EuroVis Workshop on Visual Analytics (2022)
M. Angelini and J. Bernard (Editors)

A Pipeline for Tailored Sampling for Progressive Visual Analytics

Marius Hogräfer[†1], Jakob Burkhardt[1], and Hans-Jörg Schulz[‡1]

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

Abstract

Progressive Visual Analytics enables analysts to interactively work with partial results from long-running computations early on instead of forcing them to wait. For very large datasets, the first step is to divide that input data into smaller chunks using sampling, which are then passed down the progressive analysis pipeline all the way to their progressive visualization in the end. The quality of the partial results produced by the progression heavily depends on the quality of these chunks, that is, chunks need to be representative of the dataset. Whether or not a sampling approach produces representative chunks does however depend on the particular analysis scenario. This stands in contrast to the common use of random sampling as a "one-size-fits-most" approach in PVA. In this paper, we propose a sampling pipeline and its open source implementation which can be used to tailor the used sampling method for an analysis scenario at hand. This pipeline consists of three configurable steps – linearization, subdivision, and selection – and for each, we propose exemplar operators. We then demonstrate its utility by providing tailored samplings for three distinct scenarios.

CCS Concepts

• ***Human-centered computing*** → ***Visual analytics;*** • ***Theory of computation*** → *Sketching and sampling;*

1. Introduction

A common challenge for interactive visual analysis is bringing the user into the loop during long-running computations [EHR*14, MPG*14]. A promising solution to this challenge is Progressive Visual Analytics (PVA) [SPG14, FFNS18]. Therein, analysts are presented with intermediate, incomplete results from these long-running computations, allowing them to gather early insights rather than having to wait until all data is fully processed. It has been shown that analysts in progressive systems often outperform those of non-progressive systems in terms of efficiency [ZGC*17].

One common way of generating partial results is by chunking up the data into smaller pieces, and then incrementally computing and visualizing results for these pieces over time. In practice, this seemingly simple process turns out to be rather complex, since splitting up the data means that the interactive visual analysis process – inherently an uncertain and often exploratory endeavor – becomes even more uncertain: any patterns that analysts find in the visualization could change in the future, once more data is processed. The general goal for sampling in PVA is thus to produce representative samples of the data that reduce the likelihood of the visualization changing over time. Yet, how exactly this "representativeness" is characterized depends on the analysis scenario. Prior work has thus proposed dedicated sampling methods that produce representative samples in particular scenarios. For example, Chen et al. present a method that preserves the local density and outliers in the visualization, as well as temporal coherence of subsequent chunks when progressively sampling for scatter plots [CZF*22]. Others have explored sampling methods for spatiotemporal data [WGT*20] or for prioritizing salient features of the visualization in the sampling [RAK*17]. Nevertheless, the default sampling method for scenarios, for which no such dedicated approach exists, remains random sampling. There are however some downsides to drawing random samples: (1) it can produce misleading visual artifacts on some visualizations [ZOLP17], (2) it can be a poor fit for tasks like outlier detection, where a few rare items in the data are actually of interest [CZF*22], and (3) it can fail to represent class distributions of imbalanced datasets.

We thus propose a sampling pipeline for PVA that can be configured to better fit the particular analysis scenario than random uniform sampling, providing a fallback option for analysis scenarios not yet covered by dedicated solutions. We introduce ProSample, a comparison tool for pipeline configurations, which allows comparing how the progressive results differ between two pipelines. Using ProSample, we showcase our sampling pipeline for three scenarios.

2. Related Work

Dividing a dataset into chunks is an integral part to the PVA process and "getting it right" is important, since any flaws introduced by this chunking will be present at any downstream operation. For

† Email: mhograefer@cs.au.dk
‡ Email: hjschulz@cs.au.dk

DOI: 10.2312/eurova.20221079

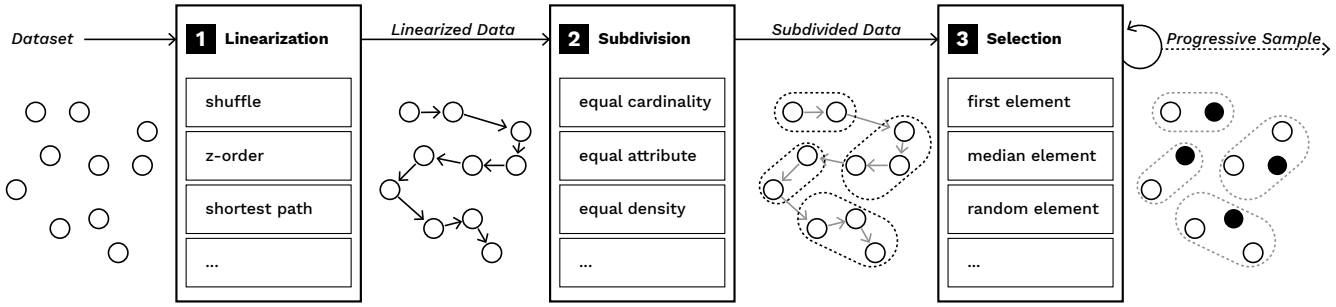

Figure 1: *Our sampling pipeline for Progressive Visual Analytics, including some exemplar operators used at each step: (1) The input data is first standardized into linear order in the linearization step, (2) then that data is structured into groups with the subdivision step, and lastly, (3) elements from these groups are then picked in the selection step to form a chunk that is representative in the context of the analysis scenario.*

instance, Procopio et al. note the role sampling plays in reducing the error bars in the visualization, i.e., the uncertainty of the analysis results that analysts work on [PMS*21]. Given this importance, it is surprising that the most commonly used sampling method remains random sampling, or shuffling the input dataset once and then chunking it in order, in cases where the data size causes random sampling to take too long.

Some prior work in the field of PVA has looked at specific sampling scenarios, such as the work by Chen et al. on progressive scatterplots that we already mentioned in the Introduction [CZF*22]. Turkay et al. propose an adaptive sampling for progressive visualization of high-dimensional data, which dynamically adjusts the size of the (in their case random) sample to ensure that the computation produces new results within a certain time interval [TKBH17]. Another example is the selective wander join method proposed by Procopio et al., which addresses the challenges of sampling for complex database queries that contain data joins, i.e., the data is retrieved from multiple tables at the same time [PSWC19]. The method also lends itself for progressively sampling from groups of skewed data, which can be desirable for prioritizing data of interest.

Under the term "approximate query processing" (AQP) and "online sampling", dedicated sampling methods have been developed to better capture data qualities like skew or error bounds. Sample+Seek [DHC*16] and BlinkDB [AMP*13] are examples of sampling methods for reducing or bounding errors and response times of queries on large datasets. An example for applying AQP in PVA is so-called optimistic visualization [MFDW17], which helps analysts recover from false conclusions drawn from the approximate data. This is done by visualizing the difference between the approximate result at the time analysts drew their conclusions and the current state of the progressive computation, allowing analysts to identify any discrepancies. Others use features of the visualization to optimize the sampling. Rahman et al. present a sampling method that aims to retrieve data for salient features in the visualization first, then sampling less salient features [RAK*17]. Park et al. present visualization-aware sampling, which uses prior knowledge about the visual encoding (scatter plots or maps) to produce an appropriate sample from large datasets, suggesting that running their algorithm multiple times may be beneficial to incremental visualization [PCM16]. Wang et al. present STULL, a progressive sampling method for spatiotemporal data that retrieves samples that show the same distributions as in the full dataset [WGT*20].

3. A Sampling Pipeline for PVA

On one hand, the related work shows how important it can be to have specific sampling techniques tailored to a particular scenario at hand. On the other hand, it also shows that we are far from having such a specific sampling technique for all possible scenarios. This is why we introduce a sampling pipeline for PVA, which is shown in Fig. 1. The main design goal of this pipeline is flexibility – i.e., to be adaptable or configurable to fit a wide range of possible sampling scenarios. Yet at the same time, we also want to be able to reuse parts of a sampling technique across different scenarios. Hence, we strive for a standardized process where we make as little assumptions about the analysis scenario as possible apart from the format of the inputs and outputs of each step in the pipeline. The pipeline steps can then be concretized using different operators that are reusable across different scenarios [HA06].

Our proposed sampling pipeline for PVA consists of three steps: linearization (which standardizes the input data), subdivision (which structures the linearized data), and selection (which generates the chunks from that structure). We detail these three steps below and provide general considerations for configuring each step.

3.1. Linearization

The linearization step standardizes the input data into a simple list of elements. This is necessary, as the type of the dataset influences the way it can be processed (e.g., graph data requires different algorithms than tabular data or geospatial data). By standardizing the input data, we essentially remove these specific characteristics from the input dataset, reducing the assumptions we have to make about the data at larger stages in the pipeline. This increases both the flexibility and reusability of downstream operators.

The proposed linear structure is conceptually simple and most data types can be transformed into it: a graph can be linearized by traversing it in order of its shortest path [ORS07], geospatial

and volumetric data can be linearized using space-filling curves like the z-order [ZOLP17, ZZY*20] or the Hilbert [DDW14, WFG*19, ZJW21] curve, hierarchies can be linearized using a BFS or DFS traversal, and n-dimensional data can be linearized using a knn-based heuristic to solving a traveling salesman problem over the data [JM07]. The concrete operators used in this step are thus very much dependent on the type of the input data. Once in linear form, it is further possible to reorder the data to account for requirements of the analysis scenario. For example, we can randomly shuffle the list to overcome sorting biases, or purposefully sort the list by some attribute value. Having the data in a certain order can be beneficial for operators in later steps of the pipeline, yet it comes at the cost of additional computation time.

3.2. Subdivision

The subdivision step takes the linearized data and partitions it. This can be seen as cutting-up the long list of all data into a set of smaller consecutive lists of data items.

This subdivision is very much dependent on the task to be carried out. If the task is still unspecified, we can simply subdivide the data into sets of equal cardinality – i.e., the same number of items in them. For an overview task that is to first show the extent of the data and not so much its density [Shn96], we can subdivide the data into sets that maximize coverage over a given value range. Depending on which attribute's value range is used, this puts additional focus on a data dimension of interest. When exploring spatially clustered data items, we can run Lloyd's algorithm on the linearized data [Llo82], essentially computing a 1D k-means clustering on it. The number of groups we subdivide the data into is scenario-specific. For example, if analysts want to just get a rough overview of the data, we can set the cardinality of the groups so that the progression produces results within acceptable response times. When using a nominal dimension to facet the data into groups, the number of facets is an appropriate fit. When considering multiple dimensions, we might use the number of classes found by the clustering algorithm as number of groups. A hierarchical subdivision (e.g., first dividing by coverage for one data attribute and then subdividing each set further by another data attribute) is possible.

3.3. Selection

The selection step then defines a strategy for constructing chunks from the structured data that most benefit the analysis scenario. To that end, it subsequently selects items from the subdivisions that best match the user interest in the data.

The choice of a selection operator depends in many ways on the user role as it is defined by Micallef et al. [MSA*19]. A *progressive observer* who monitors an evolving progressive visualization is interested in seeing a reasonable representative of the full dataset at any time. In this case, the selection could thus simply draw the medians from each subdivision. This is different for a *progressive searcher*, who uses the progression to quickly find an answer without having to look at all the data. If the searcher is interested in extreme values, the selection should draw min/max values from each subdivison. If the searcher is interested in the largest clusters, the selection operator should draw exclusively from the largest subdivision generated by the k-means operator mentioned above and then work its way downward to the smaller clusters. Finally, the *progressive explorer* uses PVA to be able to quickly switch between different configurations at runtime, depending on observations and insights gained from the partial results. Hence it is not a single selection strategy in which the explorer is interested, but in the ability to switch between different selections and their parametrizations to adjust the incoming data chunks to their current needs. For example, the size of the selection (and thus of the resulting chunks) may need to be changed to meet a desired response time. The statistical characteristics of the selection can be varied as well, either to yield selections that are (close to) representative of the entire dataset, or that are purposefully biased in order to retrieve certain data first, thus steering the progression towards data of interest [ED02, CKBE19].

4. Usage Examples of the Sampling Pipeline

We showcase the versatility of our approach by applying it to a use case for which we configure and compare three sampling pipelines. To that end, we make available an open source implementation of our pipeline, called *ProSample*, which allows analyzing the effect of different pipeline configurations on the PVA process (see `https://github.com/vis-au/prosample`). In ProSample, the user can configure two sampling pipelines, which are then used to simultaneously process a dataset, showing the results in side-by-side views as regular or binned scatter plots. An optional third view encodes the delta between the two views in a binned scatter plot. The views can be explored with zoom and pan, using linked navigation. The interface of ProSample is implemented using D^3 [BOH11] and runs in current browsers. Our implementation of the backend providing the configurable pipelines is done in Python, using the numpy package [VCV11].

4.1. Scenarios

The scenarios are based on a dataset of mountain peaks from OpenStreetMap [Ope], which contains about 650,000 items, providing longitude, latitude, and number of edits. For comparability, we sample one element from each group of an `equal cardinality` subdivision into 10,000 items, and the selection operations consider only the dimension containing the number of edits. We precomputed linearizations.

We begin by configuring the sampling for an **overview** task. We use the `random` linearization to transform the spatial dataset to a list and structure it into groups using the `equal cardinality` subdivision, since we are still unfamiliar with the underlying dataset. As selection method we set the `random` strategy, so that items per chunk are randomly picked from each subdivision group. We can see how the progressive visualization in ProSample early on shows the outlines of the continents, and we can also quickly identify regions on the map that contain many mountain peaks, such as Central Europe, the Himalayas, and the Andes (see highlighted regions in Fig. 2). If we were unfamiliar with the dataset, the first observation lets us quickly notice that the dataset contains spatial data, collected on a global scale, and the second observation allows us to identify that the measured points are mountain peaks. Based on these insights, we move on to the next scenario, where

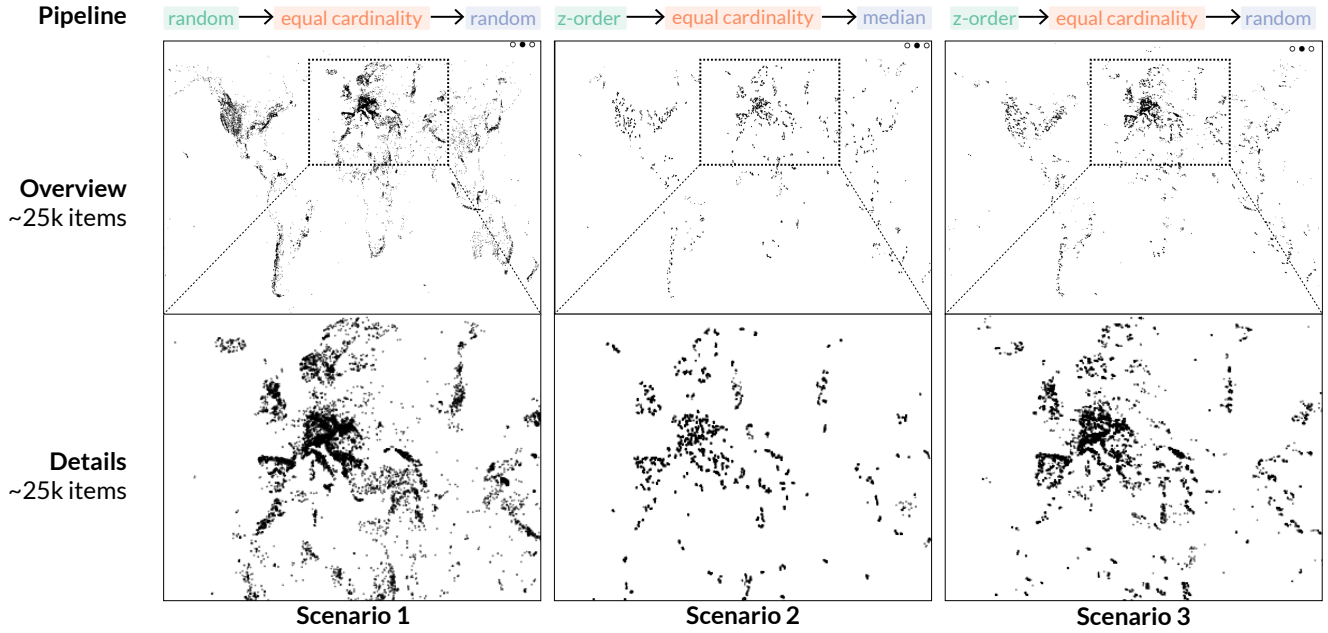

Figure 2: *The three scenarios discussed in the paper, showing the used pipeline configuration in terms of linearization, subdivision, and selection strategies, as well as an overview and detail view of the mountain peaks dataset after processing around* 25 *thousand items.*

we tailor the pipeline for a **density** analysis, in which we look first and foremost for regions with many mountain peaks. We do so by using a `z-order` linearization, which maintains spatial proximity between items in the standardized order, meaning that subsequent points in that list not are indeed located close to each other. Thus, when using the `median` selection operator, we will sample dense regions of the data. We notice that the data in early chunks is "clumped" into highly dense regions, which is exactly what we wanted, but it misses the contextual information of the sparser regions. To also yield this **context**, we adjust the configuration again by exchanging the selection step for the `random` operator, which selects items across the groups defined by the subdivision. We can see the effect in the zoomed-in views in Fig. 2, in that the sampled items are less "clumped" as before. With this context, we can for example identify the Alps in the context of Central Europe.

5. Conclusion and Future Work

We have presented a sampling pipeline for PVA, which can be used to tailor the sampling process to the scenario at hand. We demonstrated the flexibility of this pipeline in three scenarios, through the comparison tool ProSample that is available as open source.

Having a framework for tailorable PVA sampling in place opens up space for future applications. For instance, we want to explore guidelines for choosing the most beneficial sampling strategy for a particular PVA scenario, that go beyond general considerations as we outlined in Sec. 3. This will require more qualitative, and certainly more quantitative evaluation of our framework. Qualitative evaluations could widen the scope to also consider the dedicated progressive sampling techniques presented mostly from database perspectives (see Sec. 2), while quantitative evaluations could measure the performance of progressive sampling using our framework, compared to these existing techniques. Future work also needs to improve the practical implementation of our framework, extending the operators that are available so far in ProSample. Tool support is a general challenge in PVA, with only few research-focused frameworks like ProgressiVis [Fek15] and P5 [LM20] in existence, and therefore most research being conducted on custom implementations. A progressive sampling library could however be a starting point towards more reusable solutions. Lastly, we also want to expand on the principal considerations of sampling in PVA. In this paper, we have discussed sampling solely as a method for dividing the input dataset into smaller chunks that are then passed downstream to a PVA pipeline. However, given the thread-based process model presented by Schulz et al. [SASS16], dividing the data into chunks could also happen at different points in the PVA pipeline. In fact, their model suggests that each operator in the progressive pipeline can freely decide on when to process its input. This raises research questions regarding appropriate sampling pipelines dedicated for the view rendering step, best practices for combining operators using different sampling pipelines, or even having more than one pipeline per operator. As each operator only has access to parts of the full dataset, an important question that arises is how our pipeline can be extended to account for this. The naïve way is to linearize and subdivide in batches, yet that may be detrimental to the performance, so alternative strategies like using a dynamic tree structure to organize the subdivisions deserve consideration. Our pipeline provides parts of the groundwork towards these questions.

Acknowledgements

This work has been funded in part by the Innovation Fund Denmark through the Grand Solution project *Hospital@Night*.

References

[AMP*13] AGARWAL S., MOZAFARI B., PANDA A., MILNER H., MADDEN S., STOICA I.: BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *Proc. of EuroSys* (2013), ACM, pp. 29–42. doi:10.1145/2465351.2465355. 2

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3 Data-Driven Documents. *IEEE TVCG 17*, 12 (2011), 2301–2309. doi:10.1109/TVCG.2011.185. 3

[CKBE19] CUI Z., KANCHERLA J., BRAVO H. C., ELMQVIST N.: Sherpa: Leveraging User Attention for Computational Steering in Visual Analytics. In *Proc. of VDS* (2019), IEEE, pp. 48–57. doi:10.1109/VDS48975.2019.8973384. 3

[CZF*22] CHEN X., ZHANG J., FU C.-W., FEKETE J.-D., WANG Y.: Pyramid-based Scatterplots Sampling for Progressive and Streaming Data Visualization. *IEEE TVCG 28*, 1 (2022), 593–603. doi:10.1109/TVCG.2021.3114880. 1, 2

[DDW14] DEMIR I., DICK C., WESTERMANN R.: Multi-Charts for Comparative 3D Ensemble Visualization. *IEEE TVCG 20*, 12 (2014), 2694–2703. doi:10.1109/TVCG.2014.2346448. 3

[DHC*16] DING B., HUANG S., CHAUDHURI S., CHAKRABARTI K., WANG C.: Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. In *Proc. of SIGMOD* (2016), ACM, pp. 679–694. doi:10.1145/2882903.2915249. 2

[ED02] ELLIS G., DIX A.: Density control through random sampling: an architectural perspective. In *Proc. of IV* (2002), pp. 82–90. doi:10.1109/IV.2002.1028760. 3

[EHR*14] ENDERT A., HOSSAIN M. S., RAMAKRISHNAN N., NORTH C., FIAUX P., ANDREWS C.: The human is the loop: new directions for visual analytics. *Journal of Intelligent Information Systems 43*, 3 (2014), 411–435. doi:10.1007/s10844-014-0304-9. 1

[Fek15] FEKETE J.-D.: ProgressiVis: a Toolkit for Steerable Progressive Analytics and Visualization. In *Proc. of the Workshop on Data Systems for Interactive Analysis* (Chicago, United States, 2015), pp. 1–5. URL: https://hal.inria.fr/hal-01202901. 4

[FFNS18] FEKETE J.-D., FISHER D., NANDI A., SEDLMAIR M.: Progressive data analysis and visualization. *Dagstuhl Reports 8*, 10 (2018), 1–40. doi:10.4230/DagRep.8.10.1. 1

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

[JM07] JOHNSON D. S., MCGEOCH L. A.: *Experimental Analysis of Heuristics for the STSP*. Springer, Boston, MA, 2007, pp. 369–443. doi:10.1007/0-306-48213-4_9. 3

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. doi:10.1109/TIT.1982.1056489. 3

[LM20] LI J. K., MA K.-L.: P5: Portable progressive parallel processing pipelines for interactive data analysis and visualization. *IEEE TVCG 26*, 1 (2020), 1151–1160. doi:10.1109/TVCG.2019.2934537. 4

[MFDW17] MORITZ D., FISHER D., DING B., WANG C.: Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. In *Proc of CHI* (2017), ACM, pp. 2904–2915. doi:10.1145/3025453.3025456. 2

[MPG*14] MÜHLBACHER T., PIRINGER H., GRATZL S., SEDLMAIR M., STREIT M.: Opening the Black Box: Strategies for Increased User Involvement in Existing Algorithm Implementations. *IEEE TVCG 20*, 12 (2014), 1643–1652. doi:10.1109/TVCG.2014.2346578. 1

[MSA*19] MICALLEF L., SCHULZ H.-J., ANGELINI M., AUPETIT M., CHANG R., KOHLHAMMER J., PERER A., SANTUCCI G.: The Human User in Progressive Visual Analytics. In *Proc. of EuroVis Short Papers* (2019), Eurographics, pp. 19–23. doi:10.2312/evs.20191164. 3

[Ope] OPENSTREETMAP: Mountain Peaks Data. downloaded 01-May-2021. URL: https://wiki.openstreetmap.org/wiki/Tag:natural=peak. 3

[ORS07] ONUS M., RICHA A., SCHEIDELER C.: Linearization: Locally Self-Stabilizing Sorting in Graphs. In *Proc. of the ALENEX Workshop* (2007), SIAM, pp. 99–108. doi:10.1137/1.9781611972870.10. 2

[PCM16] PARK Y., CAFARELLA M., MOZAFARI B.: Visualization-aware sampling for very large databases. In *Proc. of ICDE* (2016), IEEE, pp. 755–766. doi:10.1109/ICDE.2016.7498287. 2

[PMS*21] PROCOPIO M., MOSCA A., SCHEIDEGGER C., WU E., CHANG R.: Impact of Cognitive Biases on Progressive Visualization. *IEEE TVCG* (2021), 1–13. doi:10.1109/TVCG.2021.3051013. 2

[PSWC19] PROCOPIO M., SCHEIDEGGER C., WU E., CHANG R.: Selective Wander Join: Fast Progressive Visualizations for Data Joins. *Informatics 6*, 1 (2019), 1–21. doi:10.3390/informatics6010014. 2

[RAK*17] RAHMAN S., ALIAKBARPOUR M., KONG H. K., BLAIS E., KARAHALIOS K., PARAMESWARAN A., RUBINFIELD R.: I've Seen "Enough": Incrementally Improving Visualizations to Support Rapid Decision Making. *Proc. of VLDB Endowment 10*, 11 (2017), 1262–1273. doi:10.14778/3137628.3137637. 1, 2

[SASS16] SCHULZ H., ANGELINI M., SANTUCCI G., SCHUMANN H.: An Enhanced Visualization Process Model for Incremental Visualization. *IEEE TVCG 22*, 7 (2016), 1830–1842. doi:10.1109/TVCG.2015.2462356. 4

[Shn96] SHNEIDERMAN B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proc. of VL* (1996), IEEE, pp. 336–344. doi:10.5555/832277.834354. 3

[SPG14] STOLPER C. D., PERER A., GOTZ D.: Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE TVCG 20*, 12 (2014), 1653–1662. doi:10.1109/TVCG.2014.2346574. 1

[TKBH17] TURKAY C., KAYA E., BALCISOY S., HAUSER H.: Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis. *IEEE TVCG 23*, 1 (2017), 131–140. doi:10.1109/TVCG.2016.2598470. 2

[VCV11] VAN DER WALT S., COLBERT S. C., VAROQUAUX G.: The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering 13*, 2 (2011), 22–30. doi:10.1109/MCSE.2011.37. 3

[WFG*19] WEISSENBÖCK J., FRÖHLER B., GRÖLLER E., KASTNER J., HEINZL C.: Dynamic Volume Lines: Visual Comparison of 3D Volumes through Space-filling Curves. *IEEE TVCG 25*, 1 (2019), 1040–1049. doi:10.1109/TVCG.2018.2864510. 3

[WGT*20] WANG G., GUO J., TANG M., QUEIROZ NETO J. F. D., YAU C., DAGHISTANI A., KARIMZADEH M., AREF W. G., EBERT D. S.: STULL: Unbiased Online Sampling for Visual Exploration of Large Spatiotemporal Data. In *Proc. of VAST* (2020), IEEE, pp. 72–83. doi:10.1109/VAST50239.2020.00012. 1, 2

[ZGC*17] ZGRAGGEN E., GALAKATOS A., CROTTY A., FEKETE J., KRASKA T.: How Progressive Visualizations Affect Exploratory Analysis. *IEEE TVCG 23*, 8 (2017), 1977–1987. doi:10.1109/TVCG.2016.2607714. 1

[ZJW21] ZHOU L., JOHNSON C. R., WEISKOPF D.: Data-Driven Space-Filling Curves. *IEEE TVCG 27*, 2 (2021), 1591–1600. doi:10.1109/TVCG.2020.3030473. 3

[ZOLP17] ZHENG Y., OU Y., LEX A., PHILLIPS J. M.: Visualization of Big Spatial Data using Coresets for Kernel Density Estimates. In *Proc. of VDS* (2017), IEEE, pp. 23–30. doi:10.1109/VDS.2017.8573446. 1, 3

[ZZY*20] ZHOU Z., ZHANG X., YANG Z., CHEN Y., LIU Y., WEN J., CHEN B., ZHAO Y., CHEN W.: Visual Abstraction of Geographical Point Data with Spatial Autocorrelations. In *Proc. of VAST* (2020), IEEE, pp. 60–71. doi:10.1109/VAST50239.2020.00011. 3