A Appendix

This appendix includes material that was unable to fit into the main body of the paper. Below we describe the notations employed in our use case. Then, in Fig. 10 we show an annotated screenshot of our tool, NotaScope. Finally, in Fig. 7 we exemplify a minimum spanning tree (enabled by NotaScope) of the space implied by our compression distance metric.

A.1 Notations

The notations used in our case study were defined as follows:

- The ggplot2 notation consists of R code constrained by having only the `tidyverse` [82] package installed in a standard R runtime, biased towards using functions from that package rather than base R functionality wherever possible. The `tidyverse` package includes the ggplot2 visualization system [80] as well as the dplyr [83] data-transformation system. ggplot2 consists of a grammar based on Wilkinson's [84].
- The matplotlib notation consists of Python code constrained by having only the `matplotlib` [18] and `pandas` [34] packages installed in a standard Python runtime, biased towards using Matplotlib's axis-level function API rather than the legacy `pyplot` API, as recommended in the documentation. This is an imperative API with functions to progressively mutate a figure. matplotlib does not have any data-transformation capabilities, so pandas is used wherever necessary in this notation. matplotlib is the most-downloaded Python visualization system and Pandas is the most-downloaded data-frame system used to manipulate tidy data in Python.
- The pandas.plot notation is defined in the same way as the matplotlib notation, except that Pandas' built-in `.plot()` API is used wherever possible. This API is a thin wrapper around Matplotlib to "easily create decent-looking plots" [45] We include this notation as distinct from matplotlib due to the popularity of Pandas and to study the effects of a minor variant of a notation on the measures we have developed.
- The seaborn notation consists of Python code constrained by having only the `seaborn` [79] package installed in a standard Python runtime, biased towards using seaborn's figure-level API. Seaborn is itself a wrapper around Matplotlib and also depends on Pandas. It has some data-transformation functionality built-in and pandas' capabilities are used wherever necessary. seaborn's figure-level API is built around just three functions, for "distributional", "categorical" and "relational" figures. It is the most-downloaded statistical-graphics-focused library in Python.
- The seaborn.objects notation is defined in the same way as the seaborn notation, except that the new, work-in-progress object-level API is used wherever possible. The object-level interface was developed to be a more consistent and extensible interface than the figure-level one. Its grammar consists of Mark, Stat, Move, and Scale objects. We include this notation to study the differences between two notations that share many design decisions.
- The plotly.go notation consists of Python code constrained by having only the `plotly` [48] and `pandas` packages installed in a standard Python runtime, biased towards using Plotly's lower-level `graph_objects` interface. The `graph_objects` interface enables Python users to generate JSON figure descriptions to be rendered by the Plotly.js library, and includes almost no data-transformation features, so pandas' data-transformation capabilities are used. The structure of this API is oriented around the accumulation of trace objects which represent series to be drawn. Plotly is the second-most-downloaded visualization library in Python.
- The plotly.express notation is defined in the same way as the plotly.go notation except that the built-in high-level Plotly Express API is used whenever possible. Plotly Express was developed to enable the creation of terser specifications than is possible with the plotly.go notation, by including some data-transformation capabilities. Wherever those capabilities are insufficient to specify an example, pandas' are used in this notation. Plotly Express is included within the `plotly` package and was designed to have a similar relationship to the plotly.go notation as the seaborn's figure-level interface does to matplotlib.
- The Vega-Lite notation consists of JSON code constrained by having only the `vega-lite` [63] module installed in a standard NodeJS runtime. Vega-Lite is based on a highly consistent and orthogonal grammar, with built-in data-transformation capabilities.
- The Altair notation is defined by having only the `altair` [73] and `pandas` packages installed in a standard Python runtime. Altair is a Python interface to Vega-Lite.

Fig. 10: To conduct our analysis we built a tool called NotaScope, which supports a wide variety of mechanisms for comparing notations. See the video figure for a walk-through.

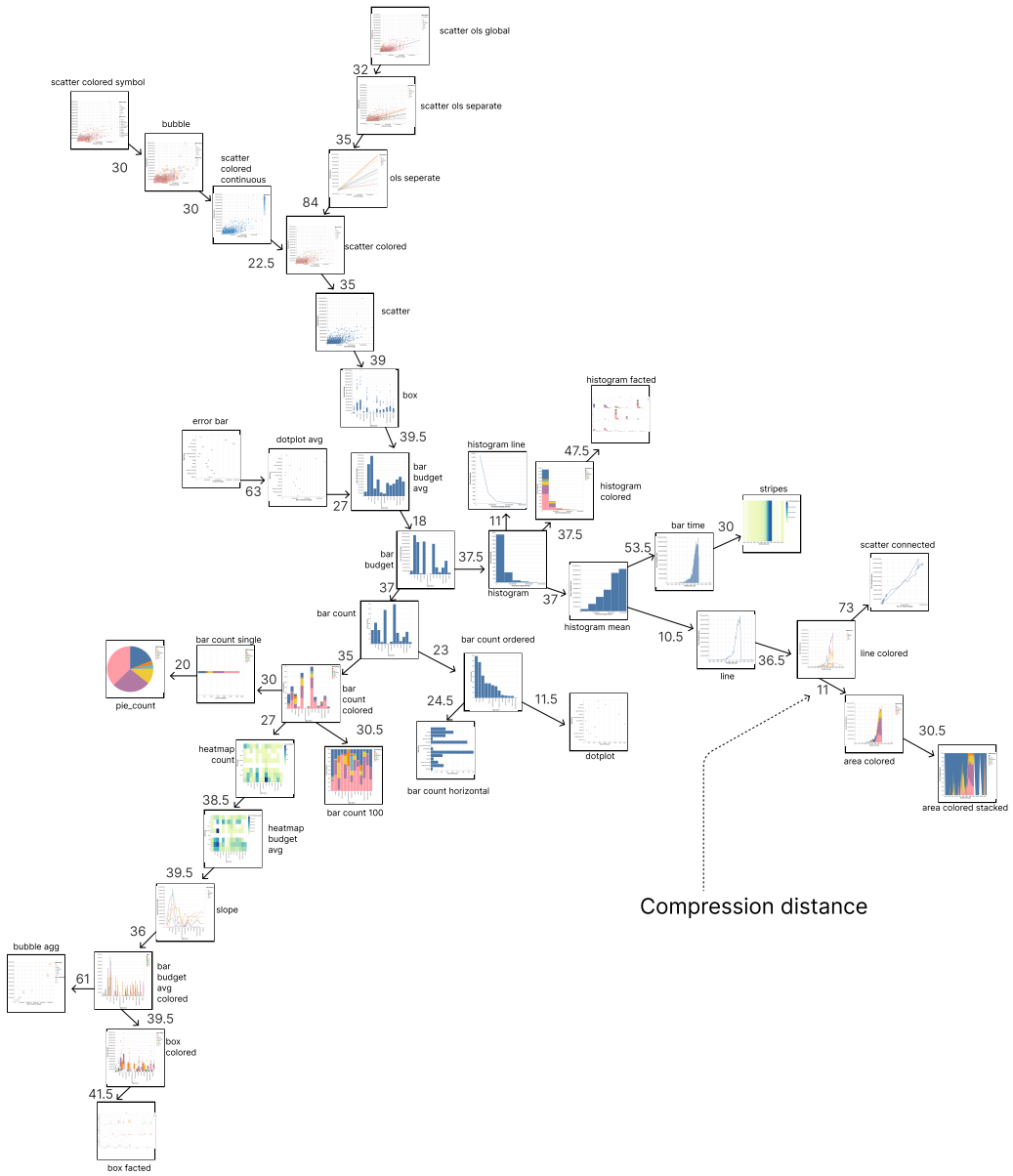

Fig. 11: Notascope is capable of producing a wide range of visualizations, beyond those that we explore in the paper. For instance, here we show a minimum spanning tree for the Vega-Lite specifications from our gallery using our compression distance metric. Such visualizations allow the reader to explore a path of minimum alteration through the space described by the gallery.