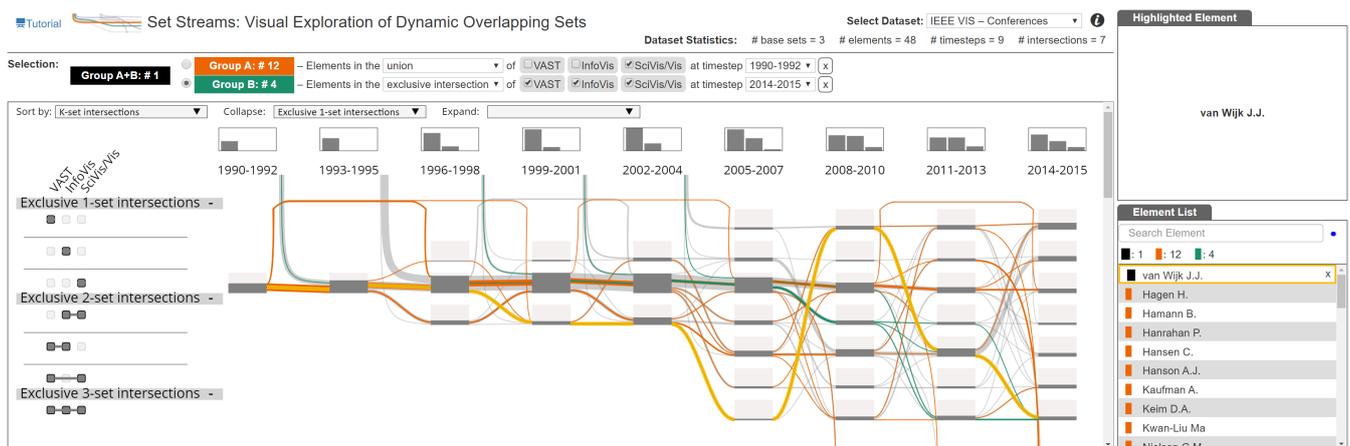


# Set Streams: Visual Exploration of Dynamic Overlapping Sets

Shivam Agarwal<sup>†</sup> and Fabian Beck<sup>‡</sup>

paluno, University of Duisburg-Essen, Germany



**Figure 1:** Set Streams interface: In the main visualization, each row is a specific set intersection, and each column represents a timestep; changing set membership is visualized by streams. The example shows as elements the most frequent authors of the IEEE VIS conference series and their contributions to the conference tracks as set memberships changing across time. Marked in orange are the contributors of SciVis/Vis in 1990–1992, marked in green the authors who have contributed to all three tracks in 2014–2015. Marked in black are the common elements of these two sets of authors; author van Wijk is additionally highlighted in yellow on user selection.

## Abstract

In many applications, membership of set elements changes over time. Since each element can be present in multiple sets, the sets also overlap. As a result, it becomes challenging to visualize the temporal change in set membership of elements across several timesteps while showing individual set intersections. We propose Set Streams, a visualization technique that represents changing set structures on a timeline as branching and merging streams. The streams encode the changing membership of elements with set intersections. A query-based selection mechanism supports a flexible comparison of selected groups of elements across the temporal evolution. The main timeline view is complemented with additional panels to provide details about the elements. While the proposed visualization is an application-independent visualization technique for dynamic sets, we demonstrate its effectiveness and applicability through three diverse application examples and expert feedback.

## 1. Introduction

Classification of data items into categories guides many data analysis tasks. If not measured or determined from an external source, analysts formulate criteria for automatic classification or work

through the data items in a manual or semi-automatic classification process. Often, these categories reflect the central high-level abstractions that provide the main structure for the data, like the professional role or main expertise of a person, the market position of a company, the dominating types of crimes in an area, etc. Since each category is well-defined and contains several data items, it can be modeled as a set. Whereas this data structure is simple, the analysis becomes complex when data items belong to more than one set, and as a result, sets overlap. Additionally, in many practically

<sup>†</sup> e-mail: shivam.agarwal@paluno.uni-due.de

<sup>‡</sup> e-mail: fabian.beck@paluno.uni-due.de

relevant scenarios, the data changes over time and makes the analysis even more challenging: roles of persons are adapting, market positions are changing, and crime hotspots are shifting. Whereas there exist techniques for visualizing static sets or dynamic partitions, hierarchies, or graphs, only surprisingly few techniques have been suggested for visualizing dynamic sets. In particular, we lack techniques that show the *flow* of elements between the set intersections over time.

We hence propose a novel technique to support the visual exploration of dynamic sets highlighting changing element membership as *streams* on a timeline. As illustrated in Figure 1, the main visualization shows time from left to right as part of a grid where rows represent individual set intersections. Each cell of the grid shows a particular set intersection at a specific timestep, with the fill level of the rectangle quantifying the number of associated elements. The glyphs in the first column of the grid identify the set intersections. For understanding the visualization, it is crucial to note that elements are only represented once across all set intersections per timestep: if they belong to multiple sets, they are only visualized as part of the most specialized set intersection (i.e., the intersection of all sets the element is part of; we call this *exclusive intersection*). Only this allows us to connect nodes of adjacent timesteps with *streams* to form a linearly arranged, timeline-based *Sankey diagram*; *streams* are sometimes metaphorically also called *flows*, *rivers*, or *ribbons*. These streams represent and quantify in their width groups of elements that perform equivalent set membership changes between two timesteps, or unchanged elements if the stream flows horizontally. To support an interactive exploration, the visualization approach features a query-based selection mechanism that allows a coloring-based comparison of different groups of elements. Further views provide details on the set elements.

Our approach can be considered as a new information visualization technique, with additional options to support data exploration. Since the approach is intended to be general and application-independent, we have not tailored it for a specific use case, but instead show its versatile applicability in a diverse set of application examples. In summary, our contributions include

- an application-independent, novel technique for visualization of dynamic overlapping sets (Section 4.1),
- a versatile querying technique for visual selection of elements across the dynamic sets (Section 4.2),
- three application examples complemented with feedback from domain experts to demonstrate the effectiveness and generality of proposed techniques (Section 5), and
- an interactive web-based tool that implements the proposed techniques [AB20].<sup>†</sup>

## 2. Related Work

The visualization of set membership is an established area and several approaches have been developed to visualize static sets, as surveyed by Alsallakh et al. [AMA\*16]. We take inspiration from *Up-Set* [LGS\*14] and adopt a grid-based layout where individual set

and set intersections are shown in separate rows. However, *Up-Set*, as well as most other set visualization techniques, is limited to static set structures—developing visualization techniques for dynamic sets is listed as open research challenge in the literature survey [AMA\*16, Section 6.1].

**Dynamic Set Visualization.** Some approaches have already started to address this challenge. *TimeSets* [NXWW16] shows the temporal changes in sets by extending KelpFusion [MRS\*13]. However, it models events as elements that occur only at a single timestep. Hence, set memberships of elements in *TimeSets* do not change over time, unlike in our model. Valdivia et al. [VBP\*20] visualize dynamic hypergraphs, where each hyperedge between nodes can be considered as a set. However, inferring temporal changes is difficult because tracing the same or similar hyperedges across time is not directly supported. *BubbleSets* [CPC09] places elements vertically where the horizontal axis represents time. It shows set membership and evolution of elements by computing overlays while using colors for individual sets. Animation-based approaches visualize individual changes in sets with time; to reduce the user's gaze shift while watching the animation, grouping and sequence of the temporal changes is optimized [MWT19]. Individual changes are easy to follow in an animation. However, if there are many changes across time, it becomes difficult to track and recall them. Our approach focuses on visualizing temporal changes through static visual elements on a timeline to provide an overview of all changes in sets across multiple timesteps.

**Visualizing Dynamic Partitions.** Visualizing dynamic changes in partitions (i.e., sets without overlap) is a related area. Rosvall and Bergstrom [RB10] present the base technique of visualizing dynamically changing non-overlapping communities as branching and merging flows. Other approaches extend or modify this approach by linking it to geographic visualizations [vLBA\*12] or by adding information about underlying graph structures [VBAW15]. Our approach has been inspired by these approaches and we borrow their base layout (which makes them visually similar to ours), but in contrast to these approaches, we focus on explicitly visualizing overlaps of dynamic set structures instead of non-overlapping partitions interpreted as dynamic communities. Additionally, the proposed visualization addresses several challenges: ensuring scalability, enabling a flexible query-based selection, and finally visualizing results of query on existing visualization.

**Visualizing Dynamic Hierarchies and Graphs.** Neighboring areas are visualizing dynamic changes in structures such as hierarchies (i.e., sets with only subset relationships) and graphs (i.e., a generalization of relational data). If only allowing inclusion relations between sets, the sets form a hierarchy. Visualizing the evolution of such hierarchical sets becomes a problem of hierarchy comparison and evolution [GK10]. While most approaches in this area focus on the comparison of two hierarchies, few approaches also cover the representation of sequences of hierarchies, for instance, to show the evolution of hierarchically structured code [TA08] or clusters in a dynamic graph structure [VBW16]. Dynamic graph visualization [BBDW17] is related as overlapping set structures can also be transformed into graphs, for instance, a bipartite graph or a hypergraph. However, most dynamic graph visualization tech-

<sup>†</sup> Hosted at: <https://vis-tools.paluno.uni-due.de/setstreams/>

niques, at least timeline-based ones, are not prepared to handle such special types of graphs.

### 3. Design Considerations

We started this project with some fundamental considerations, which guided the overall design of the visualization. They are based on insights and related approaches from literature as well as long-term experience developing visualization approaches for discrete dynamic data. We introduce icons for identifying the three resulting design decisions and use these icons for referencing the decisions throughout the paper.



**Get a temporal overview.** For showing temporal changes in a visualization, temporal encoding as an animation and spatial encoding as timelines are the two most common forms of encoding. Whereas animation often provides an easy-to-understand representation, it is also known to have limitations regarding a good readability of non-trivial data [TMB02]: “Animations are often too complex or too fast to be accurately perceived. Moreover, many continuous events are conceived of as sequences of discrete steps.” For instance, in context of dynamic graph visualization, first approaches were animation-based, but the research focus is shifting more and more to timeline-based approaches [BDDW17]. For our approach, we also favor a spatial encoding on a timeline because a primary goal is providing an overview of time. A resulting challenge is though that we have to find a condensed representation of the set structures. Since we want to show several timesteps juxtaposed from left to right on one screen, ideally, these should have the form of a column or vertical stripe. We ruled out alternative, less common encodings of time, for instance, in color or in other visual variables such as shape, size, etc. as they make comparing the state of the data at a certain timestep difficult.



**Follow elements across time.** Our focus for analyzing temporal changes is on the elements that change their set membership over time. For representing this, we consider the metaphor of a flow of elements through a temporal, but static structure of set intersections. If an element changes its attributes and moves from one set to another, we want to draw a connecting line. This is a popular representation widely used and often called *Sankey diagram* (which dates back to works from the 19<sup>th</sup> century by Minard, 1869 and Sankey, 1898). A Sankey diagram can be linearized to be used on a timeline while highlighting the temporal changes in groups of items through bands or ribbons, which is sometimes called *alluvial diagram* [RB10]. However, such flow-based diagrams are not directly applicable to overlapping sets. To work around the overlap problem, elements can get represented multiple times if they contribute to several sets [BMBW15]. But this takes away the focus from the elements changing their membership and the overlap of sets, and shifts it to a simpler aggregation of the data. Since this shift of focus would be against our goals, we explore an alternative approach: we artificially *flatten* the overlapping sets by considering each non-empty exclusive intersection of sets as an independent node. Through that, we can get a non-overlapping categorization of elements on which we can apply a flow-based encoding. Finally, branching and merging streams represent set membership changes in the visualization.



**Compare groups of elements.** One issue in streams, however, is that individual elements or groups of elements get absorbed by the stream. An element that flows into a stream, in the next timestep, cannot be discerned from the other elements of the stream anymore. We hence need a mechanism to highlight elements of interest across time. While, during an exploration, the elements that are of interest might change frequently, this mechanism requires an interactive selection of individual elements and groups of elements. Such selections are commonly highlighted in a visualization using color. Since we have not used color for encoding another variable already, we can follow this approach. However, we want to go a step ahead of regular selections and also support the comparison of different groups. At least two different groups of elements should be comparable. For a color-based encoding of this, we need four different colors: one for non-selected elements, one for elements of the first group, one for elements of the second group, and one for elements that are in both selected groups. We further require a flexible querying approach to select such groups with respect to different criteria, for instance, based on their set membership at a point in time.

Beyond these specific considerations, when designing the approach, we followed general best practices and accepted guidelines for information visualizations, such as consistently using colors, reducing visual complexity and clutter where possible, and introducing expressive labels.

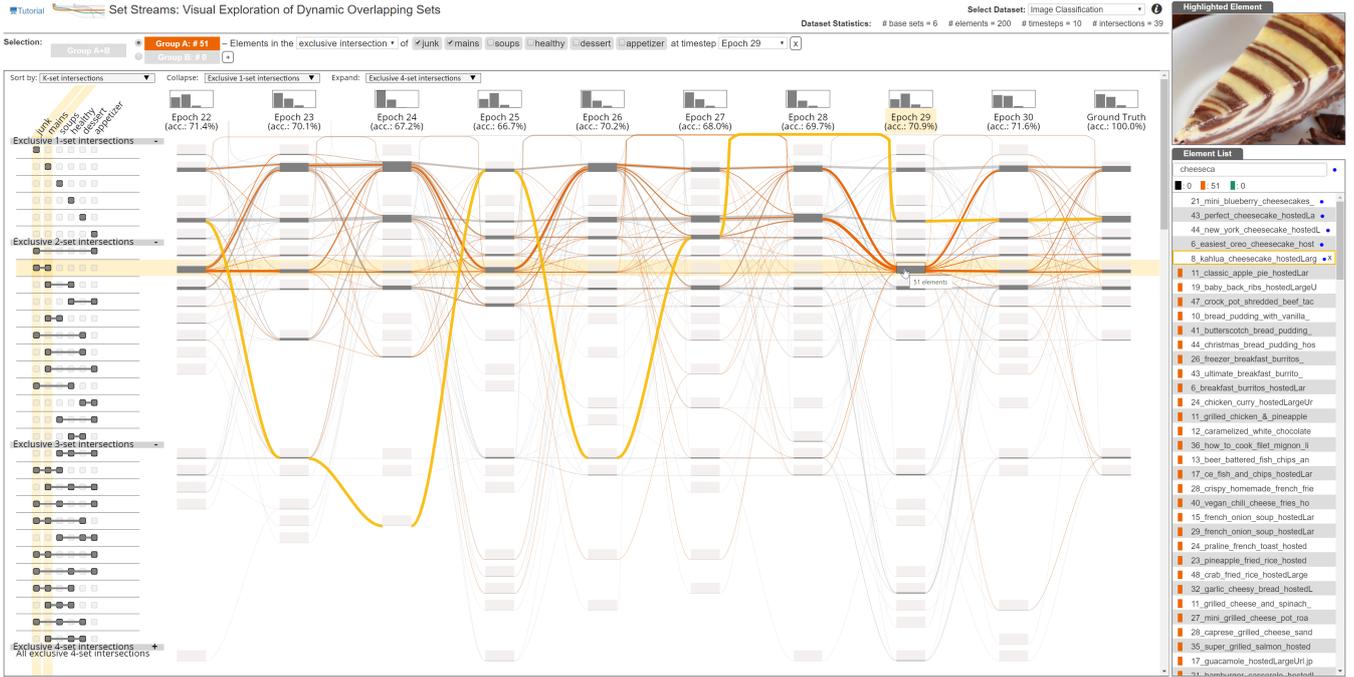
### 4. Set Streams Visualization Approach

The above design considerations already outline a timeline-based visualization approach that builds on Sankey diagrams and color-coded selections of one or two groups of elements. However, this only provides a sketch still leaving out many details of the visualization design. Also, additional features, encodings, and views are necessary to complete the sketched approach to a full-fledged interactive visualization technique. For describing the encodings and interactions in an exact and reproducible way, we build on a formal data model introduced in the following before discussing details of the visualization approach. The full interface is shown in Figure 1 for a small dataset and in Figure 2 for a larger one.

Let  $F = \{S_1, S_2, \dots, S_n\}$  be a family of  $n$  **base sets**, where each set  $S_i \subset V$  contains elements from a **universe**  $V$ . The power set of this **family of base sets**  $\mathcal{P}(F)$  describes every possible combination of these sets with  $|\mathcal{P}(F)| = 2^n$ . For each family of sets  $F_j \in \mathcal{P}(F)$ , we can compute a **set intersection** (or overlap) of the contained sets  $I(F_j) = \bigcap_{S \in F_j} S$ . An element  $v \in V$  might belong in multiple intersections, for instance,  $v \in S_1, S_2, S_3 \Rightarrow v \in I(\{S_1, S_2\}), I(\{S_2, S_3\})$ . As we want to avoid repeated encoding in different intersections for showing the flow of elements , we further define an **exclusive set intersection**

$$\bar{I}_F(F_j) = \{v \in V \mid (\forall S \in F_j : v \in S) \wedge (\forall S' \in F \setminus F_j : v \notin S')\}$$

(i.e., each element included is not included in another base sets that is not considered in the intersection). In the above example ( $v \in S_1, S_2, S_3$ ),  $v \notin \bar{I}_F(\{S_1, S_2\}), \bar{I}_F(\{S_2, S_3\})$ , but  $v \in \bar{I}_F(\{S_1, S_2, S_3\})$  if also  $v \notin S_4, S_5, \dots, S_n$ . Note that, if an element  $v \in V$  is not contained in any  $S_i \in F$ , then  $v \in \bar{I}_F(\emptyset)$ . Hence, the set of all exclusive intersections over  $F$  forms a partition of  $V$ . To model temporal



**Figure 2:** Training of a multi-label classifier for image classification: The timeline shows the predicted labels for each image across various epochs (training stages) of the classifier. The last column represents the ground-truth assignment of labels. The exclusive intersection of labels ‘junk’ and ‘mains’ is selected in Epoch 29 (orange) and highlighted on hovering.

changes with  $m$  timesteps, we introduce a **sequence of families of base sets**  $\mathcal{F} = (F^1, F^2, \dots, F^m)$  over the same universe  $V$ , where each  $F^k = \{S_1^k, S_2^k, \dots, S_n^k\}$  consists of the same number of  $n$  base sets. Through this, we can follow a set, which represents a categorical attribute of the elements, across time.

#### 4.1. Timeline Visualization

The timeline-based visualization of set evolution forms the main view of the visual interface. It has a grid-like structure. While timesteps are organized from left to right, the rows represent the different exclusive intersections.

The icons at the left of each row identify the respective exclusive intersection  $\bar{I}_F(F_j)$ : Each base set  $S_i \in F$  is represented as a small rectangle, while those that are involved in the exclusive intersection ( $S_i \in F_j$ ) are marked in darker gray and connected by a line. Empty exclusive intersections (i.e., if  $\bar{I}_{F^k}(F_j) = \emptyset$  in all timesteps  $k = 1, 2, \dots, m$ ) are not shown as rows to save space. Only indirectly represented is the set of elements not contained in any of the base sets  $\bar{I}_{F^k}(\emptyset)$  as discussed below. The rows are ordered by default based on the cardinality of the respective family of sets  $|F_j|$  (i.e., how many base sets are contained therein). This allows adding headlines to each group of rows that have the same cardinality, starting with *exclusive 1-set intersections*, followed by *exclusive 2-set intersections*, etc.

A cell of the grid represents a particular exclusive set intersection at a specific timestep. To display the state of each of these

intersections, we draw a rectangular box in each grid cell as a node of the Sankey diagram. The number of elements contained in the respective exclusive intersection  $|\bar{I}_{F^k}(F_j)|$  is encoded by the height of the gray bar inside the box of the cell. If the set or the set intersection is empty at a particular timestep (i.e., does not contain any exclusive element), we do not draw the box at the corresponding column. Hovering over a cell highlights the timestep label, the row, and participating sets in the respective intersection.

Curved lines connect the nodes from left to right through the grid and form the branching and merging streams. Each element contributes to exactly one stream per transition from one timestep to the next according to which exclusive intersection it belongs at the earlier and the later timestep. Formally, two nodes representing the exclusive intersections  $\bar{I}_{F^k}(F_j)$  and  $\bar{I}_{F^{k+1}}(F_{j'})$  are connected if  $w := |\bar{I}_{F^k}(F_j) \cap \bar{I}_{F^{k+1}}(F_{j'})| > 0$ . Additionally, the width of the stream encodes the number of elements  $w$  undergoing the same transition between the two timesteps. The elements added in a timestep (i.e., that are not present in the previous timestep) are shown by streams originating from above placed at the left of the corresponding timestep. Similarly, elements that do not belong to any set in any of the next timesteps are shown going down at the right side of current timestep. Elements that belong to some sets not in the next but in later timesteps are shown by streams which go upwards above the first row of the grid; they rejoin in the respective later timestep similar to incoming streams. To avoid unnecessary clutter, we sort the edges based on the vertical position of their destination. Hence, we first draw streams that are going upwards,

followed by the one that connects the same exclusive intersection in the next timestep, and finally those that are going downwards.

A cardinality distribution at each timestep is shown by a histogram placed above the corresponding column of a timestep. Each bar represents a cardinality  $c$  of the family of sets involved in respective exclusive intersections. The height of the bar is determined by how many elements are in the respective exclusive intersections at timestep  $k$  that have cardinality  $c$ :

$$\left| \bigcup_{F_j \in \mathcal{P}(F^k); |F_j|=c} \bar{I}_{F^k}(F_j) \right|$$

The rows of the grid can be interactively aggregated and sorted, deviating from the default arrangement as described above. Rows can be aggregated based on their cardinality of participating sets, treating the union of all exclusive intersections with this cardinality as the aggregated representation in a single row (similar to what a bar in the histogram represents). We also include other options of sorting the rows (exclusive set intersections): by decreasing order of the number of contained elements in a selected timestep  $k$  ( $|\bar{I}_{F^k}(F_j)|$ ) or summed across all timesteps ( $\sum_{k=1}^m |\bar{I}_{F^k}(F_j)|$ ). Additionally, as shown in Figure 3, rows can be sorted by the decreasing order of their stability, which is computed as the ratio of stable elements to all elements contained in the intersection at a timestep, summed across all timesteps. We also include an option to sort rows based on their similarity, which is computed as number of elements switching their membership between two exclusive intersections across all timesteps; we use a greedy approach: place the exclusive intersection with the highest number of incoming elements first and then always place the most similar one next. Finally, the rows can also be sorted by assigning priority to a set, which first lists all exclusive intersections in which the set is involved and sorts them with increasing cardinality, followed by the remaining set intersections in default order. Figure 3 demonstrates the aggregation and sorting features where all rows with cardinality three are aggregated and then sorted based on their stability. Collapsing and expanding as well as sorting rows can be triggered by respective drop-down selections at the top of the timeline visualization.

#### 4.2. Query-based Selection

To support the comparison of different groups of elements across time , we propose a visual query technique. We face two challenges: (i) the specification of queries should be simple and intuitive, and (ii) various different groups of elements should be selectable. Finally, the resulting selection needs to be overlaid on the timeline-based visualization.

After experimenting with different solutions and discarding more complex ones, we decided to embed the query into a short sentence where different parameters are selectable. This makes the query readable and self-explanatory, while still providing sufficient flexibility to specify various queries. The query includes the following parameters:

- **Set Operation:** In a drop-down selection field, user can select both the non-exclusive intersection  $I(F_j)$  and the exclusive intersection  $\bar{I}_F(F_j)$ . In addition, a set union operation is available, which is defined as  $U(F_j) = \bigcup_{S \in F_j} S$ .

- **Base Sets:** Marking checkboxes in a list, users can activate arbitrary combinations of base sets to specify  $F_j$ .
- **Timestep:** Finally, users choose the timestep  $k$  from a drop-down selection field.

We allow the specification of two of such queries, which can be added and cleared with respective buttons. Both resulting groups of elements—*Group A* and *Group B*—get assigned a distinct color (A: orange; B: green). These colors are used to highlight the respective streams in the visualization. Since elements can be shared between the two groups, a third color (black) is necessary to visually discern the shared elements if both groups are activated.

Besides specifying the query through the above parameterized sentence, groups of elements can also be selected by interacting with the visualization. Clicking on a node in the stream visualization, which represents an exclusive intersection  $\bar{I}_{F^k}(F_j^k)$ , the equivalent query gets activated, automatically selecting the parameters *exclusive intersection*, checking the respective sets of  $F_j^k$ , and selecting timestep  $k$ . As a result, all elements and streams that relate to the node get highlighted as Group A by default, or Group B if toggling the respective radio button on the left side of the query. Similarly, clicking on an edge selects the respective elements of this transition. As an edge selection goes beyond what can be represented in the query form described above, we switch to an alternative sentence describing the respective selection.

#### 4.3. Linked Views

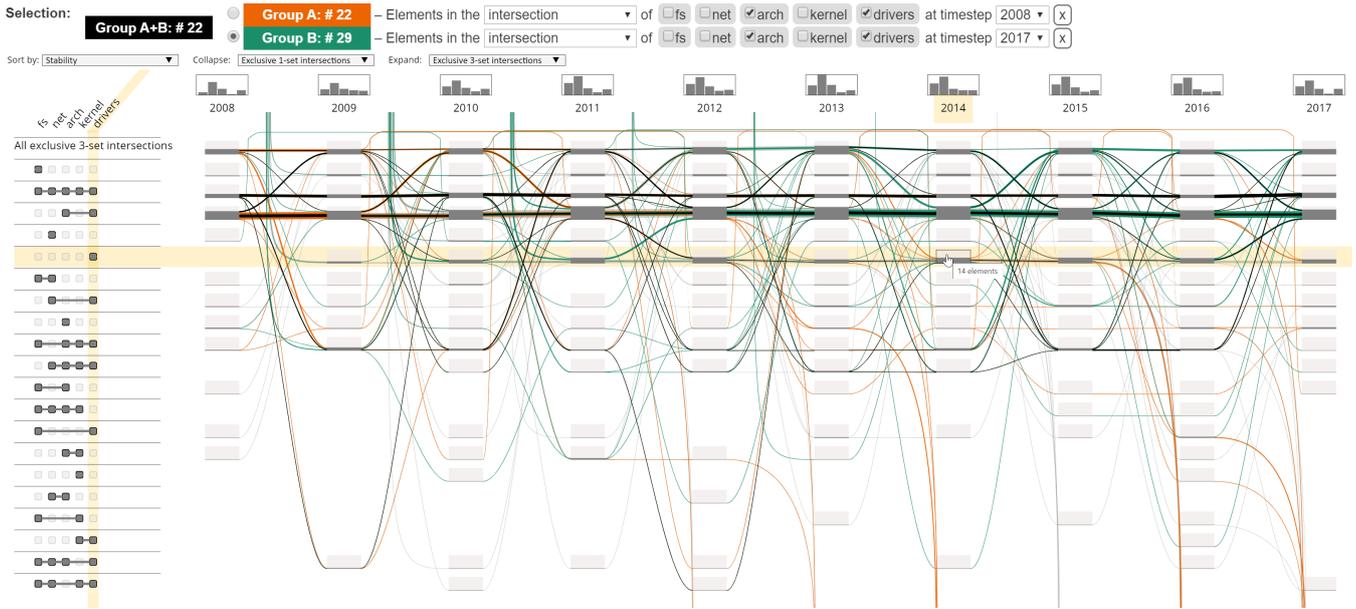
We further enrich the user interface with additional, linked views to provide details on demand. A list at the right side of the visualization shows all elements in individual rows. A search bar enables finding specific element by their name, which are then marked with a blue dot in the list (Figure 2). The colored bars at the beginning of a row indicate that the element is part of the currently selected groups . Single elements can be highlighted from the list by click. A panel above the element list contains additional details of the dataset, for instance, the name of the highlighted element or, if available, an image. The selected single element is highlighted in yellow as another overlay on the stream visualization. The element list is ordered and shows first the search results, then the selected elements of Group A+B, Group A, Group B, and last all other elements. Alphabetic order is the secondary sorting criterion (if nothing is selected/prioritized and for sorting within the groups).

### 5. Application Examples

We discuss the usefulness and generality of the proposed system through three application examples covering areas such as bibliometrics, software engineering, and machine learning. We complement this with feedback provided by experts users from the three fields, who evaluated and extended the discussed examples.

#### 5.1. Expertise of Researchers

Expertise and background of researchers can be estimated by the publication channels or keywords of their publications. The researchers form the elements while the fields, as identified by the channels, represents the sets. Visualizing this information across



**Figure 3:** Software evolution: Contributors of the Linux project form the elements, assigned to different parts of the system ('fs', 'net', 'arch', 'kernel', and 'drivers') according to their code contributions within a year (2008–2017). Exclusive 3-set intersections are aggregated. Rows are sorted based on stability. Intersections of 'arch' and 'drivers' are selected for the year 2008 (orange) and 2017 (green).

time, we can observe, for instance, new topics or venues trending. The evolution of set memberships shows what researchers worked on before moving to the new topic and if they continued working on the previous topics. Other insights relate to the connectivity of sub-communities and their historic development. We demonstrate this application with a dataset covering the most frequent authors of the *IEEE VIS* conference series and its tracks. Please also note the two other examples available in the interactive tool, covering keyword-based classifications of researchers in the visualization and the broader computer science community.

**Dataset.** The *Visualization Publication Data Collection* [IHK\*17] with publication data of *IEEE VIS* serves as a sample (1990–2015, 2752 publications). The different tracks of the conference series represent the sets: *SciVis/Vis* (the original *Vis* conference is considered here as a predecessor of *SciVis*), *InfoVis*, and *VAST*. Authors of publications form the elements. We filter the authors with a minimum of 15 publications, which results in 48 authors. We aggregate publications over periods of three years each to one timestep.

**Findings.** Figure 1 shows the resulting visualization for this dataset. First, investigating the general structure, we clearly observe the origin of the conference series in the *SciVis/Vis* track, from which first the *InfoVis* track is branching (1996–1998) and later the *VAST* track (2005–2007). The introduction of the *VAST* track was successful in the sense that the established researchers represented in this dataset also started publishing in this track from the beginning. However, nobody immediately switched to only publishing at the *VAST* track only. From 2005, in general, the many diagonal connections between the nodes show good mobility between the tracks.

Also, mainly in the last two timesteps, all combinations of tracks are non-empty and populated by multiple authors; this shows that the tracks do not separate the community. In the period of 2005–2010, the situation was clearly not as balanced between the tracks yet, but still more focused on *SciVis/Vis*. Second, we contrast particular groups of authors. For instance, as marked in Figure 1, we can compare the group of early contributors (marked in orange) and the group of recent generalists (marked in green). The groups only share one researcher (van Wijk, in black, but also highlighted in yellow). Many of the early contributors are still active in the community (only few orange edges leading out at the bottom) and have spread across all combinations of tracks. Further, it is interesting to note that the recent generalists all started their contributions to the conference with contributions to *SciVis/Vis* (green incoming edges all lead to the exclusive *SciVis/Vis* node).

### 5.2. Software Evolution

Studying the evolution of a software project provides insights on the team structure, the status of the development, and expertise areas. Both software developers as well as team leaders or managers rely on such information to find the right person to ask about a certain part of the system, to keep up-to-date with the development, or to understand the history of the existing projects when boarding a new team. Similar data has been studied by others with stream-based visualizations, for instance, focusing on code structure [TA08] or also discussing developer contributions [BMBW15].

**Dataset.** As an example of a large software system, we investigate the evolution of *Linux* from 2008 to 2017. We split the project into five main development parts that we call *modules*. Contributors

are assigned to a module if their commits to the project repository changed some files in the respective module. Filtering the *Linux* developers to those that made at least 100 commits, the resulting dataset contains 111 contributors (elements). To focus our analysis, as shown in Figure 3, we aggregate all exclusive 3-set intersections and sort the rows based on stability of the exclusive intersections.

**Findings.** Despite investigating a range of ten years, the overall structure of the contributors stays quite stable. For instance, there exists a stable set of generalists across time (the third row) who contributed to all modules throughout the whole period (Jiri Kosina, Greg Kroah-Hartman, David S. Miller, Linus Torvalds, Al Viro). A remarkable intersection that leads to another consistent pattern is between the modules *arch* and *drivers*. These two modules seem to be closely interlinked. In Figure 3, the (non-exclusive) intersection of both modules is marked in orange for 2008 and green for 2017. We find that 22 committers were common in the two selections. Most of these common committers (black) are either generalists or contributed only in *drivers* and *arch* (thick black lines in respective rows). Also, inflowing green edges and outflowing orange edges indicate the entering and leaving of respective contributors. The histograms reveal a consistent spike in the 2-set intersections, which relates to this as well. Comparing the different modules with each other, we observe that developers who only contribute to one module can be found for *drivers* and *fs* mostly.

### 5.3. Multi-Label Classification

Labeling—assigning labels to elements such as images—can be characterized as *supervised classification* and is a standard machine learning problem. When the labeling is not restricted to assigning exactly one label per element, but multiple labels can be assigned, showing the results of the labeling process becomes a problem of visualizing overlapping sets. And a temporal component needs to be considered when data analysts who set up the learning approach want to study the training process. Visualizing the evolution of the multi-label classification can help to understand the progress of the learning process, can hint at elements that are harder to label correctly, or can provide indicators for improved learning strategies.

**Dataset.** As an example for multi-labeling, we consider an image classification scenario where images of dishes are classified according to the type of dish.<sup>‡</sup> Since one dish can be, for instance, both junk food as well as a main dish, the labels overlap. In our set visualization, the images form the elements while the labels are the sets. The specific dataset contains 200 images and 6 labels. We visualize the results of training a convolutional neural network. Since there are too many training epochs, we focus on the last epochs of training. To also include the ground truth (i.e., the correct labeling that was manually created for the training and test data), we add it as the last timestep. As important contextual information, we further list the accuracy for each timestep.

<sup>‡</sup> <https://nanonets.com/blog/multi-label-classification-using-deep%2Dlearning/>

**Findings.** A general observation from Figure 2, both from the accuracy and the set evolution, is that the training is toggling: between higher ( $> 70\%$ ) and lower ( $\leq 68\%$ ) accuracy as well as between classifying most elements as *mains* only and as *junk* and *mains*. Interestingly, the changes in accuracy and sets do not align (i.e., the changes appear in different epochs)—this shows that, even when accuracy indicates a quite stable transition, bigger changes might actually happen *behind the scenes*. Marking the exclusive intersection of *junk* and *mains* in Epoch 29 (orange group in Figure 2) as an example for this, we observe: (i) the state with respect to this intersection was similar in Epoch 22 and 25 (though having different accuracy rates), (ii) only about half of the marked elements are correctly classified while the others spread with respect to the ground Truth, (iii) before, in Epoch 28, many of the marked elements were (mostly wrongly) classified as *dessert*. Exploring different examples with on-demand individual highlights, we can find stable examples that are in most epochs classified with the correct label(s), but also unstable outliers of particular relevance for further improving the machine learning approach. For instance, the highlighted picture of a cheesecake in Figure 2 (yellow) should be classified as *dessert*, but jumped around different exclusive intersections until, in Epoch 29 and 30, finally being classified correctly.

### 5.4. Expert Feedback

To confirm the validity and usefulness of the findings described above as well as to receive general feedback, we invited different expert users to test the approach as part of an online study. We used our professional network to recruit at least one expert for each application example. In total, 5 expert users (E1–E5) participated, E1 and E2 having significant experience in bibliographic analysis, E2 and E3 in software evolution research, and E5 in training classifiers. We provided them the tool (including a tutorial) and a preliminary version of the paper (including everything except this section on expert feedback). As part of an online questionnaire, we initially asked them to go through the tutorial and explore the tool before starting the questionnaire, which all participants confirmed. The first task was to reproduce the observations described in the result section of the respective application example (each participant was assigned the application example fitting his/her expertise). Second, the experts were asked to extend the analysis of the application example and report the insights found. Then, the experts were invited to comment on (a) the sorting and aggregation and (b) the query-based selection capabilities of Set Streams. The study concluded with options to provide overall feedback on most and least useful features, missing information or features in the tool, additional analysis tasks that could be performed, and additional remarks. The study was designed to take about 60 minutes. The questionnaire and all responses are available in the supplementary material [AB20].

**Reproduced and Extended Findings.** All experts commented that they were able to reproduce most of our findings, while some of them had problems due to clutter (E1) or relating the findings with the figure (E4). The experts were able to extend the analysis and discovered: most common exclusive set intersections (*InfoVis* and *VAST* – E1; *arch*, *kernel*, and *module* – E3), stability of element memberships in at-least one set across all timesteps (E2), uncommon exclusive set intersections (*kernel* and *drivers* – E4; *net* and

arch – E4), and unusual behavior of some elements not belonging to any set in a few timesteps (E5). Although the experts found additional insights, one expert (E3) commented that the analysis becomes difficult without being closely associated with the dataset and without having a specific question in mind.

**Functionalities.** All experts liked the functionality of sorting rows: based on a timestep (E4), based on the ground truth to get overview of dataset (E5), and based on the priority of specific sets (E1). The experts also liked the aggregation functionality and commented that it helped to reduce the clutter (E1). However, E1 and E4 also mentioned that using dropdown lists for these features is not intuitive and E3 suggested that it could be improved by providing more information on the sorting criteria via tooltip. All five expert liked the query-based selection feature: it is self-explanatory and flexible (E1), formulating queries by selection is easy (E5), and it was useful to compare two groups of elements (E1 and E4). However, the experts also commented that differentiating gray and colored edges becomes difficult (E3 and E5) and too many choices could confuse the users (E1 and E2).

**Overall Feedback.** Three experts (E1, E4, and E5) mentioned that they found the query-based selection to be most useful. In contrast, the answers are too brief and diverse to identify less useful features. The experts suggested using natural language to describe the selection (E4), tooltips to convey extra information (E1 and E3), and integrating domain-specific information such as, classification accuracy of an element at a specific epoch (E5). Expert E2 commented that Set Streams already supports too many tasks and could be limited to reduce visual complexity. Experts E3 and E4 also mentioned that it is difficult to perform free-exploration tasks in Set Streams. Expert E5 suggested to stabilize the ordering of names in the element list and provide a search feature. We have already incorporated the latter two suggestions for the final version.

## 6. Discussion and Future Work

Testing the visualization in different application examples has helped us identify important characteristics of the approach. We discuss advantages and limitations with respect to central dimensions. Our approach can be considered a base technique for dynamic set visualization and is extensible in various directions.

**Data Ordering.** Ordering and grouping the rows of the grid by degree of the intersections provides a clear structure . Elements move down in this structure if they generalize (i.e., they take membership in additional sets), but stay within the same group of rows if they only switch a set membership. Unchanged set membership leads to a stable horizontal line. Some visual clutter is produced by crossing streams. Proposed sorting criteria were found useful during the analysis of different datasets. Sorting based on the similarity of intersection reduces the number of crossings. However, other reordering methods—like suggested for other alluvial diagrams [VBAW15] and similar to edge crossing reduction for layered graph layouts such as in the Sugiyama algorithm [STT81]—could also be implemented. The lines between the timesteps can be bundled to reduce the number of overlaps with individual thin lines.

**Data Aggregation.** Along similar lines, more versatile data aggregation could help to focus on specific aspects. Currently, we aggregate the elements within the same exclusive intersection  $\bar{I}_F(F_j)$  and make subgroups and individual elements selectable on demand. We also enable aggregation of exclusive intersections based on their cardinality, which simplifies the representation. Since the set intersections represented by the rows are mutually exclusive , in general, aggregation of arbitrary rows would be possible (the exclusive sets form a partition of  $V$ ; when aggregating two or more of these intersections by set union, the resulting family of sets is still a partition of  $V$ ). However, it would be challenging to design an easy-to-use interface for such a versatile aggregation mechanism.

**Selection.** We have restricted the selection mechanism  to two groups. Selecting only one group would not have allowed to compare groups, but just to contrast the elements of the group to all other elements. Supporting the selection of more than two groups of elements might be desirable, but cannot be easily implemented based on a color-based selection highlighting—the number of required colors grows exponentially with the number of selected groups because all possible overlaps between the selections need to be considered. If the selected groups do not significantly overlap, it might be sufficient to assign one color per group and an additional one for all overlaps. However, in the examples we studied, the overlap of the intersections was of particular interest. Alternatively, a selection can be handled just as a static or dynamic set and included as such into the main visualization. This would support selection of arbitrary numbers of groups, however, comes at the cost of significantly changing the streams with every new selection and potentially destroying the users' mental map of the visualization.

**Scalability.** Our approach scales well with the number of elements as demonstrated with hundreds of elements. Only reading and selecting thin lines representing few elements becomes difficult. With respect to time, like other approaches based on Sankey diagrams on a timeline , we can show about a dozen timesteps before readability gets affected. However, a rapid temporal scrolling technique [BBV\*12] could be implemented for longer time sequences. The most limited dimension of scalability is the number of base sets, especially if there exists overlap between them in various combinations. The worst case is that we need to show all  $2^n$  exclusive intersections. This is a typical problem of set visualizations. For instance, region-based or line-based overlay techniques [AMA\*16, Section 4.2] do not scale any better if sets significantly overlap. Some approaches circumvent the problem by just showing overlaps of maximum two sets, but this limits the analytical power. Interactive filtering techniques can be explored to reduce the number of elements and set intersections. An interactive aggregation approach (see above) might also improve this aspect of scalability.

**Applicability.** We have kept the approach general to show that the idea is applicable in diverse scenarios. The only exception with a slight scenario-specific adaption is the details panel, which shows different content depending on the loaded dataset. For real-word usage of the approach, we would still recommend tailoring the approach to the application at hand—only through this, the full potential of the approach can be used, such as showing application specific statistics or image thumbnails in the intersection nodes as

part of the stream visualization . Target users of the approach could include professionals of diverse kinds, for instance, committee members of academic conference to find reviewers based on their publication pattern, software maintainers of a repository, and machine learning practitioners, among others.

## 7. Conclusion

We have presented a novel technique for visualizing dynamic sets. According to our design considerations, it provides an overview of time on a timeline , where the flow of elements between set intersections is indicated through branching and merging streams  and groups of elements can be interactively selected for comparison . While the application examples have demonstrated the versatility and usefulness of the approach, we rather consider it as a base technique that can be extended in various directions. Besides tailoring it to specific applications, open research challenges involve alternative and extended strategies for ordering and aggregations of set intersections as well as for multi-group selections.

## Acknowledgments

Shivam Agarwal is indebted to Ankita Bajpai for her feedback on initial designs of the visualization. The authors are thankful to the participants of the expert study and anonymous reviewers for their valuable feedback. This work has been partly funded by Deutsche Forschungsgemeinschaft (DFG) part of research grant 288909335.

## References

- [AB20] AGARWAL S., BECK F.: Supplementary material for Set Streams: Visual exploration of dynamic overlapping sets, 2020. doi:10.17605/OSF.IO/86BKJ. 2, 7
- [AMA\*16] ALSALLAKH B., MICALLEF L., AIGNER W., HAUSER H., MIKSCH S., RODGERS P.: The state-of-the-art of set visualization. *Computer Graphics Forum* 35, 1 (2016), 234–260. doi:10.1111/cgf.12722. 2, 8
- [BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum* 36, 1 (2017), 133–159. doi:10.1111/cgf.12791. 2, 3
- [BBV\*12] BECK F., BURCH M., VEHLow C., DIEHL S., WEISKOPF D.: Rapid serial visual presentation in dynamic graph visualization. In *Proceedings of the 2012 IEEE Symposium on Visual Languages and Human-Centric Computing* (2012), VL/HCC, IEEE, pp. 185–192. doi:10.1109/vlhcc.2012.6344514. 8
- [BMBW15] BURCH M., MUNZ T., BECK F., WEISKOPF D.: Visualizing work processes in software engineering with Developer Rivers. In *Proceedings of the 3rd IEEE Working Conference on Software Visualization* (2015), VISSOFT, IEEE, pp. 116–124. doi:10.1109/VISSOFT.2015.7332421. 3, 6
- [CPC09] COLLINS C., PENN G., CARPENDALE S.: Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1009–1016. doi:10.1109/TVCG.2009.122. 2
- [GK10] GRAHAM M., KENNEDY J.: A survey of multiple tree visualisation. *Information Visualization* 9, 4 (2010), 235–252. doi:10.1057/ivs.2009.29. 2
- [IHK\*17] ISENBERG P., HEIMERL F., KOCH S., ISENBERG T., XU P., STOLPER C. D., SEDLMAIR M. M., CHEN J., MÖLLER T., STASKO J.: Vispubdata.org: A metadata collection about IEEE Visualization (VIS) publications. *IEEE Transactions on Visualization and Computer Graphics* 23, 9 (2017). doi:10.1109/TVCG.2016.2615308. 6
- [LGS\*14] LEX A., GEHLENBORG N., STROBELT H., VUILLEMOT R., PFISTER H.: UpSet: visualization of intersecting sets. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 1983–1992. doi:10.1109/TVCG.2014.2346248. 2
- [MRS\*13] MEULEMANS W., RICHE N. H., SPECKMANN B., ALPER B., DWYER T.: KelpFusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics* 19, 11 (2013), 1846–1858. doi:10.1109/TVCG.2013.76. 2
- [MWTI19] MIZUNO K., WU H.-Y., TAKAHASHI S., IGARASHI T.: Optimizing stepwise animation in dynamic set diagrams. *Computer Graphics Forum* 38, 3 (2019), 13–24. doi:10.1111/cgf.13668. 2
- [NXWW16] NGUYEN P. H., XU K., WALKER R., WONG B. W.: Time-Sets: Timeline visualization with set relations. *Information Visualization* 15, 3 (2016), 253–269. doi:10.1177/1473871615605347. 2
- [RB10] ROSVALL M., BERGSTROM C. T.: Mapping change in large networks. *PLoS one* 5, 1 (2010), e8694. doi:10.1371/journal.pone.0008694. 2, 3
- [STT81] SUGIYAMA K., TAGAWA S., TODA M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics* 11, 2 (1981), 109–125. doi:10.1109/tsmc.1981.4308636. 8
- [TA08] TELEA A., AUBER D.: Code Flows: Visualizing structural evolution of source code. *Computer Graphics Forum* 27, 3 (2008), 831–838. doi:10.1111/j.1467-8659.2008.01214.x. 2, 6
- [TMB02] TVERSKY B., MORRISON J. B., BETRANCOURT M.: Animation: can it facilitate? *International Journal of Human-Computer Studies* 57, 4 (2002), 247–262. doi:10.1006/ijhc.2002.1017. 3
- [VBAW15] VEHLow C., BECK F., AUWÄRTER P., WEISKOPF D.: Visualizing the evolution of communities in dynamic graphs. *Computer Graphics Forum* 34, 1 (2015), 277–288. doi:10.1111/cgf.12512. 2, 8
- [VBP\*20] VALDIVIA P., BUONO P., PLAISANT C., DUFOURNAUD N., FEKETE J.-D.: Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). doi:10.1109/TVCG.2019.2933196. 2
- [VBW16] VEHLow C., BECK F., WEISKOPF D.: Visualizing dynamic hierarchies in graph sequences. *IEEE Transactions on Visualization and Computer Graphics* 22, 10 (2016), 2343–2357. doi:10.1109/TVCG.2015.2507595. 2
- [VLBA\*12] VON LANDESBERGER T., BREMM S., ANDRIENKO N., ANDRIENKO G., TEKUSOVA M.: Visual analytics methods for categoric spatio-temporal data. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology* (2012), VAST, IEEE, pp. 183–192. doi:10.1109/VAST.2012.6400553. 2