark certain nodes as points of interest. Using annotations and bookmark simplifies extracting workflow after an analysis goal is achieved.

Capturing the abstract interactions enables editing of the analysis session to curate workflows as well as reuse the sessions on the same dataset or, more interestingly, on an updated dataset. We capture the analysis process as an abstraction of the interactions, which makes the captured analysis agnostic to the environment.

3.3. Curating Workflows

As we discussed in the previous section, the captured analysis provenance can be cluttered by the iterative nature of the analysis process. Hence, extracting and cleaning a particular analysis branch or a subsequence of the analysis as a reusable workflow is desirable.

We previously introduced different levels of selections that can be captured. These different types of selections can — and commonly are — combined. An analyst can start with a rough selection of points either by specifying the points directly or specifying a range. They can modify this selection and add or remove points to it. They can then decide to use semantic selection to refine their existing selection. A provenance graph will be populated with multiple nodes as the analyst iterates over different points before settling on a pattern. While curating the workflow, an analyst can easily remove superfluous selection attempts, keeping only the most informative selection to drive downstream interactions (see Figure 1), thereby creating a succinct reusable workflow.

It is also possible to automatically prune workflows, to remove actions that do not have an impact on the eventual result. For example, when a "clear selection" action is detected as part of a filter

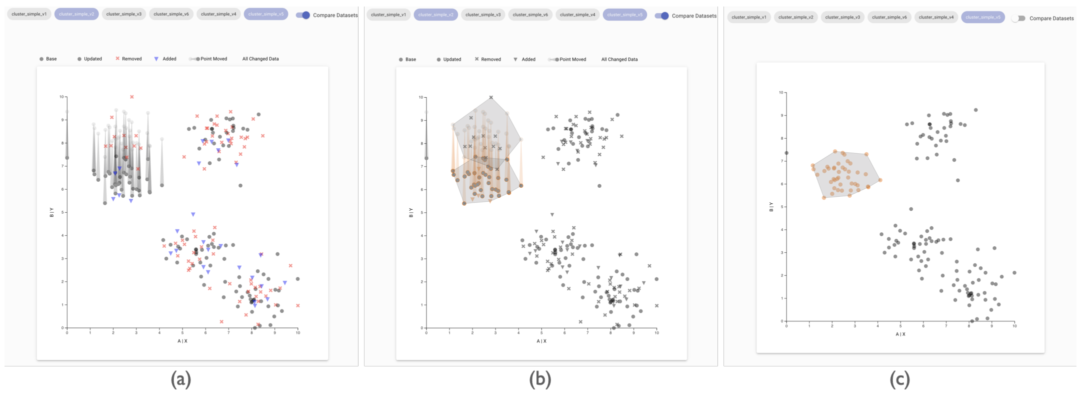


Figure 3: *Comparing two datasets and reapplying a workflow. (a) The comparison mode explicitly shows the differences between two selected versions of a dataset. The scatterplot encodes newly added items in the updated dataset as blue triangles ▼, removed items are shown as red crosses ✖, and items that have shifted positions show a comet-like trail ●from their original to their new position. (b) The selection made on the original dataset moves down to the new cluster and handles new and removed items correctly. (c) The updated selection.*

sequence, all previous selections and the clear selection operation could be automatically removed.

3.4. Reusing Workflows

Capturing the analysis sessions and workflows in a reusable manner makes it easy for an analyst to rerun the analysis on the same dataset (for reproducibility) and, more importantly, apply the analysis to an updated version of the dataset (for reusability).

Tabular datasets can change in a limited number of ways: attributes associated with rows can change, and rows can be added or removed. Also, dimensions or rows (items) can be added, removed, or reordered [NSH*17]. For our purposes, we limit ourselves to changes of existing attributes, adding or removing items, and updating attribute values, as order is relevant only for certain representations and adding or removing of dimensions is beyond the scope of our work.

We have different approaches to reapply selections, which is straightforward for IDs and ranges. The method for reapplying semantically captured selections varies by the type of selection. If we capture pattern-based intent [GGC*21], we can rerun the appropriate algorithm along with captured parameters to recreate the selection. Of these three types of selections, the ID-based selections are the least useful for reusing a workflow on an updated datasets. Range-based selection perform better; however, they fail to capture updates to the dataset that happen outside the range. Semantic selections have the potential to be robust to updates in the dataset.

However, even with semantic selections, automatically reapplying the analysis session or a workflow to an updated dataset might not always make sense. The dataset changes might be drastic enough that a previous analysis session is not appropriate anymore. Hence, it is important to give analysts the ability to review the captured sessions for different version of dataset and mark whether different interactions are valid for those versions.

3.5. Bridging between Environments

The environment-agnostic nature of our approach allows us to integrate workflows with computational analysis systems and reexecute a workflow on an updated dataset just like a function. The abstract representation of the analysis can be expressed in any of commonly used data interchange formats like JSON, YAML, etc. We can build visual analysis tools that can capture the analysis sessions in one of the formats, and companion libraries to use the captured analysis sessions in other environments like notebooks.

4. Reusing Workflows Prototype

To demonstrate the feasibility of our approach for capturing and reapplying workflows, we developed prototype tooling to demonstrate all aspects of the approach. In an interactive visualization tool, we demonstrate how we can capture the analysis process and reapply to updated datasets in the same tool. The prototype is available at `https://reapply-workflows.github.io/reapply-workflows/`. We also implemented a Python library that can load captured workflows and that can be used in computational environments like Jupyter notebooks.

4.1. Interactive Visualization Tool

The interactive visualization tool allows analysts to create projects and upload different versions of a dataset. Analysts may select dimensions of the tabular dataset to visualize in one or multiple **scatterplot(s)** (Figure 2(a)). We provide rectangular and free-form "paint-brushes" of three sizes for selection. Analysts may also add a **parallel coordinate plot** to visualize multiple dimensions at once, which supports brushing on the axis. Please see Supplementary Section 3 for a screenshot of the complete prototype.

To capture the analyst interactions, we use the Trrack library [CGL20] we previously developed and store the actions in a directed acyclic provenance graph. Each node in the provenance graph is the abstract representation of a corresponding interaction the analyst makes. We visualize the provenance graph in a tree-like layout (Figure 2(c)), where each action is described and can be annotated by the user. Analysts can go back in the provenance graph to a previous step and start off a new branch, which supports the iterative nature of visual analysis process.

We use techniques from our previous work [GGC*21] to cap-

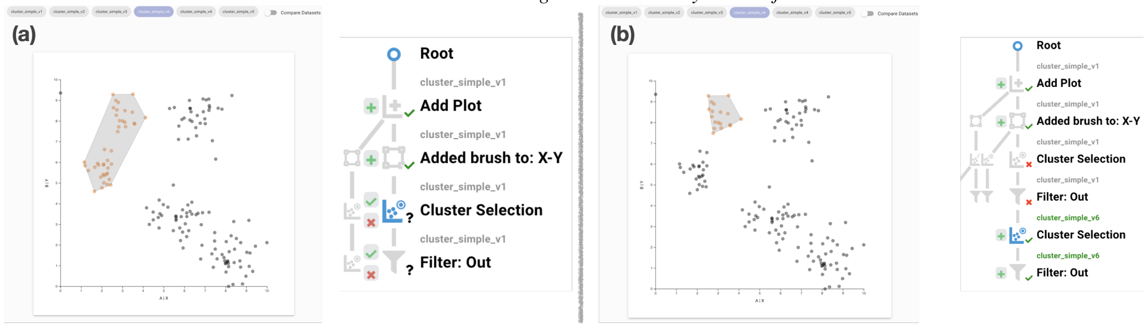


Figure 4: *Reviewing and updating workflows that were applied to an updated dataset. Continuing the analysis from Figure 2, the analyst loaded a new version of the dataset where a part of a cluster broke off and moved down. (a) By default, the system considers these two clusters to be one larger cluster as the previously selected larger cluster biases the outcome toward a single cluster. The review interface indicates that certain actions have not been reviewed for this version of the dataset, by showing a question mark (?) next to the node. To select just the upper cluster, the analyst first confirms the "Add Plot" and "Added Brush" actions, which are then shown as approved with a check-mark (✓). (b) The analyst then rejects the "Cluster Selection" action and picks a new cluster prediction that captures their intent.*

ture semantically rich pattern-based intents. Our prototype monitors user-selections in any plot and compares them to a large set of patterns computed for a given dataset using various algorithms and parameterizations. The different patterns are ranked based on the Jaccard Similarity between the selection and the prediction, as shown in Figure 2(b). In Figure 2, we see a rectangular selection that partially covers a cluster, and the system ranks a clustering pattern as a good match. When an analyst hovers over the cluster prediction, the extent of the cluster is shown as a polygon, and the items that are not part of the selection are highlighted. When an analyst chooses to confirm this prediction as the intended pattern, our system stores the details of this pattern.

We predict five different patterns: clusters, inliers and outliers, correlations, multivariate optimization, and ranges. For each of these patterns, we store the information necessary to recreate the pattern in an updated dataset. For example, for clustering, we store which type of algorithm was used (e.g., KMenas or DBScan) with which parameters, in addition to attributes about the specific cluster that is selected, such as its centroid. Figure 2 shows an example where an analyst first added a plot of a dataset that exhibits clusters. The analyst then continued with a crude rectangular selection. The system recommends a cluster as a match in the prediction interface. Hovering over that cluster prediction reveals the cluster's properties. The analyst decides this is a good match for the intended selection and confirms the prediction. Finally, the analyst filters out the selected items. Each of these steps is then reflected in the provenance graph. The analyst could now go on and continue with subsequent analysis steps and only create a robust workflow based on these steps at a later time.

4.1.1. Comparing and Updating Datasets

Our systems explicitly visualizes differences between different versions of a dataset and how a current analysis would be applied to different versions. Figure 3 shows a comparison between the dataset shown in Figure 2 and an updated dataset, and a subsequent successful application of a cluster-based selection. We see in Figure 3(a) how the dataset changed compared to the one in Figure 2. Figure 3(b) shows how the selected cluster changed between the datasets; the hulls of both clusters are shown. Figure 3(c) shows the cluster selection in the updated dataset. If the initial selection had been captured using range selection, i.e., not using a semantic pattern, the items that shift outside the range would not have been captured.

Figure 4 shows another version of the dataset, where the selected and subsequently filtered cluster broke apart into two clusters. Depending on the intent of the analyst, several options are plausible: remove both clusters, remove only the top or bottom cluster, or remove none of the clusters. Here, an automatic determination is impossible, as the right action depends on the analyst's higher level intent. Hence, the analyst has to review this situation and make a decision on how to proceed, as illustrated in Figure 4. By default, the system assumes that one large cluster is the best match (Figure 4(a)), as it best corresponds to the previously selected cluster overall. Assuming that the analyst intends to select only the top, smaller cluster, they can reject the "Clusters Selection" node in the provenance graph, and replace it with a better matching cluster, as shown in Figure 4(b).

4.1.2. Curating Workflows

Finally, the analysts can curate a reusable workflow based on the provenance data (see Figure 5). After switching to the appropriate analysis branch, the analyst can open the workflow editor, and create a new workflow from the current branch. Doing so loads the current branch as an editable workflow. The analyst can remove individual nodes, name the workflow, and sync it to the workflow database. While theoretically possible, our current implementation does not support automatic pruning. The workflows stored in the workflow database are available for reuse in the tool as well as the Python library (see subsection 4.2).

4.2. Bridging to Computational Notebooks

At the heart of our suite of tools is the **Reapply Workflows** library that performs all the predictions and the matching of actions between updated datasets. The library is used in our prototype tool but can equally be used by third parties, e.g., to load workflows in notebooks. The piece that connects the visualization tool and the computational environment is a **workflow database**. An analyst can work in the visualization tool to perform a visual analysis and capture workflows, as discussed before. Whenever a workflow is

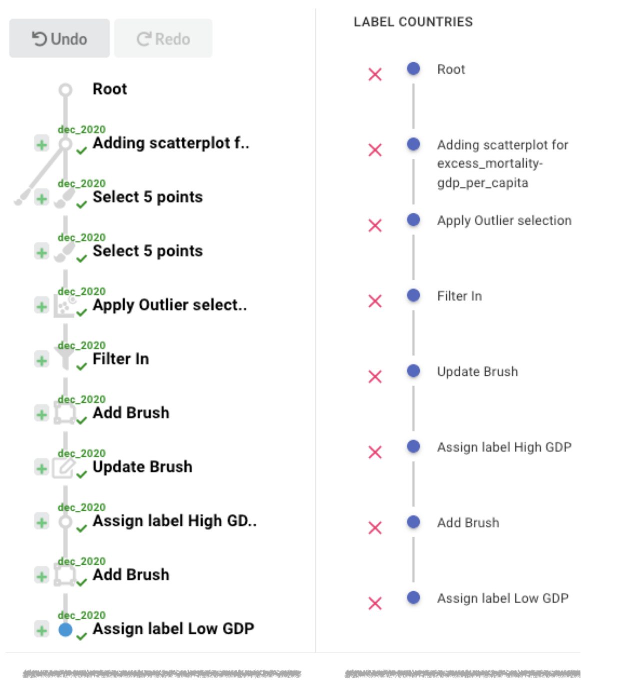


Figure 5: *The workflow editor. An analyst can create a new workflow from the interaction history captured in (a), the provenance graph, and curate it in (b), the workflow editor interface by removing unnecessary actions.*

created or modified, it is also stored in the workflow database. The workflows can then be loaded by the library from the database to a computational environment, such as a Jupyter notebook.

Figure 6 shows the process of loading and using a workflow in a notebook. The library interfaces with the workflow database to provide convenient access, printing descriptions, and an inspection of the steps in a workflow. Ultimately, workflows can be applied to a pandas dataframe. The output of applying the workflow depends on the actions in the workflow. If the workflow results in a selection, the output dataframe has an extra boolean column *isSelected* that denotes the selected items. More complex workflows with data transformation make modifications to the output dataframe, e.g., filters return a subset of the original dataset after executing the filter, labeling and categorization operations result in an extra column with the relevant label or category assignment, and aggregation workflows add a new row to the dataset with the aggregated values. Section 2 in the supplementary material shows the flow of information between our prototype, the library, and the workflow database graphically.

4.3. Implementation

The prototype is a web-based application developed with React and TypeScript. The backend is a Flask server that leverages the Reapply workflow library for computations and workflow-related features. The source code can be found at `https://github.com/visdesignlab/reusing-intent`. The library is written in Python and uses scikit-learn to run the prediction algorithms. The library is available on the TestPyPi package index by the name *reapply-workflows*. We show our computational demos in Google Colab notebooks, which are equivalent to Jupyter notebooks but can be collaboratively edited and are hosted by Google.

We store datasets in a SQL database. Different versions of the dataset are tracked with a separate *record* table. The changes between the dataset are computed on the fly, and hence no additional storage is needed for keeping track of diffs. We pre-compute patterns used by the prediction system for pairs of dimensions to help speed up the initial predictions. We switch to on-the-fly computations when any data transformation changes the dataset. The analysis sessions and the curated workflows are stored in Google Firebase Realtime database as JSON.

5. Comparison with Alternative Approaches

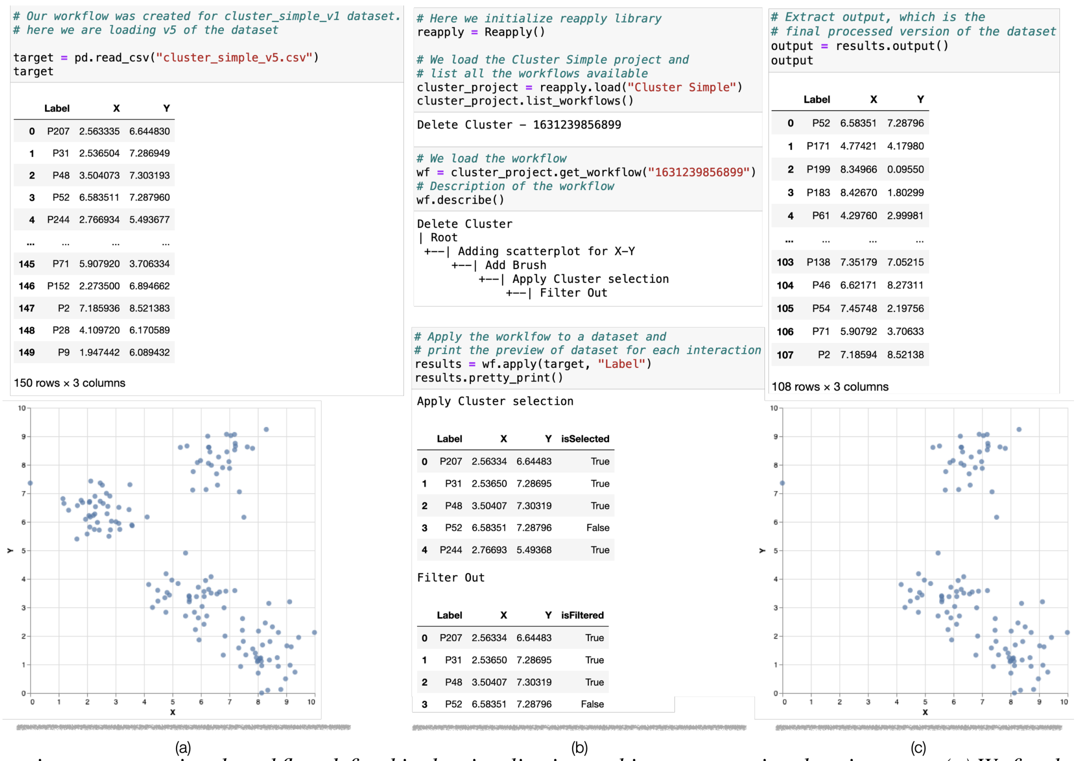
In this section, we compare our approach with Tableau Prep, B2 [WHS20] and VisFlow [YS17]. All these systems have in common that they capture workflows, yet they all use different approaches.

Tableau Prep and VisFlow use **explicit modeling** of workflows. Tableau Prep provides a graphical workflow editor, where analysts can drag and drop nodes as steps of a workflow. The visualizations in Tableau Prep are limited to distributions although data can be imported into Tableau subsequently, which limits the possibility of open exploration before workflow generation. VisFlow is a graphical workflow editor similar to Tableau Prep, but supports adding visualizations as a part of workflow nodes. Both VisFlow and Tableau Prep do not support exporting the generated workflows outside their environments. The workflows in Tableau Prep and VisFlow do not support semantic interactions and rely on rules for selecting the data. Further, both tools include the dataset as part of their workflow, making reapplying workflows difficult. A downside of such explicit workflow modeling systems is that they are more akin to graphical programming than to open exploration and refinement of datasets, which comes with the usual burdens of programming: high complexity and a steep learning curve.

B2 [WHS20] is a Jupyter extension that aims to bridge the gap between interactive visualizations and computational environments. The strength of the B2 approach is that it integrates the interactive visualizations directly in the notebook and provides tight coupling between code blocks and the visualizations. B2 currently supports selections; any further data transforms require coding. Further, the selections are limited to brushing and do not capture semantics behind the selection. B2 supports limited provenance tracking for interactive selections by generating code snippets to reflect the visual selection. The code snippets are timestamped to keep track of the order. Older snippets are automatically commented-out, which adds clutter to the code blocks. The lack of explicit tracking makes it difficult to maintain parallel data analysis approaches. It would be possible to combine our approach of semantically capturing selections and provenance tracking with B2's tight integration of code and interactive visualization.

6. Validation

We use three strategies to validate our approach: usage scenarios, demonstrating the usefulness of our techniques in a realistic scenario (see the following section and Supplementary Section 2);



```
# Our workflow was created for cluster_simple_v1 dataset.
# here we are loading v5 of the dataset

target = pd.read_csv("cluster_simple_v5.csv")
target
```

	Label	X	Y
0	P207	2.563335	6.644830
1	P31	2.536504	7.286949
2	P48	3.504073	7.303193
3	P52	6.583511	7.287960
4	P244	2.766934	5.493677
...	...	...	...
145	P71	5.907920	3.706334
146	P152	2.273500	6.894662
147	P2	7.185936	8.521383
148	P28	4.109720	6.170589
149	P9	1.947442	6.089432

150 rows × 3 columns

```
# Here we initialize reapply library
reapply = Reapply()

# We load the Cluster Simple project and
# list all the workflows available
cluster_project = reapply.load("Cluster Simple")
cluster_project.list_workflows()
```

```
Delete Cluster - 1631239856899
```

```
# We load the workflow
wf = cluster_project.get_workflow("1631239856899")
# Description of the workflow
wf.describe()
```

```
Delete Cluster
| Root
 +--| Adding scatterplot for X-Y
     +--| Add Brush
         +--| Apply Cluster selection
             +--| Filter Out
```

```
# Apply the worklfow to a dataset and
# print the preview of dataset for each interaction
results = wf.apply(target, "Label")
results.pretty_print()
```

```
Apply Cluster selection
```

	Label	X	Y	isSelected
0	P207	2.56334	6.64483	True
1	P31	2.53650	7.28695	True
2	P48	3.50407	7.30319	True
3	P52	6.58351	7.28796	False
4	P244	2.76693	5.49368	True

```
Filter Out
```

	Label	X	Y	isFiltered
0	P207	2.56334	6.64483	True
1	P31	2.53650	7.28695	True
2	P48	3.50407	7.30319	True
3	P52	6.58351	7.28796	False

```
# Extract output, which is the
# final processed version of the dataset
output = results.output()
output
```

	Label	X	Y
0	P52	6.58351	7.28796
1	P171	4.77421	4.17980
2	P199	8.34966	0.09550
3	P183	8.42670	1.80299
4	P61	4.29760	2.99981
...	...	...	...
103	P138	7.35179	7.05215
104	P46	6.62171	8.27311
105	P54	7.45748	2.19756
106	P71	5.90792	3.70633
107	P2	7.18594	8.52138

108 rows × 3 columns

Figure 6: *Executing a computational workflow defined in the visualization tool in a computational environment. (a) We first load the dataset. (b) We load the workflow library and the workflow we are interested in. We then apply the workflow to the dataset. The tool plots a preview of the actions. Note that new* `isSelected` *and* `isFiltered` *Boolean column are introduced when a brush and filter are added in the preview. (c) After the filter is applied, the number of rows is reduced from 150 to 108. A visualization of the result shows the cluster was removed. Visit the the notebook.*

synthetic datasets to demonstrate the robustness of reusing analysis workflows (see Supplementary Section 1); and interviews with professional data analysts (Section 6.2).

6.1. Usage Scenario: Outlier Countries for COVID-19

We analyze outlier countries with respect to COVID-19 metrics. The dataset [RMR*20] includes various COVID-19 related metrics for multiple countries across the world. COVID-19 data attributes change frequently and are a good way to demonstrate our approach, since selections and conclusions must be robust to updates in data. Let us look at a scenario for which we want to investigate countries that have an aberrant trend in the number of new cases and number of new deaths related to COVID-19. We start with data for January 2021 and load a scatterplot for **new monthly cases** vs **new monthly deaths**. We immediately see that many countries are far away from the cluster of countries close to the origin. We then select a few of these countries using a paint brush selection. The system computes predictions and suggests an outlier-based selection (see Figure 7(a)). We use this suggestion to refine our selection. We switch to different months of the dataset to see if the selection is applied correctly. We are happy with the selection, so we filter-in these items to focus on these outliers.

We then categorize the outliers. We select all the countries with high monthly cases and high monthly deaths with a rectangular brush and categorize them as countries with “High Deaths–High Cases”. We then select countries with low monthly cases but high monthly dates and categorize them as countries with “High Deaths–Low Cases”, and proceed to “Low Death–High Cases” (Figure 7(b). We switch to different datasets and verify that the categorization is applied correctly. When we are satisfied with the result, we approve the interactions in the provenance history. Figure 7(c) shows an extreme example, June 2021, where cases and deaths in most countries are much lower, clustering close to the origin of the chart. Applying the categories (Figure 7(d)) results in several outliers being unassigned, hinting at the fact that even moderate COVID activity is an outlier in that version of the dataset.

After curating the provenance history in a **Categorize Outliers** workflow, we store it to the workflow database. We then move to a Jupyter notebook to load this workflow and analyze these newly categorized countries. We can create a histogram of the categorized countries stacked by the region to get an idea about how different regions were affected by COVID-19, where we see that high deaths have shifted to South American countries in June, which were barely affected in January, and that South American countries are predominantly in the High-Deaths–Low-Cases category.

6.2. Feedback Session with Data Practitioners

We evaluated our method through interviews with four data practitioners from different domains — nursing (P1), public health (P2), surgery (P3), and chemical engineering (P4) — who regularly do data analysis. We first interviewed them about their current data analysis process. We then introduced the participants to the prin-

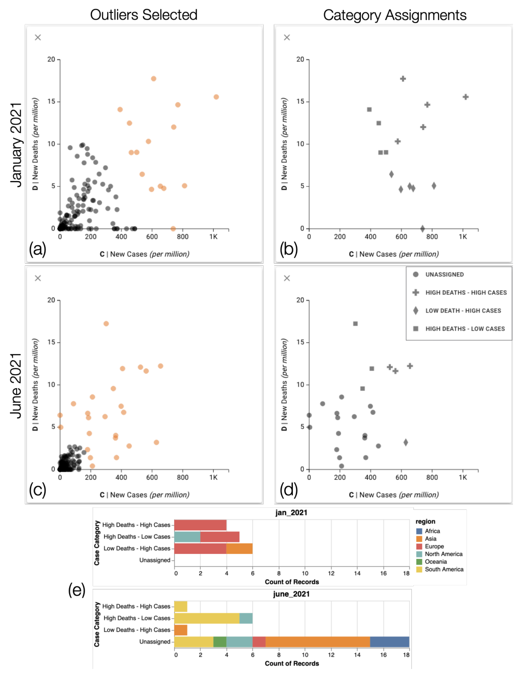


Figure 7: *Categorization of countries in a COVID-19 dataset. (a) A scatterplot for January 2021 shows the number of new cases vs. the number of deaths. We selected outliers, capturing countries with many cases, many deaths, or both. (b) We filter out all countries except the selected outliers, and categorize the countries. (c) We switch to an updated dataset for June 2021 and see that the number of cases and deaths have gone down across the world, but the pattern-based selection has correctly selected the outliers. (d) The system automatically applies the range-based categorization to the countries that fall within the previous ranges. (e) A subsequent analysis in a notebook reveals that the worst of the pandemic has shifted to South America. [Interactive Figure], [Notebook].*

ciples of our technique and gave them a live demo of the prototype tool and the Colab notebook, after which the participants were asked to give feedback on the techniques and speculate how they could be applied in their work. We have analyzed the transcript of the interviews and grouped the responses into themes, which we describe below. The interview questions and transcripts of the interviews are available as supplementary material.

Provenance Tracking Our participants used different tools to analyze their data, but all participants reported that they frequently explore alternative analysis approaches. Participants who use scripts report that they use comments and code blocks to keep track of the different analyses they do. One participant who used Tableau mentioned the limited utility of the undo/redo stack in keeping track of diversions in the analysis process. The participants particularly liked the provenance-tracking approach we demonstrate in our prototype tool. P1 said that *"I do like the, the branching piece from here, because it's visually a lot easier than to click the back button, or forward button, because it also is telling me a little label of what changed with that."* and P3 said that *"I definitely liked the way it branches. I think that's a super cool aspect of it. And then being able to kind of settle on one sort of branch analysis I'd be able to explore like, that is really powerful."*

Capturing Semantics Our participants expressed varying sentiments with regard to the semantic selection approach [GGC*21] we leverage in this paper. P3 said that *"Yeah, I think that's super smart. . . . they're saying, here's what I think is a cluster and then the program is saying, 'okay, looks like this is what you're trying to define. Is that correct? That would be really helpful."*. Although all participants acknowledged the usefulness of semantic selections, two participants, who have a strong statistics background, mentioned their hesitation in relying on the predictions without detailed information about the algorithms and the parameters used in the prediction — information our system captures but does not currently provide easy access to. P2 said that *". . . I wanted to understand what was happening in the background"*. P4, whose data analysis relies on segmentation of microscopy images, wanted the ability to add domain specific models to the prediction system.

Visual Data Wrangling Participants expressed interest in using a visualization to directly interact with the data for selection, creating groups, and labelling. P1, for example, said that *". . . you do regrouping in python, but then you always forget your variable name. And then you're always looking through your data frame for what is it type of thing, where Tableau, it's easy to do the groupings, and it just kind of makes a new variable right underneath it."*

Workflows All participants agreed that reusable workflows would be useful in their analysis. P4, for example, said *"I definitely think it will be applicable because most of the time, we actually don't inherently change the method itself. . . . So I definitely can see this to be helpful."* When asked about whether they would like to explicitly create workflows, or curate workflows after an analysis, the participants noted that their data analysis sessions often start with open-ended exploration of data to detect interesting patterns. P3 described their analysis process as *"definitely more exploratory."* Participants liked the ability to curate their workflows from an existing analysis session: *"I like this, because it's much more natural"* (P1).

Bridging Between Tools Participants were excited about the potential of using the workflows as a bridge between different tools they use. P1, who switches between different tools frequently, mentioned the need to repeat certain steps as they switch: *". . . replicating essentially a lot of the filters and the sorting . . . "*. The current approach of this participant usually involves modifying the data in one tool and then loading the modified data in a different tool. On demonstrating the use of workflows in the Python notebook, P1 said, *"great to be able to click on the Select 53 points, and then see the all code, you know, to that would do the 53 points."*

Collaboration with Domain Experts P2 and P3 work with clinicians and were interested in the applicability of our technique as a means for collaboration. They work in R and SAS heavily, whereas their clinical collaborators have no familiarity with scripting tools. P3 said that *"I work with a lot of clinicians, a lot of doctors, and they are interested in research, but they don't have much research background, right. And this would be something that I think would be really, really beneficial for them, because they are going to be, they're going to want to do a lot more kind of looking at the data and sort of touching the data, and it's going to be really important that they have that log where they can come back, give me data* *to just sort of reflect on."* and *"If they were able to show me their workflow, and I was able to go through and see, how it progressed, I think that would be really helpful"*.

Overall, our participants expressed positive sentiments regarding our techniques to capture reusable workflows and thought that the techniques would be useful in their current analysis environment. P1 said that, *"I think it's great. I would love love to see how you would implement this in Tableau"*. They were most excited about the provenance tracking and the ability to bridge between interactive visual analysis tools and computational environments.

7. Discussion

Generalization to Other Visualization Techniques and Data Types. Our technique is based on capturing interaction provenance as an abstraction that contains the information required to recreate the interaction (as opposed to a stream of mouse/keyboard events). We demonstrate what the abstraction looks like for common interactions such as selections, filtering, labeling, categorizing and aggregation. Our methods are transferable between different visualization techniques if they support equivalent interactions and datatypes. For example, to add a parallel coordinates plot, we did not have to modify our library developed initially with scatterplots. The interactions we describe are meant to be examples. Other types (e.g., selecting a neighborhood in a network or sorting a table) can be included if they are captured in the provenance graph and the library is extended to handle the type of operations and data. Our current implementation and our choice of algorithms are specific to tabular data. However, our general approach is applicable to network data, image data, or volumetric data, provided we can identify suitable methods for robust selections.

Certainty of Fit for Reuse. When we apply a selection to a new dataset, we currently assume that an analyst will review the update selection. Although a review is certainly necessary if the data changed significantly, minor changes might not require a manual review. We could conceivably compute metrics about how "sure" we are about a specific operation, as it is applied to a new dataset. If, for example, all points are in the same selection and have moved little, we might not need a review. If, in contrast, the dataset has changed significantly and the selection is affected, we could print a warning, emphasizing the need for a review.

Interaction Directly in Notebooks. Visualization libraries such as Altair [VGH*18] and B2 [WHS20] have made interactive selections in visualizations within a Jupyter notebook possible. Papers like B2 [WHS20] explore this approach thoroughly. Conceptually, our technique can support actions in embedded visualizations; hence, we plan on extending our library, so that selections made within a notebook can also be autocompleted and extracted into a workflow. Although we expect that other aspects, such as compound actions and reviewing of workflows, are infeasible to integrate natively within a notebook, robust, pattern-based selections would enhance an analyst's ability to leverage the interactive capabilities of such simple visualizations. Having the interactive visualizations directly integrated in the notebooks would reduce friction of switching between multiple environments, while limiting the complexity of visualization approaches that are suitable.

Reapplying to Unrelated Datasets. Our methods for transferring actions to updated datasets are robust up to a point. The clustering case in Figure 4 shows situations in which the automatic transfer does not succeed: when a pattern changes so much that a different interpretation is possible. While we remedy these situations through our review process, it would be worthwhile to automatically make alternative suggestions on which actions could be taken. Our current technique of capturing the workflow relies on tracking view specifications and downstream selections and transforms. Applying the dataset to unrelated dataset will almost always result in incorrect results. However, we see potential in using workflows as templates for recurring tasks, such as data cleanup on datasets generated by the same instrument although for different experiments. Here, a human analyst could update the parameters of a selection, or supplement it, but reuse a subsequent data transformation.

Alternate Ranking Strategies Our prototype uses the Jaccard similarity to rank different predictions. Jaccard similarity is sensitive to size of data and can be distorted if there is large variation between the sets being compared. Our technique can be extended to support alternative rankings, and we can potentially add a custom ranking approach tailored to a type of dataset. For example, we can modify the Jaccard metric by adding a regularizing parameter based on the size of the dataset to reduce the penalty for uneven set sizes.

8. Conclusion

We have introduced a method to capture interactive actions taken in a visualization in a semantically meaningful way and to reuse sequences of actions (workflows) on updated datasets. In this way, we make actions taken in a visualization just as robust to changes as if they were implemented in a function in code. We introduce methods that match up selections between updated datasets that go beyond just reapplying a simple rule, instead leveraging various pattern-detection algorithms and knowledge about the properties of a prior selection. We introduce a mechanism to review changes and update workflows if necessary. Finally, we have demonstrated that this approach also allows us to bridge between an interactive visualization system and a computational workflow.

Whereas robust workflows could be implemented in code or using graphical workflow modeling tools, we argue that our approach is easier to execute and allows for an unencumbered exploration process. Our prototype and our examples show that our approach works for a range of patterns and for datasets that change in significant ways. We believe that our approach could also be transferred to many other types of data and types of visualizations.

9. Acknowledgements

We wish to thank the reviewers for their feedback on our submission. We would also like to thank Derya Akbaba and Jack Wilburn from Visualization Design Lab for their help with feedback interviews. We gratefully acknowledge funding by the National Science Foundation (IIS 1751238).

References

[ABJ*04] ALTINTAS I., BERKLEY C., JAEGER E., JONES M., LUDASCHER B., MOCK S.: Kepler: An extensible system for design and execution of scientific workflows. In Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004. (June 2004), pp. 423–424. doi:10.1109/SSDM.2004.1311241. 2

[BC87] BECKER R. A., CLEVELAND W. S.: Brushing Scatterplots. Technometrics 29, 2 (1987), 127–142. doi:10.1080/00401706.1987.10488204. 3

[BCD*09] BERTHOLD M. R., CEBRON N., DILL F., GABRIEL T. R., KÖTTER T., MEINL T., OHL P., THIEL K., WISWEDEL B.: KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. SIGKDD Explor. Newsl. 11, 1 (Nov. 2009), 26–31. doi:10.1145/1656274.1656280. 2

[BCS*05] BAVOIL L., CALLAHAN S. P., SCHEIDEGGER C., VO H. T., CROSSNO P., SILVA C. T., FREIRE J.: VisTrails: Enabling Interactive Multiple-View Visualizations. In Proceedings of the IEEE Conference on Visualization (VIS '05) (2005), pp. 135–142. doi:10.1109/VISUAL.2005.1532788. 2

[CCSK19] CAMISETTY A., CHANDURKAR C., SUN M., KOOP D.: Enhancing Web-based Analytics Applications through Provenance. IEEE Transactions on Visualization and Computer Graphics 25, 1 (Jan. 2019), 131–141. doi:10.1109/TVCG.2018.2865039. 2

[CGL20] CUTLER Z. T., GADHAVE K., LEX A.: Trrack: A Library for Provenance Tracking in Web-Based Visualizations. In IEEE Visualization Conference (VIS) (Salt Lake City, UT, USA, 2020), IEEE, pp. 116–120. doi:10.1109/VIS47514.2020.00030. 2, 5

[CQW*14] CHEN Y. V., QIAN Z. C., WOODBURY R., DILL J., SHAW C. D.: Employing a Parametric Model for Analytic Provenance. ACM Transactions on Interactive Intelligent Systems 4, 1 (Apr. 2014), 6:1–6:32. doi:10.1145/2591510. 2

[DGST09] DEELMAN E., GANNON D., SHIELDS M., TAYLOR I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25, 5 (May 2009), 528–540. doi:10.1016/j.future.2008.06.012. 2

[DHRL*12] DUNNE C., HENRY RICHE N., LEE B., METOYER R., ROBERTSON G.: GraphTrail: Analyzing Large Multivariate, Heterogeneous Networks While Supporting Exploration History. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12) (2012), ACM, pp. 1663–1672. doi:10.1145/2207676.2208293. 2

[GGC*21] GADHAVE K., GÖRTLER J., CUTLER Z., NOBRE C., DEUSSEN O., MEYER M., PHILLIPS J. M., LEX A.: Predicting intent behind selections in scatterplot visualizations. Information Visualization 20, 4 (Oct. 2021), 207–228. doi:10.1177/14738716211038604. 3, 5, 9

[GLG*16] GRATZL S., LEX A., GEHLENBORG N., COSGROVE N., STREIT M.: From Visual Exploration to Storytelling and Back Again. Computer Graphics Forum (EuroVis '16) 35, 3 (2016), 491–500. doi:10.1111/cgf.12925. 2

[GNTT10] GOECKS J., NEKRUTENKO A., TAYLOR J., TEAM T. G.: Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol 11, 8 (2010), R86. 2

[HAW08] HEER J., AGRAWALA M., WILLETT W.: Generalized Selection via Interactive Query Relaxation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA, 2008), CHI '08, ACM, pp. 959–968. doi:10.1145/1357054.1357203. 3

[HMSA08] HEER J., MACKINLAY J., STOLTE C., AGRAWALA M.: Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. IEEE Transactions on Visualization and Computer Graphics (InfoVis '08) 14, 6 (2008), 1189–1196. doi:10.1109/TVCG.2008.137. 2

[HS12] HEER J., SHNEIDERMAN B.: Interactive dynamics for visual analysis. Communications of the ACM 55, 4 (2012), 45–54. doi:10.1145/2133806.2133821. 3

[Hun07] HUNTER J. D.: Matplotlib: A 2D Graphics Environment. Computing in Science Engineering 9, 3 (May 2007), 90–95. doi:10.1109/MCSE.2007.55. 2

[KNS04] KREUSELER M., NOCKE T., SCHUMANN H.: A History Mechanism for Visual Data Mining. In Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04) (2004), pp. 49–56. doi:10.1109/INFVIS.2004.2. 2

[Knu84] KNUTH D. E.: Literate Programming. The Computer Journal 27, 2 (Jan. 1984), 97–111. doi:10.1093/comjnl/27.2.97. 2

[Lam08] LAM H.: A Framework of Interaction Costs in Information Visualization. IEEE Transactions on Visualization and Computer Graphics 14, 6 (Nov. 2008), 1149–1156. doi:10.1109/TVCG.2008.109. 2

[MW95] MARTIN A. R., WARD M. O.: High Dimensional Brushing for Interactive Exploration of Multivariate Data. In Proceedings of the IEEE Conference on Visualization (Vis '95) (1995), IEEE Computer Society Press, pp. 271–278. doi:10.1109/VISUAL.1995.485139. 3

[NCE*11] NORTH C., CHANG R., ENDERT A., DOU W., MAY R., PIKE B., FINK G.: Analytic Provenance: Process+Interaction+Insight. In CHI '11 Extended Abstracts on Human Factors in Computing Systems (2011), CHI EA '11, pp. 33–36. doi:10.1145/1979742.1979570. 2

[NSH*17] NIEDERER C., STITZ H., HOURIEH R., GRASSINGER F., AIGNER W., STREIT M.: TACO: Visualizing Changes in Tables Over Time. IEEE Transactions on Visualization and Computer Graphics (InfoVis '17) 24, 1 (2017), 677–686. 5

[PJ95] PARKER S. G., JOHNSON C. R.: SCIRun: A Scientific Programming Environment for Computational Steering. In Proceedings of the ACM/IEEE Conference on Supercomputing (SC '95) (1995), ACM, p. 52. 2

[RMR*20] RITCHIE H., MATHIEU E., RODÉS-GUIRAO L., APPEL C., GIATTINO C., ORTIZ-OSPINA E., HASELL J., MACDONALD B., BELTEKIAN D., ROSER M.: Coronavirus Pandemic (COVID-19). Our World in Data (Mar. 2020). 8

[Shn83] SHNEIDERMAN B.: Direct Manipulation: A Step Beyond Programming Languages. Computer 16, 8 (Aug. 1983), 57–69. doi:10.1109/MC.1983.1654471. 2

[SO20] SCHMIDT J., ORTNER T.: Visualization in Notebook-Style Interfaces. In Proceedings of the Workshop on the Gap between Visualization Research and Visualization Software (VisGap) (May 2020). doi:10.2312/visgap.20201104. 2

[Sv08] SHRINIVASAN Y. B., VAN WIJK J. J.: Supporting the analytical reasoning process in information visualization. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2008), CHI '08, pp. 1237–1246. doi:10.1145/1357054.1357247. 2

[vdEvW13] VAN DEN ELZEN S., VAN WIJK J. J.: Small Multiples, Large Singles: A New Approach for Visual Data Exploration. Computer Graphics Forum (EuroVis '13) 32, 3pt2 (2013), 191–200. doi:10.1111/cgf.12106. 2

[VGH*18] VANDERPLAS J., GRANGER B. E., HEER J., MORITZ D., WONGSUPHASAWAT K., SATYANARAYAN A., LEES E., TIMOFEEV I., WELSH B., SIEVERT S.: Altair: Interactive statistical visualizations for Python. Journal of open source software 3, 32 (2018), 1057. 2, 10

[WHS20] WU Y., HELLERSTEIN J. M., SATYANARAYAN A.: B2: Bridging code and interactive visualization in computational notebooks. In ACM User Interface Software & Technology (UIST) (2020). 2, 7, 10

[XOW*20] XU K., OTTLEY A., WALCHSHOFER C., STREIT M., CHANG R., WENSKOVITCH J.: Survey on the Analysis of User Interactions and Visualization Provenance. Computer Graphics Forum 39, 3 (2020), 757–783. doi:10.1111/cgf.14035. 2

[YS17] YU B., SILVA C. T.: VisFlow - Web-based Visualization Framework for Tabular Data with a Subset Flow Model. IEEE Transactions on Visualization and Computer Graphics (InfoVis '16) 23, 1 (2017), 251–260. doi:10.1109/TVCG.2016.2598497. 2, 7

[ZSN*15] ZAMAN L., STUERZLINGER W., NEUGEBAUER C., WOODBURY R., ELKHALDI M., SHIREEN N. I., TERRY M. A.: GEM-NI: A System for Creating and Managing Alternatives In Generative Design. CHI (2015). doi:10.1145/2702123.2702398. 2