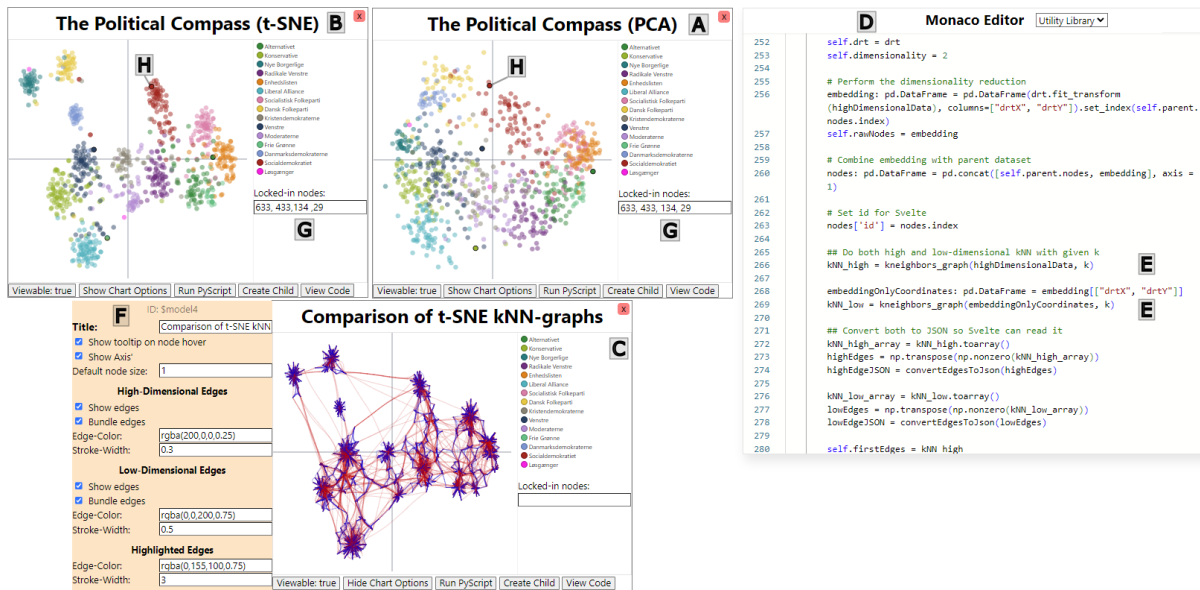# A Web Framework for Explainable and Malleable Visualisation

Simon Malthe Hansen[1] , Ira Assent[1] , Hans-Jörg Schulz[1]

[1]Department of Computer Science, Aarhus University, Denmark

Figure 1: *Create multiple, movable Charts each with their own visualisation (A, B, C). Each Chart has unique, editable and seamlessly executable Python code for pre-processing their own dataset and -model (D) and can use common Python libraries (E). The visual model of each Chart can be changed with a dockable options pane (F), and data points can be highlighted to compare data between Charts (G, H).*

Abstract

We present a novel web-framework which combines malleability in both the visualisation and pre-processing steps of the data visualisation pipeline. The framework lets users create Charts which can be visually modified to the use case, and each have their own fully editable Python code model with access to Python's extensive libraries. This puts the user in control over both pre-processing in Python and the final visualisation, making the effects of each pipeline step explainable and transparent.

CCS Concepts
• Human-centered computing → Visualization toolkits; • Software and its engineering → Integrated and visual development environments;

1. Introduction

In data visualisation, the algorithmic analysis of datasets is often done in a pre-preprocess using analytic tooling from the field of data science, usually in Python. Whereas the mapping and rendering is done "live" using web-frameworks like D3 [BOH11] or Vega Lite [SMWH17] to create interactive charts and diagrams. This divide between the analytic pre-processing of the data and its interac-

tive visualisation hinders visualisation authors to easily understand how a visualisation comes about, let alone to change it in a quick manner that encompasses all stages of the visualisation pipeline.

Recent frameworks have reduced this gap in the visualisation pipeline by letting users edit the visualisation code, but often are limited to JavaScript [MCM*18], focused on literate computing [BMR*19], or introduce additional overhead [CTTAPGL22].

To address this challenge, we introduce a light-weight framework using web technologies, which combines customisable visual layouts with a Python editor for on-the-fly changes to both the pre-processing and mapping steps of the data visualisation pipeline. To illustrate our framework's potential, we use it to compare dimensionality reduction algorithms, each of which brings out different aspects of a dataset. We do so by applying it to the creation of a Political Compass visualisation of the Danish political spectrum.

2. Architecture

Our framework is built on three web-technologies – Svelte [BG23], PyScript [SS23], and Monaco [Mic24] – and it is available at `https://vis-au.github.io/webframework/`. It allows creation of an arbitrary number of visualisations, called *Charts*. Following the Visualisation Reference Model design pattern [HA06], Charts serve the *Visualisation* role. They are defined in Svelte, and each of them manages a clear separation of the data and visual models, holds two controllers and has a single view.

The data model for each Chart is split into two parts: (1) A Python code file holding the user-written pre-processing instructions, which is controlled by and edited through Monaco. (2) The output of said Python code after it has been interpreted by PyScript. This returns a Python *chart*-object, which holds the data in the form of a set of nodes (data points) and up to two optional sets of edges connecting them, in addition to relevant metadata. This chart object is accessible in Svelte, and the Chart processes it slightly before handing it to the visual model, a Svelte sub-component.

The visual model maps nodes and edges from the data model to visual marks (HTML- and SVG-elements), and appends them to the DOM to create the actual graph layout view of the Chart. A dockable options pane serves as a controller, which uses Svelte's reactivity to let the user change the visual model, and thus the view.

Each Chart can have children, which themselves are Charts. Although the Python environment of any Chart can access the Python chart-objects of all other Charts, it is especially easy for a child to access that of their parent. This lets the user split the pre-processing pipeline into multiple steps with a dedicated view for each of them.

3. Malleable and Explainable Visualisations

By splitting the underlying model for each Chart into a data model in Python, and a visual model in Svelte, both with their own controller, we let the user change both steps of the visualisation pipeline. This introduces *malleability* of the framework in both the pre-processing steps for the data model, as well as the visual mapping aspects in the final view. The user can make changes while the framework is running. This makes modifications of the models and view for each Chart a seamless part of interacting with it.

Since the malleability of the framework puts the user in control of the pipeline, they can make small changes and view the results. This makes the effects of the changes transparent to the user, giving a high degree of *explainability* over each step of the pipeline—for example, how the hyperparameters of a pre-processing algorithm change the data and thus the resulting visualisation, effectively being able to explain why the visualisation looks the way it does.

Figure 1 gives an overview of our framework, showing three Charts in (A, B, C). Each Chart can be moved with direct manipulation, adding to the malleability of the entire environment, and lets the user create a multiple view setup suitable for their use case. The user can modify the Python text model for each Chart by opening it in Monaco (D). If the dataset after the pre-processing in Python is 2-dimensional, the visual model can be translated into a Scatterplot (A) or a Graph (C) based on the settings in the options pane (F).

4. Visualising the Political Compass

To visualise the Danish political spectrum, we use a dataset based on 35 questions answered by political candidates ($N = 855$) running in the Danish 2022 general election. The dataset is pre-processed with dimensionality reduction algorithms to create a 2D view. We use our web-framework to compare the embeddings from t-SNE (B) and PCA (A), and how they change global structure.

We use template functionality from the framework's *Utility Library*, a Python code file, to load in the dataset from disk. When the Python code for a Chart is interpreted, this library is prepended. It has out-of-the-box functionality for the user to easily get started, and is fully editable, letting the user add methods, classes and objects shared between all Charts. Since PyScript runs a full Python environment, it grants access to Python's extensive libraries, which we use to run PCA (A) and t-SNE (B). The resulting visualisations shows their vastly different low-dimensional embeddings.

To compare how specific nodes (politicians) are positioned by t-SNE and PCA, we first assign each node in the data model for both Charts an *id*. We then 'lock in' the ids of interest in the visual model (G). This highlights the nodes in the actual embeddings (H), letting us see that PCA has a specific politician positioned between two parties, which t-SNE instead puts squarely inside one of the parties. This shows the stark difference of the position of individual politicians based on the pre-processing algorithm chosen.

To understand the changes in global structure after dimensionality reduction, we create two sets of edges from kNN-graphs (E). One for the high- and low-dimensional embedding respectively. We then in (F) change the visual model to render both edge sets, and bundle them to reduce clutter. The result in (C) shows the change in nearest neighbours of each node after dimensionality reduction, and how low- and high-dimensional structure differ.

5. Conclusion and Future work

We have presented a web-framework, which combines malleability in both the visual mapping and pre-processing steps of the data visualisation pipeline. This gives high transparency and explainability of the resulting visualisation. We seek to improve the framework by adding more Chart visual layouts than graphs, better supporting visualisation of metrics related to pre-processing algorithms, and improve persistence of visualisation setup and model code.

Acknowledgements

References

[BG23] BHARDWAZ S., GODHA R.: Svelte.js: The most loved framework today. In *2023 2nd International Conference for Innovation in Technology (INOCON)* (2023), pp. 1–7. `doi:10.1109/INOCON57975.2023.10101104`. 2

[BMR*19] BADAM S. K., MATHISEN A., RÄDLE R., KLOKMOSE C. N., ELMQVIST N.: Vistrates: A component model for ubiquitous analytics. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 586–596. `doi:10.1109/TVCG.2018.2865144`. 1

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2301–2309. `doi:10.1109/TVCG.2011.185`. 1

[CTTAPGL22] COSMIN-TOADER N., TRINCADO-ALONSO F., PASTOR L., GARCIA-LORENZO M.: VMetaFlow: A meta-framework for integrating visualizations in coordinated view applications. *IEEE Access 10* (2022), 94545–94559. `doi:10.1109/ACCESS.2022.3202543`. 1

[HA06] HEER J., AGRAWALA M.: Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 853–860. `doi:10.1109/TVCG.2006.178`. 2

[MCM*18] MEI H., CHEN W., MA Y., GUAN H., HU W.: VisComposer: A visual programmable composition environment for information visualization. *Visual Informatics 2*, 1 (2018), 71–81. `doi:10.1016/j.visinf.2018.04.008`. 1

[Mic24] MICROSOFT: Monaco editor, 2024. URL: `https://microsoft.github.io/monaco-editor/`. 2

[SMWH17] SATYANARAYAN A., MORITZ D., WONGSUPHASAWAT K., HEER J.: Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 341–350. `doi:10.1109/TVCG.2016.2599030`. 1

[SS23] SONG Q., SANJARI S.: Pyscript for scientific projects: an introduction, May 2023. `doi:10.5281/zenodo.7907144`. 2