Reusing Interactive Analysis Workflows
Supplementary Material

Kiran Gadhave, Zach Cutler, Alexander Lex

Overview

The supplementary material contains 3 figures which support our submission. We will link to important links for our submission here:

- Live Demo: https://reapply-workflows.github.io/reapply-workflows/#/project
- Demo Notebook: https://colab.research.google.com/drive/1EpNqN1JuicsauzixhGAv_s9mjP5qijLj?usp=sharing
- Source Code: https://github.com/visdesignlab/reusing-intent/
- Python Library: https://test.pypi.org/project/reapply-workflows/. To install the library run:

  pip install -i https://test.pypi.org/simple/ reapply-workflows

- Interview Script: https://osf.io/d8vzt/
- Interview Transcript: https://osf.io/43qfa/

1 Other Patterns and Updated Datasets

In Figure S1, we demonstrate how our methods can be applied to different patterns, and how these patterns are adapted when a changed dataset is loaded. We show an original dataset and a pattern-based selection, a comparison of how the data changed, and finally how the selection was applied to the changed dataset. For outliers, we can see that the points that move closer to the center are excluded from the outlier selections, and the points that move away are added to the outliers. For the range selection, we see the system added and removed the points that fall outside the rules for the range selection. For multivariate optimization, the system updates the selection on the new dataset, and we see an updated Pareto frontier (shown as a staircase). For correlations, we show a linear regression line and threshold within which the points are counted as part of the correlation. The system updates the selection based on points moving in and out of the threshold.

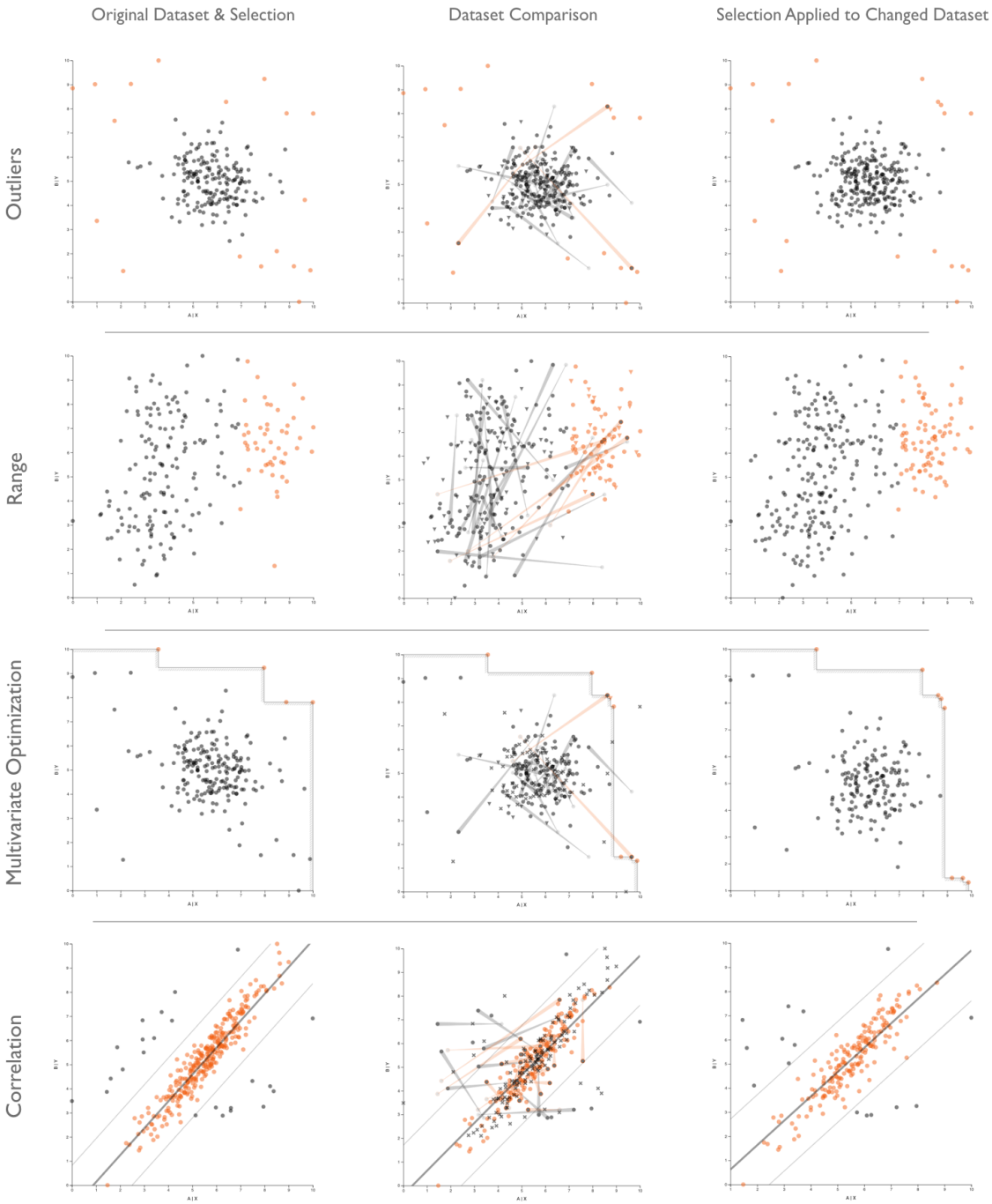

Figure S1: Semantic reapplication of different patterns in synthetic datasets. We see how different patterns (rows) are semantically applied to updated datasets. The first column is the original selection, the second column shows the compare view between the original dataset and the updated dataset, and the third column shows the final selections on the updated dataset.

2 Usage Scenario: Analyzing the Relationship of GDP and Child Mortality

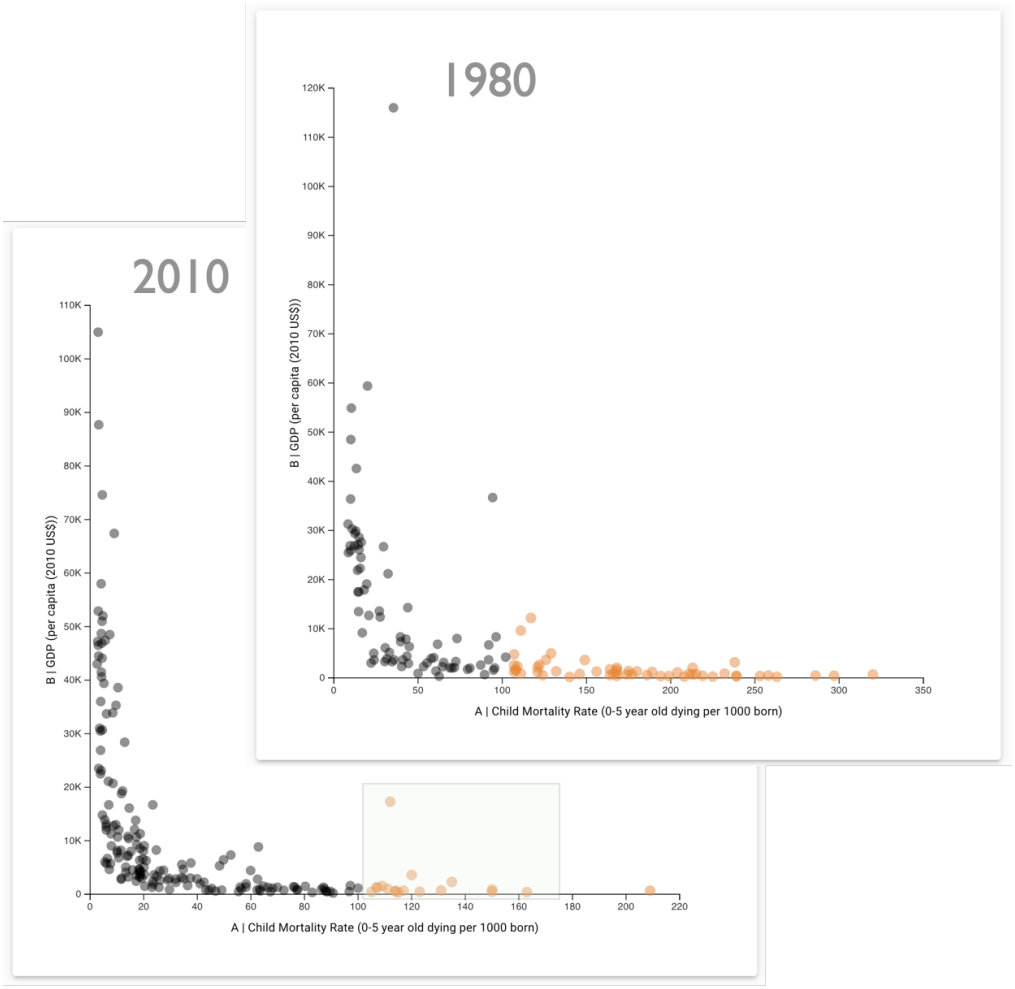

Figure S2: Preserving a selection in a temporally changing public health dataset. (a) We make a selection of countries with high child-mortality. We missed an extreme outlier on the far right, but the suggested simplified range selection catches that mistake. (b) Applying this brush to the 1980 dataset preserves the semantics of the actions, although now significantly more countries are selected (note the scale change).

For this example, we will use an annual global health dataset. We want to investigate countries that have high child mortality rates. We load in data from the Gapminder project [**?**] for the year 2010 into the visualization tool. We add a scatterplot of *child mortality rate* (CMR) vs *gross domestic product* (GDP), an indicator of a country's wealth. We immediately see a strong trend in the data: most countries with a high child mortality rate have low GDP, but there are also many countries with low GDP and low CMR. To focus on countries with high CMR, we use the rectangular brush for CMR values from 100–200. Our system suggests a simplified range selection to generalize the initial brush (see Figure S2(a)), which also includes Haiti — the rightmost point — with a value of 209.

We now decide to look at some historical data for CMR, and switch to the data for 1980. The system reapplies the interactions, and the generalized range selection includes the new countries with high CMR automatically (see Figure S2(b)). We are satisfied with the generalization and confirm the interactions in the provenance graph for the 1980 data (see Figure S2(c)).

We can now curate these interactions into a workflow, store it in the workflow database, and move over to a Jupyter notebook to continue the investigation to look for reasons of high child mortality. We can use this workflow to also automatically filter out the items of interest for other years in the dataset in the Jupyter notebook.

3 Interactions

In addition to selections/highlighting, our prototype also supports filter, label, categorize, and aggregate operations, illustrated in Figure S3.

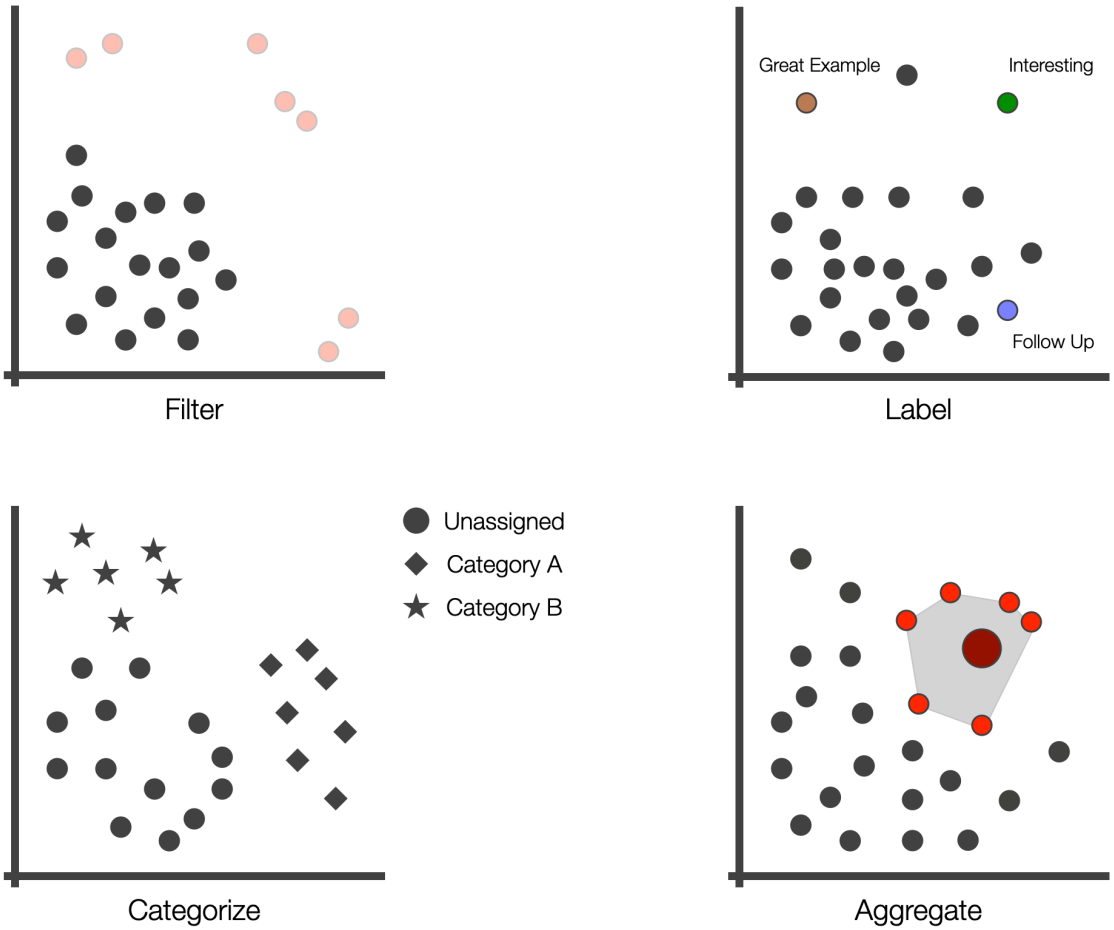

Figure S3: Overview of the data transformations supported by our system: **Filter** removes selected items, **Label** assigns a label to an item, **Categorize** classifies items into categories, and **Aggregate** reduces a set of items to one derived item (in this example, the light-red items are aggregated into the large dark-red item).

4 Analysis Flow

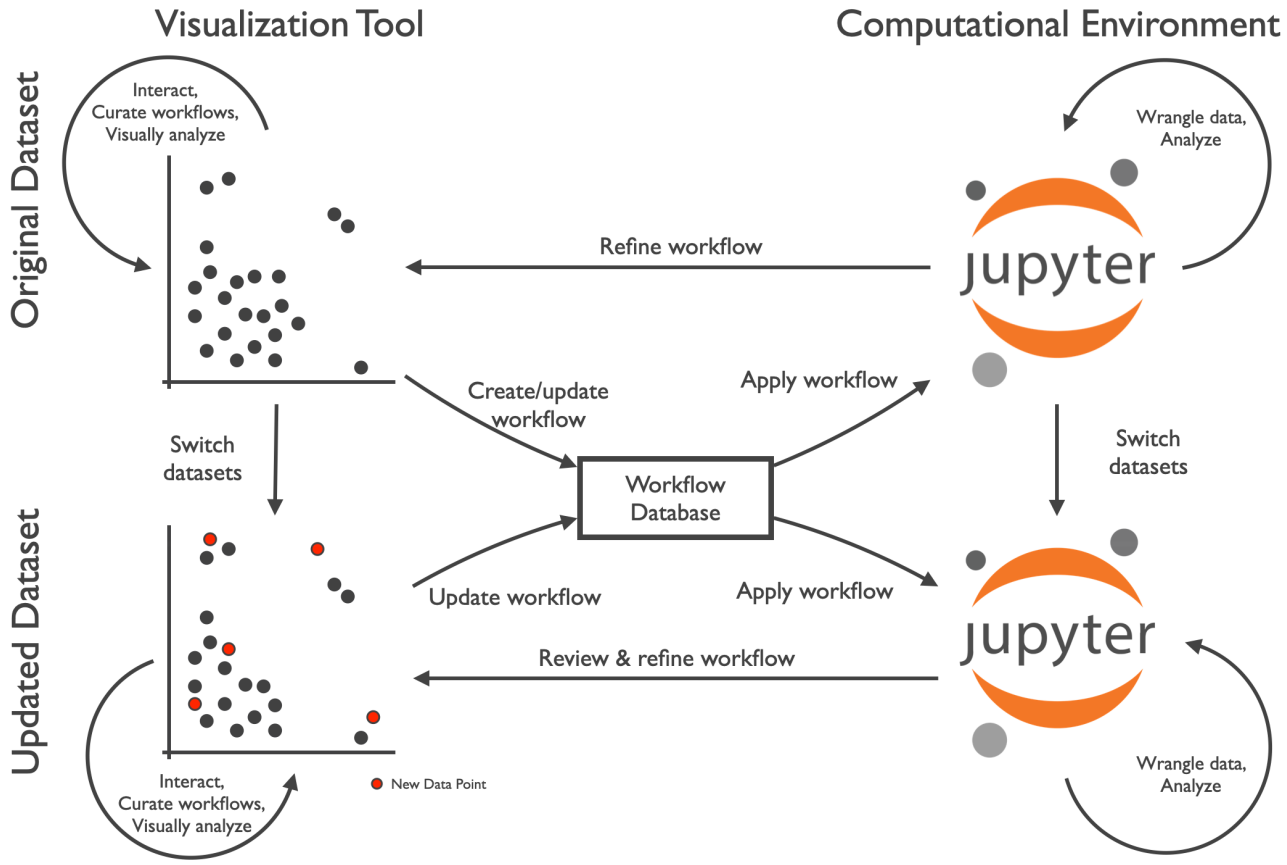

Figure S4: Analysis flow when bridging between a visualization tool, where workflows can be curated, and a computational environment, where workflows can be (re-)applied. Workflows created in the visualization tool are synced with a workflow database. Computational environments, such as a Jupyter notebook, have access to the workflows, and can apply it to the dataset by simply executing a function. When a dataset is updated in the computational environment, the workflow can be applied to the new datasets, with results immediately available. Analysts can also review and potentially refine the updated workflow in the visualization tool. If the workflow is updated in this way, it is then immediately available in the computational environment.

5 UI Prototype

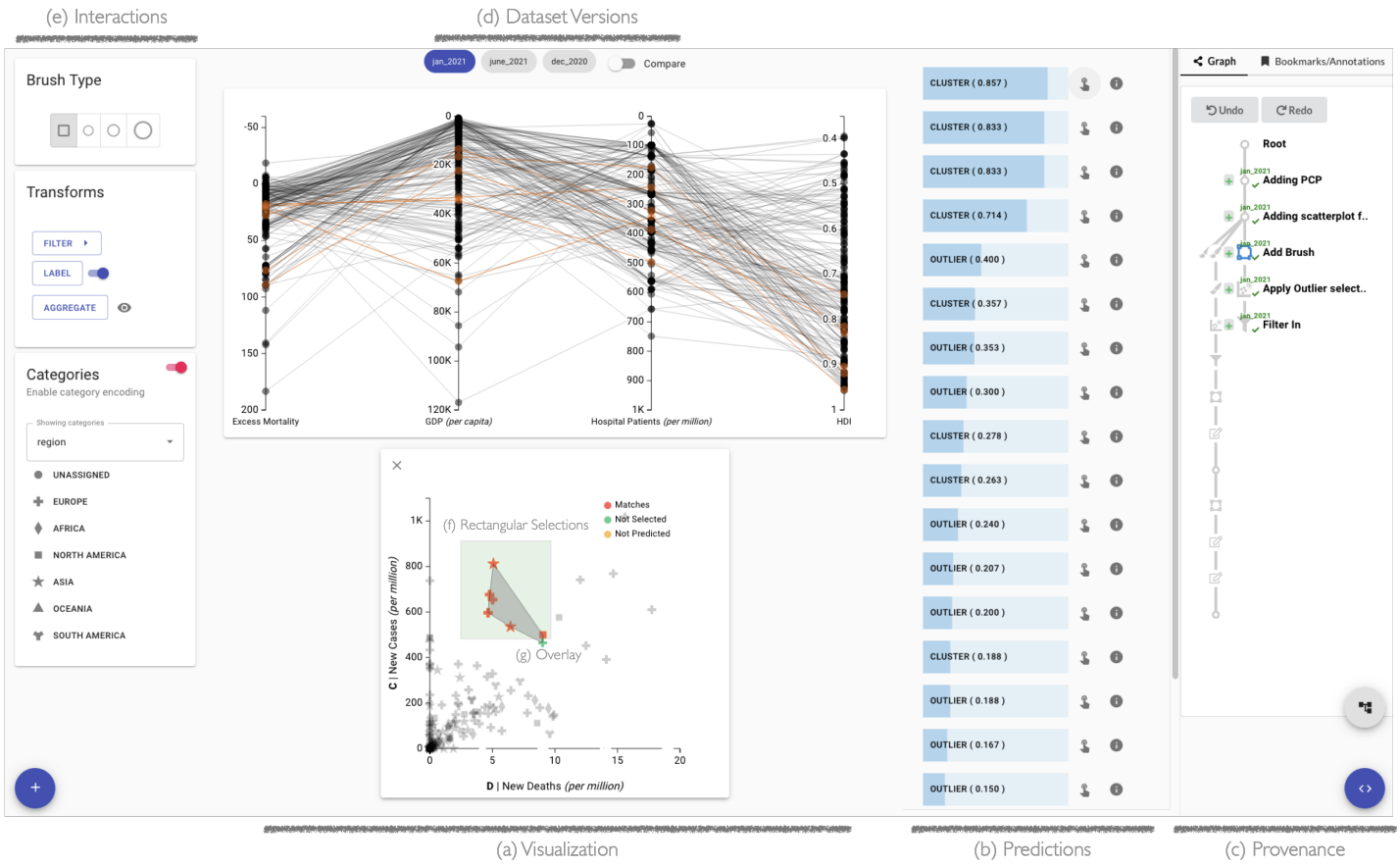

Figure S5: Prototype visualization showing a COVID-19 dataset. The analyst selected six dimensions to be shown in a scatterplot and a parallel coordinate plot (a). (f) A rectangular brush selection is used to compute predictions for patterns to capture the semantics of the selection. (b) A ranked list 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 (g) to show the the boundary of the cluster. (c) The provenance graph on the right shows the captured interactions.