Report from Dagstuhl Seminar 18411

Progressive Data Analysis and Visualization

Edited by

Jean-Daniel Fekete[1], Danyel Fisher[2], Arnab Nandi[3], and Michael Sedlmair[4]

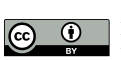
1 **INRIA Saclay – Orsay, FR, jean-daniel.fekete@inria.fr**
2 **Honeycomb – San Francisco, US, danyel@gmail.com**
3 **Ohio State University – Columbus, US, arnab@arnab.org**
4 **Universität Stuttgart, DE, michael.sedlmair@visus.uni-stuttgart.de**

Abstract

We live in an era where data is abundant and growing rapidly; databases storing big data sprawl past memory and computation limits, and across distributed systems. New hardware and software systems have been built to sustain this growth in terms of storage management and predictive computation. However, these infrastructures, while good for data at scale, do not well support exploratory data analysis (EDA) as, for instance, commonly used in Visual Analytics. EDA allows human users to make sense of data with little or no known model on this data and is essential in many application domains, from network security and fraud detection to epidemiology and preventive medicine. Data exploration is done through an iterative loop where analysts interact with data through computations that return results, usually shown with visualizations, which in turn are interacted with by the analyst again. Due to human cognitive constraints, exploration needs highly responsive system response times: at 500 ms, users change their querying behavior; past five or ten seconds, users abandon tasks or lose attention. As datasets grow and computations become more complex, response time suffers. To address this problem, a new computation paradigm has emerged in the last decade under several names: online aggregation in the database community; progressive, incremental, or iterative visualization in other communities. It consists of splitting long computations into a series of approximate results improving with time; in this process, partial or approximate results are then rapidly returned to the user and can be interacted with in a fluent and iterative fashion. With the increasing growth in data, such progressive data analysis approaches will become one of the leading paradigms for data exploration systems, but it also will require major changes in the algorithms, data structures, and visualization tools.

This Dagstuhl Seminar was set out to discuss and address these challenges, by bringing together researchers from the different involved research communities: database, visualization, and machine learning. Thus far, these communities have often been divided by a gap hindering joint efforts in dealing with forthcoming challenges in progressive data analysis and visualization. The seminar gave a platform for these researchers and practitioners to exchange their ideas, experience, and visions, jointly develop strategies to deal with challenges, and create a deeper awareness of the implications of this paradigm shift. The implications are technical, but also human–both perceptual and cognitive–and the seminar provided a holistic view of the problem by gathering specialists from all the communities.

Seminar October 7–12, 2018 – http://www.dagstuhl.de/18411
2012 ACM Subject Classification Human-centered computing → Visualization, Theory of computation → Models of computation, Computing methodologies → Machine learning, Software and its engineering → Software organization and properties
Keywords and phrases Approximate Query Processing, Online Aggregation, Exploratory Data Analysis, Visual Analytics, Progressive Data Analysis, Scalability
Digital Object Identifier 10.4230/DagRep.8.10.1
Edited in cooperation with Sriram Karthik Badam



1 Executive Summary

Jean-Daniel Fekete (INRIA Saclay – Orsay, FR, jean-daniel.fekete@inria.fr)
Danyel Fisher (Honeycomb – San Francisco, US, danyel@gmail.com)
Michael Sedlmair (Universität Stuttgart, DE, michael.sedlmair@visus.uni-stuttgart.de)

We live in an era where data is abundant and growing rapidly; databases to handle big data are sprawling out past memory and computation limits, and across distributed systems. New hardware and software systems have been built to sustain this growth in terms of storage management and predictive computation. However, these infrastructures, while good for data at scale, do not support data exploration well.

The concept of exploratory data analysis (EDA) was introduced by John Tukey in the 1970's and is now a commonplace in visual analytics. EDA allows users to make sense of data with little or no known model; it is essential in many application domains, from network security and fraud detection to epidemiology and preventive medicine. For most datasets, it is considered a best practice to explore data before beginning to construct formal hypotheses. Data exploration is done through an iterative loop where analysts interact with data through computations that return results, usually shown with visualizations. Analysts reacting to these results and visualizations issue new commands triggering new computations until they answer their questions.

However, due to human cognitive constraints, exploration needs highly responsive system response times (see https://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/): at 500 ms, users change their querying behavior; past five or ten seconds, users abandon tasks or lose attention. As datasets grow and computations become more complex, response time suffers. To address this problem, a new computation paradigm has emerged in the last decade under several names: online aggregation in the database community; progressive, incremental, or iterative visualization in other communities. It consists of splitting long computations into a series of approximate results improving with time; the results are then returned at a controlled pace.

This paradigm addresses scalability problems, as analysts can keep their attention on the results of long analyses as they arrive progressively. Initial research has shown promising results in progressive analysis for both big database queries and for machine learning.

The widespread use of progressive data analysis has been hampered by a chicken-and-egg problem: data visualization researchers do not have online database systems to work against, and database researchers do not have tools that will display the results of their work. As a result, progressive visualization systems are based on simulated systems or early prototypes. In many cases, neither side has currently incentive, skills, or resources to build the components needed.

Recently, data analysis researchers and practitioners have started conversations with their colleagues involved in the data analysis pipeline to combine their efforts. This standard pipeline includes the following core communities: data management, statistics and machine learning and interactive visualization. These initial conversations have led to fruitful evolutions of systems, combining two or three of these communities to complete a pipeline. Database and visualization have collaborated to create systems allowing progressive, approximate query results. Machine-learning and visualization have collaborated to create systems combining progressive multidimensional projections with appropriate scalable visualizations, such as Progressive t-SNE. Most current machine learning algorithms are designed to examine

the entirety of a dataset. A major contribution of work like Progressive t-SNE is to have a decomposable algorithm that can compute a meaningful partial result, which then can be passed on to a visual interface for fluent exploration. In these few existing collaborations, the researchers are able to work together and find concrete mechanisms by adapting existing systems for these without re-building them from the ground up. A systematic and widespread linkage between the involved communities, however, is still largely absent.

This Dagstuhl seminar brought the researchers and practitioners who have started this software evolutionary process to exchange their ideas, experience, and visions. We are convinced that in the forthcoming years, progressive data analysis will become a leading paradigm for data exploration systems, but will require major changes in the algorithms and data structures in use today. The scientific communities involved need to understand the constraints and possibilities from their colleagues to converge faster, with a deeper awareness of the implications of this paradigm shift. The implications are technical, but also human, both perceptual and cognitive, and the seminar will provide a holistic view of the problem by gathering specialists from all the communities.

This summary summarizes the outcomes of our seminar. The seminar focused on

- defining and formalizing the concept of progressive data analysis,
- addressing fundamental issues for progressive data analysis, such as software architecture, management of uncertainty, and human aspects,
- identifying evaluation methods to assess the quality of progressive systems, and threats to research on the topic,
- examining applications in data science, machine learning, and time-series analysis.

As a major result from the seminar, the following problems have been identified:

1. Implementing fully functional progressive systems will be difficult, since the progressive model is incompatible with most of the existing data analysis stack,
2. The human side of progressive data analysis requires further research to investigate how visualization systems and user interfaces should be adapted to help humans cope with progressiveness,
3. The potentials of progressive data analysis are huge, in particular it would reconcile exploratory data analysis with big data and modern machine learning methods,
4. Yet, there are many threats that should be addressed to bring progressive data analysis and visualization mainstream in research and application domains.

2 Table of Contents

3 Overview of Talks

3.1 Interactive Data Exploration: A Database Perspective

Carsten Binnig (TU Darmstadt, DE)

In recent years interactive data exploration (IDE) has gained much attention in both the database and visualisation community. Many specialised database execution engines have been developed to accommodate for and leverage the inherent workload characteristics of IDE. In this talk, I presented an overview of the existing techniques (e.g., for Approximate Query Processing) in databases to better support IDE workloads and discussed recent research results and presented challenges to be addressed by the database community.

3.2 Progressive Data Analysis and HCI

Adam Perer (Carnegie Mellon University – Pittsburgh, US)

In this lightning talk, I focus on two aspects of PDA and HCI: the design of PDA systems and user studies of PDA systems. Without PDA, data analysis workflows tend to force users to wait for analytics to complete. However, the design of PDA allows users to analyze and interpret partial results. I present early designs which give users two options: 1) trust the incremental result or 2) wait for everything to complete. However, further research has provided a set of design goals and requirements for PDA. Sampling the space of user studies of PDA systems suggest the need for more studies that take longer amounts of time and that utilize real analysts working on real problems, something missing from literature to date. In summery, there are many opportunities to further advance the design and evaluation of PDA systems.

3.3 Progressive Algorithms and Systems

Florin Rusu (University of California – Merced, US)

Joint work of Chengjie Qin, Yu Cheng, and Weijie Zhao
Main reference Yu Cheng, Weijie Zhao, Florin Rusu: "Bi-Level Online Aggregation on Raw Data", in Proc. of the 29th International Conference on Scientific and Statistical Database Management, Chicago, IL, USA, June 27-29, 2017, pp. 10:1–10:12, ACM, 2017.
URL https://doi.org/10.1145/3085504.3085514

In this lightning talk, we present two systems for parallel online aggregation and the sampling algorithms they implement in order to generate progressively more accurate results. The first system–PF-OLA (Parallel Framework for Online Aggregation)–defines a unified API for online estimation and provides a parallel/distributed implementation. The estimation is supported by several parallel sampling operators that make use of node stratification. We show how the PF-OLA API is used to detect better hyper-parameters for gradient descent

optimization applied to machine learning model training. The second system–OLA-RAW (Online Aggregation over Raw Data)–supports progressive estimation over raw files–not loaded inside the database. It does not require any data pre-processing, i.e., random shuffling. Estimation is supported by a novel bi-level sampling algorithm that is resource-aware, i.e., the number of rows processed (sampled) from a block is determined based on the system load. The "inspection paradox" inherent to parallel sampling is avoided by carefully inspecting blocks at certain time intervals. The common theme of these systems is that they allow for progressive data exploration. They have as a goal to find those queries that do not produce relevant results for the user. This can be done quite fast. We think this is a more realistic goal for progressive computation and visualization–steer the user away from uninteresting data.

3.4 Progressive Analysis–A VA+ML Perspective

Cagatay Turkay (City – University of London, GB)

Main reference Cagatay Turkay, Erdem Kaya, Selim Balcisoy, Helwig Hauser: "Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis", IEEE Trans. Vis. Comput. Graph., Vol. 23(1), pp. 131–140, 2017.
URL https://doi.org/10.1109/TVCG.2016.2598470

Progressiveness offer a fresh look at how visual analytics integrate machine learning algorithms within analysis processes and how machine learning algorithms can be designed and used in real-world settings. There are already several promising efforts both in visual analytics and machine learning literature that provide a basis for further research in progressive data analysis. In visual analytics, there are already frameworks, guidelines and also early examples of iterative machine learning algorithms embedded in visual analysis pipelines. In machine learning literature, the body of work on iterative learning models provide a strong basis to build upon. As expected from any emergent discipline, there are several challenges that lie at this intersection of incorporating machine learning methods in progressive settings, such as, the issue of concept drift, model adaptivity, e.g., in terms of complexity or flexibility, providing specific progressiveness qualities such as interactive steerability and convergent behaviour in terms of quality. On top of these algorithmic challenges, practical issues such as reproducibility and modelling provenance are critical for the future adoption of progressive methods. Considering the increasingly growing complexities of models and volume of data, and with all the various challenges that it brings with it, progressive analysis offers exciting opportunities for impactful, timely research.

3.5 A Visualization Perspective on Progressive Data Analysis

Marco Angelini (Sapienza University of Rome, IT)

Main reference Marco Angelini, Giuseppe Santucci, Heidrun Schumann, Hans-Jörg Schulz: "A Review and Characterization of Progressive Visual Analytics", Informatics, Vol. 5(3), p. 31, 2018.
URL https://doi.org/10.3390/informatics5030031

Progressive Data Analysis (PDA) is a novel analysis paradigm aimed at providing a result for long computational processes in the form of a sequence of approximating intermediate results, where the cause of the long time needed can be the complexity of the process and/or the amount of the data to process. Visualization plays a big role in allowing a human to evaluate, monitor and understand the results of a data analysis process, eventually steering it toward the desired goal. This lighting talk introduces the state of the art of Visualization and Visual Analytics for supporting PDA scenarios (PDAV or PVA) in terms of which characteristics govern the production of intermediate visualizations to evaluate partial results, workflow of analysis that a user follows interacting with a PVA system and differences with respect to a more traditional Visual Analytics system, management of additional like the control of the progression parametrization or the steering of the analysis, management of visually encoded uncertainty, stability, progress, and user's interaction. Important open problems, challenges and research directions conclude the talk.

3.6 Need for Progressive Analytics on Very Large Data Series Collections

Themis Palpanas

Main reference Kostas Zoumpatianos, Themis Palpanas: "Data Series Management: Fulfilling the Need for Big Sequence Analytics", in Proc. of the 34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018, pp. 1677–1678, IEEE Computer Society, 2018.
URL https://doi.org/10.1109/ICDE.2018.00211

Massive data series collections are becoming a reality for virtually every scientific and social domain, and have to be processed and analyzed, in order to extract useful knowledge. This leads to an increasingly pressing need, by several applications, for developing techniques able to index and mine such large collections of data series. It is not unusual for these applications to involve numbers of data series in the order of hundreds of millions to several billions, which are often times not analyzed in their full detail due to their sheer size. Bringing into the picture interactions with users and visualizations in support of explorative analytics introduces novel challenges related to the different kinds of queries (e.g., precise vs imprecise) and answers (e.g., exact vs approximate) that should be available to users, the amount of information to visualize, and the time performance of all these operations, which calls for progressive analytics in order to allow interactivity.

3.7 User Interface Elements and Visualizations for Supporting Progressive Visual Analytics

Sriram Karthik Badam (University of Maryland – College Park, US)

Joint work of Niklas Elmqvist, Jean-Daniel Fekete
Main reference Sriram Karthik Badam, Niklas Elmqvist, Jean-Daniel Fekete: "Steering the Craft: UI Elements and Visualizations for Supporting Progressive Visual Analytics", Comput. Graph. Forum, Vol. 36(3), pp. 491–502, 2017.
URL https://doi.org/10.1111/cgf.13205

Progressive visual analytics (PVA) has emerged in recent years to manage the latency of data analysis systems. When analysis is performed progressively, rough estimates of the results are generated quickly and are then improved over time. Analysts can therefore monitor the progression of the results, steer the analysis algorithms, and make early decisions if the estimates provide a convincing picture. In this talk, I will describe interface design guidelines for helping users understand progressively updating results and make early decisions based on progressive estimates. To illustrate the ideas, I will also present a prototype PVA tool called InsightsFeed for exploring Twitter data at scale. Our user evaluation showcased better usage patterns in making early decisions, guiding computational methods, and exploring different subsets of the dataset using the progressive user interface, compared to a sequential analysis without progression.

3.8 Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics

Charles D. Stolper (Google, US)

Joint work of Adam Perer, David Gotz
Main reference Charles D. Stolper, Adam Perer, David Gotz: "Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics", IEEE Trans. Vis. Comput. Graph., Vol. 20(12), pp. 1653–1662, 2014.
URL https://doi.org/10.1109/TVCG.2014.2346574

As datasets grow and analytic algorithms become more complex, the typical workflow of analysts launching an analytic, waiting for it to complete, inspecting the results, and then relaunching the computation with adjusted parameters is not realistic for many real-world tasks. This paper presents an alternative workflow, progressive visual analytics, which enables an analyst to inspect partial results of an algorithm as they become available and interact with the algorithm to prioritize subspaces of interest. Progressive visual analytics depends on adapting analytical algorithms to produce meaningful partial results and enable analyst intervention without sacrificing computational speed. The paradigm also depends on adapting information visualization techniques to incorporate the constantly refining results without overwhelming analysts and provide interactions to support an analyst directing the analytic. The contributions of this paper include: a description of the progressive visual analytics paradigm; design goals for both the algorithms and visualizations in progressive visual analytics systems; an example progressive visual analytics system (Progressive Insights) for analyzing common patterns in a collection of event sequences; and an evaluation of Progressive Insights and the progressive visual analytics paradigm by clinical researchers analyzing electronic medical records.

3.9 Optimistic Visualization and Progressive Interactions

Dominik Moritz (University of Washington – Seattle, US)

Joint work of Danyel Fisher, Dominik Moritz
Main reference Dominik Moritz, Danyel Fisher, Bolin Ding, Chi Wang: "Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data", in Proc. of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017., pp. 2904–2915, ACM, 2017.
URL https://doi.org/10.1145/3025453.3025456

Analysts need interactive speed for exploratory analysis, but big data systems are often slow. With sampling, data systems can produce approximate answers fast enough for exploratory visualization at the cost of accuracy and trust. We propose optimistic visualization, which lets users analyze approximate results interactively and detect and recover from errors later. It gives users interactivity and trust. In addition, progressive interactions let users start with low-resolution interactions and later interact at the pixel resolution.

3.10 Visplore–a highly responsive Visual Analytics Application

Thomas Mühlbacher (VRVis – Wien, AT) and Harald Piringer

Joint work of Visual Analytics Group at VRVis
Main reference Harald Piringer, Christian Tominski, Philipp Muigg, Wolfgang Berger: "A Multi-Threading Architecture to Support Interactive Visual Exploration", IEEE Trans. Vis. Comput. Graph., Vol. 15(6), pp. 1113–1120, 2009.
URL https://doi.org/10.1109/TVCG.2009.110

In my brief talk, I presented a highly responsive Visual Analytics software called "Visplore", which has been developed at VRVis since 2005. The software enables a flexible in-depth exploration of process- and timeseries data for process engineers, R&D staff and data scientists. Solutions based on Visplore are used by more than 10 companies in industries like automotive design, manufacturing, and energy transmission. The goal is to allow more in-depth analyses than business intelligence tools, while being simpler to use than standard statistics software. As one focus of the talk, I described the software's multi-threading architecture that enables highly responsive feedback for interaction, visualization, and computations based on millions of data records [1]. User interactions like brushing, for example, will immediately halt any ongoing computations, trigger invalidations of all dependencies, and continue computations based on the new input. As a second focus, I showcased a success story of Visplore from the manufacturing sector, where the high performance of the software has enabled process engineers to perform daily tasks of quality control and process operation based on millions of data records, leading to siginifcant cost savings and efficiency gains.

References

1 Harald Piringer, Christian Tominski, Philipp Muigg and Wolfgang Berger. *A Multi-Threading Architecture to Support Interactive Visual Exploration*. IEEE Transactions on Visualization and Computer Graphics, 15(6):1113–1120, Nov. 2009.

4 Working groups

4.1 Progressive Data Analysis and Visualization: Definition and Formalism

Charles D. Stolper (Google, US), Michael Aupetit (QCRI – Doha, QA), Jean-Daniel Fekete (INRIA Saclay – Orsay, FR), Barbara Hammer (Universität Bielefeld, DE), Christopher M. Jermaine (Rice University – Houston, US), Jaemin Jo (Seoul National University, KR), Gaëlle Richer (University of Bordeaux, FR), Giuseppe Santucci (Sapienza University of Rome, IT), and Hans-Jörg Schulz (Aarhus University, DK)

Big data, complex computations, and the need for fluent interaction are the source of challenges for contemporary visual analytics systems. Visual analytics systems must support a fluent, interactive flow between the human analyst, and visualizations and computational analysis; these challenges can impair this flow. To address them, the idea of Progressive Visual Analytics (PVA) [11, 4, 2] becomes increasingly appealing. A PVA approach can either subdivide the data under analysis in chunks, processing each data chunk individually, or it can subdivide the analytic process into computational steps that iteratively refine analytic results [10]. By doing so, PVA yields partial results of increasing completeness or approximate results of increasing correctness, respectively.

The expected benefits of introducing a progression within a Visual Analytics application include reducing latencies; allowing users to monitor the computational process; explaining the computational process through algorithm animation; reducing visual overload; and allowing the user to steer the running computation.

The Definition and Formalism group aimed to frame and formally characterize this emerging approach in the context of other well defined and related approaches.

4.1.1 Computation methods relevant in the progressive context.

Several approaches exist in the context of producing results from different data sources according to different constraints about quality, time and resources.These approaches can be described by their characteristics: by the data they deal with (i.e., bounded, unbounded), the way they digest such data (i.e., as a whole or in chunks), the number of outputs they produce, the flexibility they offer on response time, the measures they provide about uncertainty, and the possibility of being steered during their execution.

As an example, streaming computations deal with unbounded data and produce an unbounded number of results, while any-time computations deal with bounded data and are ready to produce a single output at user demand, output characterized by a quality depending on the computation time. Progressive computation, instead, deals with bounded data providing a converging sequence of output characterized by increasing quality, allow to be steered and are able to produce the first result in a previously specified time frame, characteristic that they share with bounded queries.

In order to better characterize a progressive computation we have investigated similarities and dissimilarities among various time and quality related computations (e.g., streaming data, online, approximation, anytime, iterative, etc.), characterizing them along input data, output, process, and accuracy. In the following section we analyze the most relevant proposals in the field, showing their properties and the relationships hey have with progressive computation.

4.1.2 Definition

A progressive process produces a sequence of useful/meaningful *intermediate results* in contrast with pure iterative processes that inherently generate intermediate results whose utility is mainly related to monitoring, tuning, steering, and explaining the process itself. The first result is constrained by a time value and must be be generated within that value. The sequence should be of increasing utility to the user, where utility can be defined according to the following aspects:

- actual result measures, like uncertainty, progression (data, time, iterations);
- comparison with previous result: delta in the data, (perceptual) delta in the visualization;
- analysis of fluctuating results that may be skipped to avoid confusing the user
- results that are produced at a too fast rate that may be delayed to not confuse the user with a too fast update generating a manageable rate for user understanding;

To sum up, a *progressive* computation is:

- a computation
  - **bounded on time and data**
- which **reports intermediate outputs**, namely
  - a result,
  - a measure of quality, and
  - a measure of progress,
- within **bounded latency**,
- **converging** towards the true result, and
- **controllable by a user during execution** either by
  - aborting, pausing, etc., or
  - by tuning parameters (steering).

4.1.3 Formalism & Comparison

As per the previous section, here we give more formal definitions with unified notations, of the computation approaches related to *Progressive* computation, namely: *Approximate*, *Optimistic*, *Online*, *Streaming* and *Iterative* computations.

Let F be a standard data analysis function with parameters $P = \{p_1, \ldots, p_n\}$. The computation of F takes a time t, starts with a machine state S_z, and modifies this state that becomes S_{z+1}. Also, for data analysis, we distinguish the parameters P from a set of input *data* D that the function takes as input. The function also yields a result vector R, and possibly a quality vector Q and a progression vector π:

$$F(S_z, P, D) \mapsto (S_{z+1}, R_z, t_z, Q_z, \pi_z) \tag{1}$$

The *eager* function F_e is the baseline that uses a fixed set of parameters P and can work on the full dataset D in one run in time t_D, yielding ideally the final exact result R_D with baseline quality Q_D.

To compare the different computation approaches, we focus on 3 properties of F: *accuracy*, *performance*, and *real-time* capabilities. Unlike its mathematical counterpart, the result R can suffer from *inaccuracies*[1]. Some methods are *real-time*, meaning that there is a mechanism to guarantee that the computation will complete within a specified constant time.

[1] "Accuracy" is defined by ISO 5725-1 [1] as "the closeness of a measurement to the true value"

Approximate Query Processing (AQP) returns either an approximate result unbounded quality in a user specified time θ (F_t) or an approximate result with bounded error ϵ in unbounded time (F_q).

Optimistic computation is a combination of AQP with bounded in time and unbounded quality in a first step and eager computation that yields the exact result R_D and baseline quality in the second step, in unbounded time, i.e., $t_D + \theta$ for AQP_t. It returns explicitly the progress π it made at the first and final steps.

Online computation is related to the way the data D are made available to the function F [3]. Informally, when an online function F_o is called iteratively with growing chunks of data D_z ($D_{z-1} \subset D_z \subset D$), it returns its results more accurately and more efficiently than if it were called directly from scratch with D_z. Online algorithms are popular in machine-learning and analytics in general, in particular to handle dynamic data.

Formally, the growth of the data tables can be modeled with a partition of D into n_C non-intersecting subsets or chunks C_i: $\bigcup C_i = D$, $C_i \neq \emptyset$ and $C_i \cap C_j = \emptyset$. At step z, F_o is called with $\bigcup_{j=1\ldots z} C_j$. Online functions guarantee that:

- **Accuracy** each result R_z is accurate and after running on all data chunks R_{n_C} is equal to R_D.
- **Time** the total time t_z to compute $F_o(S_z, P, \bigcup C_{1\ldots z})$ after R_{z-1} is much shorter than the total time t_D to compute $F_e(S_1, P, \bigcup C_{1\ldots z})$,
- **Unbounded time** however, online functions offer no guarantee that t_z is bounded.

4.1.4 Streaming methods

They are related to memory usage and time constraints [6]. According to Wikipedia: "*streaming algorithms are algorithms in which the input is presented as a sequence of items and can be examined in only a few passes (typically just one). These algorithms have limited memory available to them (much less than the input size) and also limited processing time per item.*"

Compared to an online function, a streaming function F_s only takes the chunks C_z and not the whole dataset D. The maximal size of a chunk can be limited by the algorithm.

Streaming functions properties are:

- **Accuracy** results R_i can be inaccurate so $R_z \approx R_D$,
- **Time** the total time t_z to compute $F_s(S_z, P, C_z)$ after R_{z-1} is much shorter than the total time to compute $F_e(S_1, P, \bigcup C_{1\ldots z})$, which would be $\sum_{j=1,z} t_j$,
- **Real-time** streaming functions are real-time so t_z is bounded,

Due to the real-time constraints, streaming algorithms can sometimes compute rough approximations of their results. Furthermore, streaming algorithms are usually difficult to design and to implement; relying on streaming algorithms only to implement an entire data analysis infrastructure severely limits the range of possible applications that could be built. For example, at the time of the writing on this document, the Spark streaming machine-learning library [8] provides 86 classes compared to the 245 provided by the scikit-learn machine-learning library [7].

Iterative computation is defined in Wikipedia as: "a mathematical procedure that generates a sequence of improving approximate solutions for a class of problems." An iterative function performs its computation through some internal iteration and can usually provide meaningful partial results R_j after each iteration, although some extra work might be required to obtain the partial result in an externally usable form [9].

Compared to an online function, an iterative function F_i always takes the whole dataset D and returns improving results R_z that eventually becomes the exact R_D. Iterative functions guarantee that:

- **Accuracy** the results R_i will converge to the real value R_D: $\lim_{j \mapsto \infty} R_j = R_D$, but the convergence may not improve at each step,
- **Time** the total time to compute R_D is usually controlled indirectly either through a tolerance parameter or a maximal number of iterations; none of these parameters allow analysts to predict the completion time,
- **Bounded time** iterative functions offer no guarantee that t_z is bounded.

Progressive computation is designed to deliver results for analysts at a controlled pace and with controlled accuracy. A progressive function F_p has a form very similar to the AQP_{time} function except it works on a subset $D_z \subset D$ of the full data with no constraint on D_z regarding any other $D_{j<z}$. In addition to the result and time, progressive function output explicitly quality Q and progress π.

Progressive functions guarantee that:

- **Accuracy** just like online and iterative functions, the results R_i will converge to the real value R_D. in a finite number Z of steps: $\lim_{j \mapsto Z} R_j = R_D$. Additionally, like streaming functions, each call will provide an estimate of the result R_i that can be inaccurate, but like iterative functions, the accuracy can be improved by calling F_p several times with the same input ($D_z = D_{z-1}$),
- **Time** progressive functions are similar to online functions when D_z is growing, and similar to iterative functions when D_z is constant,
- **Real-time** progressive functions offer the guarantee that t_z is bounded ($t_z \leq \theta$).
- **Steerable** Progressive functions parameter P_z can be modified between each call. This is called *computational steering* [5], and has emerged from analysts who could monitor the evolution of their computations by visualizing the progression, and then wanted to influence the computation without restarting from scratch when they realized that some parameters would have to be tuned.

The table 1 summarizes all these computations.

4.1.5 Open questions and Future work

Good quality metrics. In the formal definition we have defined quality as a simple scalar. The exact quality function depends very much on the progressive algorithm under consideration, but we can make a few statements on properties of ideal quality functions:

- They are a direct function of the distance of R_z to the end result. In practice, defining this function exactly is impossible since the end result is not (yet) known. Here, we will have to settle on an approximate quality function that is close enough to the actual quality. Depending on the definition of this approximate function Q', there will be additional uncertainty introduced by the difference between Q and Q' which may be variable. It is unknown how this influences user perception of quality.
- Consistent across a single computation: in order to accurately gauge the state of the computation, there should be a total order on the values given by the quality function. In other words, the quality should be comparable across the entire computation.
- They are strictly increasing: ideally a quality function strictly increases over the duration of the progressive algorithm. Depending on the progressive algorithm however, there may be situations where quality decreases over time (for example the algorithm is moving out of a local minimum solution). In these cases a user will have to decide on the suitability

Table 1 Comparison of computation methods properties.
R is the desirable result
t is the computation time
P are the algorithm parameters
Q is the quality
π is the progression
t_D is time taken on full dataset D.
C_i are chunks of the full data: $\bigcup_{z\in\{1\dots n_C\}} C_z = D$, $C_i \neq \emptyset$ and $C_i \cap C_j = \emptyset$.
$D_i \subseteq D$.
θ is a user-defined upper-bound on latency
Output is always: $(S_{z+1}, R_z, t_z, Q_z, \pi_z)$

	Input	R	t	P	Q	π
Eager	$F_e(S,P,D)$	Exact R_D	t_D	Fixed	Baseline	Irrelevant
AQP_t	$F_t(S,P,D,\theta)$	Apx $d(R,R_D) \geq 0$	$\leq \theta$	Fixed	Unbd	Irrelevant
AQP_q	$F_q(S,P,D,\epsilon)$	Apx $d(R,R_D) \leq \epsilon$	Unbd	Fixed	Bounded	Irrelevant
Optimistic	AQP→eager	$R_2 = R_D$	$\leq t_D + \theta$	Fixed	Baseline	Explicit
Online	$F_o(S_z,P,\bigcup C_{1,\dots,z})$	Exact for $z = n_C$	$\ll t_D$	Fixed	Baseline	Implicit
Streaming	$F_s(S_z,P,C_z)$	Apx $d(R_z,R_D) \geq 0$	$\ll t_D$	Fixed	Unbd	Implicit
Iterative	$F_i(S_z,P,D)$	$R_z \xrightarrow[z\to\infty]{} R_D$	Unbd	Fixed	Unbd	Implicit
Progressive	$F_p(S_z,P_z,D_z,\theta)$	$R_z \xrightarrow[z\to Z]{} R_D$	$\leq \theta$	Ctrled	Unbd	Explicit

of an intermediate result R_z based on the time taken, a visual assessment of R_z or the current quality relative to the initial quality.

Lastly, in some cases, a suitable quality function may simply not be available. In that case, the only measure of quality may be the algorithm's progression. Whether the algorithm can still be deemed progressive (as opposed to iterative) is an open question.

Good progress metrics. The progress π_z was defined abstractly as a variable between 0 and N. Since the computation is bounded, an upper bound N is known at the start of the computation, at least as a parameter of the algorithm itself. This implies we can always output an abstract progress indicator. While this abstract progress of the algorithm is strictly increasing, it is better to present the user with more actionable progress measures, such as time remaining until finish (can be determined by sampling) or percentage of records analyzed.

Semantics of algorithm parameters. In the formal definition we have assumed a parameter set P, containing parameters driving the algorithm. In practice, this set P will consist of parameters that can influence the convergence rate of the chosen algorithm (e.g., parameters that control a speed/quality trade-off internal to the algorithm) and parameters that can influence the end result (e.g., the number of clusters in a cluster operation). Changing one of the former should ideally allow the progressive algorithm to continue from its current state. Changing one of the latter typically necessitates a full recompute. Depending on the progressive algorithm in question, there may be cases where the last intermediate result R_z can be reused in the new progressive computation to try to maintain some continuity in the progression.

Latency bound. Choosing the latency bound between intermediate results (θ) is a trade-off between faster-coming results and more precise results. At the same time, it is closely related to both the user's visual processing capabilities and the rate of data processing. Indeed, ideally, θ is expected to lay under several seconds for the system to appear reactive to the

user, but still be large enough for the changes to be digestible to the user. Since producing intermediate results typically introduces an overhead, θ should fit the rate to which the underlying computation takes place to avoid producing successive results without changes as a result of overload instead of stabilization of the algorithm. Consequently, choosing the optimal value of θ relative to a particular algorithm and user task is an open question as well as the extents to which a user could interactively change it during progression.

Mapping to use-cases. While we have presented a semi-formal definition of what progressive computation is, and what its parameters are, we still need to validate this definition against real-world use cases. It would be a good exercise to take some of the use-cases presented in other teams and see how these fit the definitions presented above. Any mismatches can then be used to sharpen or correct the definition.

References

1 ISO 5721-1. Accuracy (trueness and precision) of measurement methods and results - Part 1: General principles and definitions, 1994.

2 Marco Angelini, Giuseppe Santucci, Heidrun Schumann, and Hans-Jörg Schulz. A review and characterization of progressive visual analytics. *Informatics*, 5(3):31, 2018.

3 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

4 Jean-Daniel Fekete and Romain Primet. Progressive analytics: A computation paradigm for exploratory data analysis. *arXiv.org e-print*, 1607.05162, 2016.

5 Jurriaan D. Mulder, Jarke J. van Wijk, and Robert van Liere. A survey of computational steering environments. *Future Gener. Comput. Syst.*, 15(1):119–129, February 1999.

6 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, August 2005.

7 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

8 N. Pentreath. *Machine Learning with Spark*. Community experience distilled. Packt Publishing, 2015.

9 Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

10 Hans-Jörg Schulz, Marco Angelini, Giuseppe Santucci, and Heidrun Schumann. An enhanced visualization process model for incremental visualization. *IEEE Trans. Vis. Comput. Graphics*, 22(7):1830–1842, 2016.

11 C.D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Trans. Vis. Comput. Graphics*, 20(12):1653–1662, 2014.

4.2 Tasks, Users, and Human Biases in PVA Scenarios

Hans-Jörg Schulz (Aarhus University, DK), Remco Chang (Tufts University – Medford, US), Luana Micallef (University of Copenhagen, DK), Thomas Mühlbacher (VRVis – Wien, AT), and Adam Perer (Carnegie Mellon University – Pittsburgh, US)

4.2.1 Introduction

This report describes the result of two breakout groups that focused on the role of human users in PVA scenarios. The first group, "Tasks and User" compiled a list of concrete **example usage scenarios** of PVA, intending to (1) raise awareness of PVA benefits for all communities based on tangible stories, and (2) to investigate recurring patterns in a bottom-up fashion. Based on these scenarios, the group formulated five **high level tasks** [T0-T4] that only become possible through the progressiveness of the described applications.

The second breakout group, "Humans in the progressive Loop", characterized different **types of users** of PVA applications. The discussion focused on how their backgrounds, skills, expectations and benefits from progressiveness differ, and which implications for the design of PVA applications arise. Moreover, the group built on the usage scenarios of the first breakout group to establish a more **structured characterization of PVA scenarios** regarding the dimensions concurrency, task dependency and interactivity. Finally, the effect of **human biases** that may occur in each scenario is discussed, to outline potential challenges and pitfalls for future developers of PVA systems.

4.2.2 Example Usage Scenarios / "Stories" (Adam Perer)

During our discussions, we established a set of usage scenarios to demonstrate the breadth and variety of progressive data analysis. In each usage scenario, we highlight the progressive aspects in **bold**. For brevity, we only describe a representative subset of our stories here:

4.2.2.1 Rapid Queries on Very Large Data

User wants to write query computing an average from a large amount of data, the **execution of which could take hours**.

- User runs the query against a progressive visualization system.
- At some point, first **intermediate result** is visualized
- From this, the user notices implausible / unexpected scaling of Y-Axis
- The user's initial thought is: "maybe the dataset seen yet is not representative"
- The user modifies the sampling to **prioritize data chunks** that she thought would be more representative.
- Still, the intermediate results are of the same strange scaling
- The user has the insight: "Oh, the query uses sum instead of average"
- Based on these preliminary results, user **cancels and restarts** the corrected query

However, not all initial queries tell the full story. Consider the following scenario where an inconsistency emerges after the initial approximation:

- The user wants to see what the origin and destinations of mid-range flights for Hawaiian airlines are.
- She **queries** for the histogram and **sees** that most flights are from California to Hawaii. This observation is based on the approximate result.

- Later she **runs the query over the complete data** – which takes a few minutes to complete – and she sees that she **missed a flight** from California to Pennsylvania. This flight was missing in the approximation and only included in the later result.

4.2.2.2 Model Search

Doctors wants to find the best model for predicting body mass index (BMI) from a list of clinical data to find external risk factors. **The optimization of hyperparameters using AutoML is very costly and often takes hours to days to execute**.

- The user initiates AutoML to build a regression model based on automatically selected features from the set of all available physical and diagnostic features that are available
- The system **progressively returns potential models** and **shows models' accuracies**
- User realizes that accuracy is "too good"; almost 100% accurate for all models
- She explores the correlations between features. By examining a scatterplot matrix she notices that the factors Weight and Height perfectly correlate with BMI.
- She realizes that she had included these two factors mistakenly as features (as BMI is a direct function of Weight / Height, but she wanted "external" risk factors).
- She changes the parameterization by removing these two variables, **retriggering the execution** of AutoML, which results in more informative/realistic models.

4.2.2.3 Exploratory Search

A user wants to find frequent event sequences in a large database of time-stamped events. She launches a frequent sequence mining algorithm, specifying a support threshold that defines common patterns that occur in at least 5% of the population. This particular algorithm returns short patterns first, as they are cheaper to compute, and then longer ones as the algorithm continues, in a breadth-first manner. This particular algorithm can run concurrently and independently, spread across multiple threads and servers.

After a few seconds, patterns start returning from the algorithm and she gets a ranked list of the patterns found so far. This helps her **get an overview** of the common patterns that the user was previously unaware of, as the data was too big to query directly. At this point, she is able to **start exploring** the frequent patterns already and learn what is common in its dataset. After looking at this list, she notices that one of the patterns is irrelevant to its goal, so **directs the algorithm** to not continue looking for longer patterns that begin with this sequence. Furthermore, she notices one short pattern that seems quite promising and is curious if there are longer patterns that begin with this sequence. She tells the system to **prioritize this search space**, and look for longer patterns that begin with this sequence. This progressive process gave her the opportunity to help guide the search algorithm, which gave her a better understanding of how the search algorithm worked, as well as being able to take advantage of her domain knowledge. The progressive process also gave the user to speed up the algorithm by focusing on relevant spaces.

4.2.2.4 Domain Example: Disaster Response

A crisis manager has to analyze a **vast amount of data to trigger emergency response** under time, human and equipment constraints. She collects millions of tweets relative to a crisis from diverse sources and other sensor data like UAV photographs, satellite images or seismic sensors, wind speed sensors. She analyzes the data to detect events that require emergency response. The data can all be stored, but their amount requires automatic

processing to extract the data relevant to the crisis manager. The data are not easy to analyze, some events on tweets can be rumors, some images can be difficult to interpret. Automatic processing is necessary but the more data it processes the more confident it is relative to the certainty of occurrence and the severity of an event. The user can select the type of event to focus on depending on their certainty and severity as evaluated **as per the current status of the progressive processing**, correcting badly classified events (named entities like location, person and organization in tweets) to guide the classifier in doing its task, triggering relevant emergency response as soon as she decides an event has reached her subjective threshold of certainty and severity.

4.2.3 High-Level Tasks / Benefits of PVA (Thomas Mühlbacher)

Based on the usage scenarios from the previous section, we extracted a set of recurring high-level tasks that were only made possible by the early involvement of users in the ongoing computations. The ability of performing each task can be seen as a concrete benefit added by the progressiveness. Fulfillment of the task list may also serve as a qualitative benchmark for the evaluation of future PVA architectures. The tasks are:

T0: Making decisions under time- and resource constraints
T1: Discovering wrong assumptions early to save time and not lose train of thought
T2: Rapid exploration of very large data / information spaces
T3: Incorporating human knowledge to steer algorithms and speed them up
T4: Building trust and deepening the understanding of algorithms

4.2.4 Characterization of Users (Hans-Jörg Schulz)

Users utilize PVA in different ways and for different objectives. Accordingly, we distinguish between three main types of users:

U0: "The Consumer" who makes decisions based on the output of a PVA model (usually without knowledge of that model or even PVA as a concept)
U1: "The Domain Expert" who makes decisions with the PVA model (domain-specific, but no CS/Stats knowledge of underlying model or architecture)
U2: "The Data Scientists" who makes decisions about the PVA model (does have CS/Stats knowledge, but not necessarily domain knowledge of underlying model or architecture)

These prototypical users perform different subsets of the established progressive tasks [T0-T4], use different aspects of PVA to perform these tasks, have different expectations on PVA, which in turn incur different properties of the UI in which they work with PVA.

4.2.4.1 U0: The Consumers

The consumers want to make decisions under time- and resource constraints [T0], for which they need to rapidly explore very large data / information spaces [T2], and discover wrong assumptions early to save time and not lose their train of thought [T1]. An everyday example is the use of a progressively loading online map to explore different routes from the current position to a desired goal. To do so, they make use of intermediate PVA results:

- to identify those assumptions that were wrong (e.g., map showing the wrong place),
- to assess the progress of the loading map (e.g., am I already looking at the finest resolution?) or the "aliveness" of the process (e.g., does the server still respond?), and
- to realize a simple "steering" in the sense of quickly correcting a wrong assumption by adjusting the focus of the PVA output (e.g., panning the map to a region of interest).

Their expectation of the underlying PVA process is that of a well-behaved, calmly converging output that arrives at a predictable end. The mental analogy of a "loading process" probably describes this expectation best. To have the consumers being able to use PVA right away – without hesitation or an extra learning curve for the progressiveness – it should ensure this expectation is met by means like:

- using/building on commonplace metaphors and familiar abstractions that are already part of the interface, e.g., the "online map application",
- reducing visual complexity, fluctuations, jumps, flickering or any other indication of the underlying process not calmly and predictably producing progressive outputs, and
- maintain the users' mental map, e.g., by introducing anchors or waypoints.

4.2.4.2 U1: The Domain Experts

The domain experts want to do all the things the consumers do, but on top they want to have more control over the internal model execution – not necessarily about what these models are, but how they are parametrized and utilized. In most cases, this means to steer the algorithm based on their knowledge and speed them up while already running [T3]. An example is an online user study run via Mechanical Turk that shows after the first 20+ responses that certain questions are formulated ambiguously. So, the domain expert can step in and adjust the question on the fly to get answers in the future that are more to the point. For doing this, intermediate PVA results are used:

- to identify assumptions that are wrong (e.g., about the choice of questions in the study, or the quality of the study results)
- to decide, whether to wait or step in (e.g., wait for 20+ more results or make the change now?)
- to steer the result towards the best match with the domain knowledge of the experts (e.g., for A/B testing alternative formulations for the identified problematic questions)

Their expectation of the underlying PVA is that of a predefined, but malleable process that can be tuned while running to overcome corner cases or to deal with unexpected complications. To do so, the UI needs to give access this adaptability of the process by means of:

- not glossing over, but reflecting – maybe even emphasizing – the fluctuations, jumps, and other indications of the underlying model not behaving as expected,
- exposing degrees of freedom and steering possibilities provided by the underlying model in a suitable language (e.g., building only on domain concepts and semantic interaction, rather than exposing statistical/algorithmic parameters directly), and
- providing a feeling of uncertainty that allows to judge the results from interactively performed parameter changes – also in a domain-specific manner.

4.2.4.3 U2: The Data Scientists

The data scientists need to be able to do everything the domain experts do, but as it is usually them who are building and optimizing the underlying models, they may also need to gain a deep understanding of the algorithms to increase their trust in the model [T4]. An example is the debugging of complex models and algorithms in multiple steps and with data of increasing sizes. To facilitate this, intermediate PVA results are used:

- to identify wrong assumptions early (e.g., by testing different models,)
- to recognize processes getting stuck in local optima, and

- to efficiently allocate time for different tasks involved in model building, testing, and revision.

Their expectation of PVA is that of a completely open, transparent process that can not only be reparametrized, but in which modules and methods can be exchanged at intermittent stages. For this, the UI must:

- expose all details, parameters and process metrics to the user on demand AND making them adjustable on the fly,
- incorporate as much as possible of statistics over the outputs and the process on demand, to judge stability and convergence of the method as a whole, and
- allow possible changes in a What-If scenario that branches off alternatives and compares them later.

These three user types rather describe a continuum with other possible types in between – e.g., a U1.5 "The Apprentice / Learner of Algorithms", either for curious/suspicious domain experts (U1) or for aspiring data scientists (U2), that monitor PVA processes without changing them. The task characterizing a U1.5 is [T4], i.e., to deepen the understanding and trust in algorithmic methods by looking at their progression. In addition, we note that any person can take on multiple user types of PVA during an analysis, as it may become necessary. It may thus also make sense in many application scenarios to refer to U0-U2 as "user roles" instead of "user types".

4.2.5 Structured Characterization of PVA Scenarios (Remco Chang)

We consider three dimensions to the use of progressive visualization: concurrency, task dependency, and interactivity. Concurrency refers to whether the user interacts with each task in a serial (blocking) or concurrent manner. Task dependency refers to whether the multiple tasks that the user is trying to perform have outcomes that depend on each other. Lastly, interactivity refers to whether the user is required to interact with the progressive visualization; and if so, if the user would interact with the final outcome of the computation or interact during the process of progression. Table 2 shows these dimensions, the types of usage scenarios that they most associate with (refer to Section 4.2.2 for more detail), as well as the most threatening biases (described in Section 4.2.6).

For example, when progressive visualization is used for **debugging** queries (see Section 4.2.2.1), the user would *not interact* with the progressive visualization (other than to terminate the query), the task of executing a query is *serial*. Task dependency in this case does not apply as debugging can be used in both contexts.

For **exploring** data, the user would *interact* with the outcomes of the progressive computation after the computation is complete or the user is satisfied with the partial result (as such, the process is *serial*). In contrast, for **model steering**, the user would interact during the progressive computation to guide the execution of a machine learning process.

Progressive visualization can also be used for managing or executing multiple tasks *concurrently*. In the most simple case, the user can **monitor** the progression of multiple tasks without interacting with the visualization. Further, the user could *interact* with the tasks. For example, the user could reduce the search space in an **exploratory search** task when multiple *independent* search processes are *concurrently* executed.

Lastly, we consider the case of a *concurrent*, *dependent*, and *interactive* progressive visualization usage scenario. This scenario has not been defined in Section 4.2.2 because few modern systems can support this without the use of a progressive database engine. For example, imagine that an analyst is looking for the "Top 10" customers in their database.

Table 2 Characterization of Usage Scenarios.

Usage Scenarios	Concurrency	Task Dependency	Interactivity	Biases (negative effect)
explain	serial	indep/dep	no	illusion of transparency
debugging	serial	indep/dep	no	anchoring, recency
explore	serial	indep/dep	with results	sunk cost fallacy
steering	serial	indep/dep	with results/ progression	illusion of control, confirmation bias, superiority bias
monitoring	concurrent	indep/dep	concurrent	conservatism, change blindness
exploratory search	concurrent	indep	with results/ progression	cognitive overload, inattentional blindness
continuous querying	concurrent	dep	with results/ progression	base rate fallacy

When this search query is being executed and the partial results are progressively streamed in, the analyst *concurrently* executes a join query to see the purchases of these customers before the first query completes. This capability has the potential to significantly speed up the user's performance since the user can begin to formulate new questions and queries without being *blocked* by the original query or queries. However, the execution of this query is different from what traditional databases support, in particular because the database engine will need to join two tables that are dynamically changing.

In summary, the use of progressive visualization affords the user in performing wide ranging usage scenarios, tasks, and strategies. However, with the benefits of progressive visualization also comes potential costs to the user. In the section below, we describe the cognitive biases that can most threaten the use of progressive visualization in each of the usage scenarios.

4.2.6 Biases in Human Reasoning (Luana Micallef)

A large number of factors can affect human reasoning. One widely studied phenomena is cognitive biases in human reasoning in general [3] and in visualization in particular [1, 2]: a systematic and involuntary way of how humans deviate from rational judgement, regardless of intelligence and domain expertise. The majority of the cognitive biases lead to inaccurate judgment and distorted perception. However, a small number of biases lead to a positive effect in certain analytic tasks, such as progressive analytics. It is important to be aware of these biases and their effect, so visualizations are carefully designed to reduce negative effects of biases, while attempting to fully exploit positive ones.

For instance, reasoning about probabilities is hard for humans, and under uncertainty, probabilities are either disregarded or hugely overrated (neglect of probability bias). Moreover, humans are insensitive to sample size and often disregard it when reasoning about probabilities. Both of these biases could have a large negative effect in progressive analytics due to the partial and uncertain results generated by a progressive system.

On the contrary, a few biases and effects support the benefits of progressive analytics. The generation effect indicates that humans can better recall information that they generate

than information that they read about. Thus, users that are active in progressive analytics are more likely to understand the problem and the results of the system than passive users. This is also supported by the spacing effect, which indicates that studying a problem over time leads to improved learning than studying the same content in one instance.

Other biases and effects (e.g., confirmation bias) that are specific to the user's analytic task during the progression (e.g., when steering the progression) should also be considered. Examples of such biases that could have a negative effect are listed in Table 2.

List of participants (alphabetical order)

Breakout Group "Task and Users": Michael Aupetit, Remco Chang, Luana Micallef, Dominik Moritz, Thomas Mühlbacher, Themis Palpanas, Adam Perer, Hendrik Strobelt, Emanuel Zgraggen

Breakout Group "Humans in the Progressive Loop": Marco Angelini, Remco Chang, Jörn Kohlhammer, Luana Micallef, Thomas Mühlbacher, Adam Perer, Giuseppe Santucci, Hans-Jörg Schulz

Both groups received input from the organizers Jean-Daniel Fekete, Danyel Fisher, and Michael Sedlmair.

References

1 Dimara, E; Franconeri, S; Plaisant, C; Bezerianos, A; Dragicevic, P. "*A Task-based Taxonomy of Cognitive Biases for Information Visualization*". IEEE Transactions on Visualization and Computer Graphics, to appear.
2 Ellis, G. (Ed.) "*Cognitive Biases in Visualizations*". Springer, 2018.
3 Evans, J S B. "*Bias in human reasoning: Causes and consequences*". Lawrence Erlbaum Associates, 1989.

4.3 ProgressiveDB: Architecture Report

Florin Rusu (University of California – Merced, US), Carsten Binnig (TU Darmstadt, DE), Jörn Kohlhammer (Fraunhofer IGD – Darmstadt, DE), Stefan Manegold (CWI – Amsterdam, NL), Themis Palpanas (Paris Descartes University, FR), Nicola Pezzotti (TU Delft, NL), Charles D. Stolper (Google, US), and Chris Weaver (University of Oklahoma – Norman, US)

4.3.1 Introduction

The goal of this session was to determine the specifications of a computing architecture for progressive computation in order to support interactive visual analytics. In a nutshell, design the architecture for "Progressive Tableau". Since the participants came from two distinct groups–visualization and databases–the discussion followed a pattern in which the visualization experts defined the requirements for progressive visualization and the database experts proposed a data computing architecture that implements the specification. Essentially, visualization is the user of a database engine capable to provide progressive results that enhance the display experience. Thus, the interaction between the visual dashboard, e.g.,

Tableau, and the computing engine is done through standard SQL queries operating over relational data.

The visualization requirements asked for progressiveness to be included in all the elements of the dashboard interface. Results to long-running queries should be returned progressively for continuous display. For these partial, i.e., approximate, results to make sense, they have to be statistically meaningful. Progressive results can be materialized in an evolving time series to provide continuity. The query itself should be allowed to evolve based on the progressive results. This evolution can take the form of minor changes to the original query or the triggering of (related) different queries. The refinement process generates a hierarchy of queries connected through their lineage. The end-user analyst can interact with the queries in two ways. It can monitor the execution by inspecting the progressive results. Or it can steer the query hierarchy by halting uninteresting queries and adding derived queries. In addition to the functional requirements, visualization imposes strict response time constraints that are highly-dependent on the graphical widget. For example, the movement of a slider elicits immediate response times lower than 100 ms, while the pressing of a button increases the feedback time to 1 second. If a custom text entry is required, the feedback time can be as large as 10 seconds. This "time engineering" process interacts with progressive execution by making some graphical widgets more likely to be included in progressive visualization and disqualifying others.

Given the functional specification and the performance constraints, the database experts proposed an architecture for progressive computation. Rather than designing a novel architecture from scratch, the approach taken was to embed progressiveness in a modern relational database architecture accessed through SQL. There are two solutions to realize this approach. First, a middleware that sits between the visual interface and the database engine can implement progressiveness by intercepting the queries and split them into smaller partial subqueries. This is a completely non-intrusive solution. The second approach requires changes to the database operators and tackles the situations that cannot be handled at the middleware layer. Independent of the approach, the progressive architecture has to provide support for progressive results and progressive queries. These are discussed in the following sections.

4.3.2 Progressive Results

The main obstacle to generating progressive results in a standard database are the blocking operators, e.g., join or group-by, which require processing all of their inputs before producing any output tuple. This has led to the development of non-blocking operators, such as hybrid-hash join, which generate result tuples much earlier and continuously. Their functioning principle is to operate on blocks (or chunks) of tuples that are processed by all the operators in the query execution tree at once. This is identical to performing the entire query on a subset of the data, i.e., the selected blocks or chunks. While a partial result is generated, the important question is "What is the significance of this result?" In the case of queries that produce tuples for which the output order does not matter, the partial tuples are as good as it gets. For queries that compute (grouping) aggregates, the partial tuples have to be statistically significant in order to produce an estimate of the result. This is realized by taking random samples from the base tables and performing the subquery on them. The process can be repeated multiple times, with a partial result generated each time. Are these results progressive? Only to the extent that the estimate is refined with each additional executed subquery. This is not the case if the subqueries are independent, only if each subsequent query "builds" on top of the previous ones. This requires the operator state to be

preserved across subqueries. Since this is not done by any of the existing database engines, architectural changes may be required.

The XDB system (https://github.com/initialDLab/XDB) that implements the Wander-Join algorithm in PostgreSQL 9.4 is a possible approach. XDB provides online aggregation-style estimates which improve progressively. The progressiveness integration is at all query processing levels, including optimization and execution. A progressive query is specified as follows:

SELECT ONLINE SUM ($l_{extendedprice}$ * (1 – $l_{discount}$)), COUNT(*)
FROM customer, orders, lineitem
WHERE $c_{mktsegment}$='BUILDING' AND $c_{custkey} = o_{custkey}$ AND $l_{orderkey} = o_{orderkey}$
WITHINTIME 20000 CONFIDENCE 95 REPORTINTERVAL 1000

This tells the database engine that it is an online aggregation query which reports the estimations and their associated confidence intervals, calculated with respect to 95% confidence level, every 1,000 milliseconds, and for up to 20,000 milliseconds. In this case, the partial results–consisting of the estimate and its confidence intervals–are managed by the database engine and they are pushed progressively to the visualization client. Another alternative is to allow the client to pull new results by itself. Rather than always transferring the complete result, it is also possible to send only the difference, i.e., delta, from the previous result. In order to execute the query efficiently, B+-tree indexes are built on all the attributes having selection predicates and all the join attributes. This allows for all the blocking operators to be transformed into non-blocking operators, thus, *GetNext*() from the query execution tree root can be pushed to a single table. At this table, *GetNext*() retrieves samples that satisfy the selection predicate efficiently since there is an index. Joins become index lookups because of the indexes built for all the join attributes. As more samples are extracted from the source table, the accuracy of the progressive results increases. This requires maintaining the state of the estimate instead of just the state of the aggregate.

Another strategy for progressive result generation is to extract samples from each of the source tables and execute the complete query over the samples. A middleware architecture is likely more appropriate in this case and the database has to provide quite minor support. State has to be maintained by the middleware between queries for improved results. This requires efficient sampling from the source tables. In PostgreSQL, this can be done by selecting all the tuples stored in the same block (information is available in the catalog). For sampling without replacement, a block has to be accessed only once and this can be controlled by the middleware. In order to provide accuracy guarantees, the middleware can implement several statistical techniques, e.g., CLT-bounds or bootstrap.

4.3.3 Progressive Queries

Producing and visualizing the results of a single query in a progressive manner is not the only aspect of progressive visual data analysis. Rather, the fact that results of a long running query are produced and presented in a progressive fashion during the entire course of the query execution enables the user to intervene and steer their analysis based on the evolving results at any point during query execution, once they realize that the analysis does not yield the desired insight in the data. The user performs their steering actions via the respective interaction mechanisms in the visualization interface, i.e., adjusting sliders, selecting regions, panning, zooming, brushing, adding/removing attributes, etc. These actions result in modifications of the currently running query to produce the required data. The modifications to the query include changing predicate boundaries, adding or removing filters, adding or removing result attributes, adding or removing group-bys (drill-down and roll-up),

and adding joins for pivoting. In order to properly expose query changes to the database engine, additions to the SQL language are likely necessary. For instance, queries have to be given names. While this can be done automatically by the system, it is also necessary to expose the names to the user in order to allow non-sequential jumps. Moreover, the query changes have to be recorded. A possible solution is the addition of an ALTER QUERY statement to SQL. For example, if we want to remove the predicate on the *customer* table in the query introduced in the previous section–named Q1–we write:

ALTER QUERY Q1 WHERE: DELETE $c_{mktsegment}$='BUILDING'

A naive approach to implement progressive queries is to stop the running query–preferably whenever a result chunk has been produced–and start the execution of the revised query. While semantically correct, this approach does not explicitly convey the information to the database engine that–let alone how–the original query and its modified version are related. Consequently, the queries are treated independently, even though they share the common results. A more effective approach is to refine the previous results and devise a method to combine the new results with the old ones. Moreover, the execution of the derived query should also be optimized based on the previous queries. For instance, the modification to query Q1 given in the example ALTER QUERY statement could be handled by performing Q1 with the predicate $c_{mktsegment}$ <> 'BUILDING' and then summing-up this estimate with the progressive result of Q1 computed up to the query change. This choice may be optimal if the number of 'BUILDING' tuples is large, thus, they can be pruned away before the costly join. Another option is to perform both Q1 and its modification–essentially, a group-by on $c_{mktsegment}$ with two groups–and then summing the results. Which of the three alternatives–the third being the execution of the modified query from start–depends on many aspects, such as the time when the transition happens, the quality of the Q1 estimate, the data distribution, etc. A progressive architecture should at least be aware of all these options and consider them in a meaningful way when a query is modified.

The progressive architecture we envision allows users to not only execute a single query in a progressive manner, but also to expand their data exploration by issuing additional related queries, which can also be executed in a progressive manner. In this case, the system is making sure that all the progressive queries are collectively optimized and executed. We illustrate this functionality with the following example. Consider a user that explores the census dataset and starts by issuing a progressive SQL query to get overall statistics on the gender distribution. Initial results show that the distribution is heavily skewed towards females, so the user decides to issue a second progressive SQL query to find out the distribution of females over different geographic areas. This distribution being uniform prompts the user to issue a third progressive query that computes the distribution of females for different age groups. At this point the system is running in parallel three progressive queries, where the last two are dependent on the first one. Notice, though, that they do not replace the original query – they run alongside with the original query. This difference is captured in SQL with the FORK QUERY statement which permits more extensive changes compared to ALTER QUERY. Essentially, FORK QUERY creates a graph of query dependency relationships that have to be all satisfied. The system is responsible for creating and running a query execution plan that includes all the queries in the graph, thus optimizing the overall system resource usage. This is a complicated problem because several decisions have to be made. First, the queries have to be prioritized. A simple approach is to give preference to the newer queries over the older ones. While this makes sense if we assume that the attention span of the user is skewed towards the immediate past, we may miss or delay significant trends for the older queries. The second decision is to extract the queries that can processed concurrently and

assign them the required resources. The end goal is for the user to make fast progress on the data exploration and analysis tasks, which is possible since they are able to quickly formulate and check different hypothesis, without having to wait for each query to complete execution.

The idea of progressive multi-queries can be pushed one step further by trying to speculate on how the next query may look like. For example, consider a progressive histogram over an attribute of a table in the database. An initial progressive histogram result may be coarse-grained, with a fixed number–say 2 or 10–of bins. Progressive results may consist of finer-grained bins, such as increasing from 2 to 4 or from 10 to 100 bins. We can speculate that such a finer-grained histogram will be requested and begin processing through a speculative forked query of the initial query. Since the overlap between the two queries is significant, most of the computation can be reused across them. If the database knew that the client would change the query in this or a similar way, the database could have served more accurate results faster by scheduling and beginning executing the forked query before the client even made the query. In the course of executing progressive queries, there is a clear expectation that subsequent queries will be made to the database. Moreover, these queries are likely to be similar to the queries before and after. This similarity may come in many forms, for example, a changed WHERE-clause parameter, group-by attribute, or addition or removal of a WHERE-clause. As such, the progressive architecture can "learn" the query sequence from past user sessions and predict the next queries in the sequence with high probability. The next step is to incorporate them into the query graph and schedule them for processing. This can be easily achieved within the progressive multi-query architecture by setting the priorities accordingly.

4.3.4 Conclusions

This report identifies the main components of a progressive computing architecture starting from the functional requirements of a visual analytics system and their performance constraints. This architecture is designed within the confines of modern databases, rather than proposing something from scratch. Nonetheless, all the elements are added as novel SQL constructs. In order to generate progressive results, non-blocking operators have to avoided. For these results to be statistically significant, the processing order matters. Progressive queries can take the form of delta modifications to the query statement, concurrent multi-queries spawned by progressive results, and speculative queries "learned" from prior investigations. They can all be modeled as a query relationship graph and optimized accordingly. We believe that the proposed architecture represents a first step for a fully progressive visual analytics system and opens many research directions in this area.

4.4 Evaluation and Benchmarks for Progressive Analytics

Sriram Karthik Badam (University of Maryland – College Park, US), Jaemin Jo (Seoul National University, KR), Stefan Manegold (CWI – Amsterdam, NL), and Hannes Mühleisen (CWI – Amsterdam, NL)

Progressive data analysis system design contains several degrees of freedom. System proposals require evidence to demonstrate their feasibility. The dynamic nature of progressive systems

poses new challenges for the creation of this evidence, since two previously disconnected classes of systems are involved: visualization and data processing.

The approaches to evaluate visualization and data processing systems are fundamentally different and incompatible. Evaluating visualization systems usually borrows methodologies from Human-Computer Interaction (HCI), while data processing utilizes standardized performance benchmarks. Investigating their performance individually is near-meaningless, and an integrated evaluation is necessary to demonstrate the value of a progressive data analysis system.

In the context of progressive visual analytics, success criteria for users can for example be (1) the cognitive ability of participants to interpret uncertainty, (2) behavioral and psychological factors (e.g., keeping the attention and limiting fatigue), (3) the visibility of changes during computation, (4) how informative early intermediate results are, and 5) the speed and accuracy of insights that human can derive from progressive visualization. For data processing systems benchmarking, visual analytics performance can include (1) metrics of the indication of uncertainty in intermediate results, (2) bounds on time until the first approximation is available, and (3) the stability of those intermediate results.

However, how informative early intermediate results are and the time until those are first available are very much at odds with each other: If more time is spent on computation, these results are likely to improve in quality, but will make the user wait longer, which might overstress their patience and thus their ability to derive knowledge. What is needed is a connection of user and task goals with specific system performance metrics. This can for example be compared with the 100ms time bound commonly cited as the threshold for interactivity which is an HCI-derived metric that has direct impact on system performance requirements.

We propose an integrated evaluation and benchmarking framework that reconciles the different methods and metrics of visualizations and data processing for visual analytics. Given a specific task, the framework adopts an iterative approach for reconciliation. Each iteration consists of two phases: (1) visualization-to-algorithm (V2A) reflection and (2) algorithm-to-visualization (A2V) reflection. In the V2A reflection, metrics and limits relevant for usefulness of a progressive visualization to humans are derived. From these metrics, data processing algorithms are selected and configured. Then, the A2V reflection follows where the specific properties of those algorithms in turn are expressed in terms of a set of different metrics and limits. In turn, they are fed back as clarifications into the progressive visualization experiments, again possibly influencing metrics and limits. In addition, local iteration is possible using the set of metrics and limits from the previous global iteration, allowing for example to optimize data processing for a specific response time or – after performance bounds are known – to experimentally determine the best result presentation.

4.4.1 Scenario

Greg wants to dispatch emergency services in a flooding disaster response coordination center by monitoring the locations and times of emergency calls. The amount of calls is overwhelming the capabilities of aid resources such as search teams. Hence, prioritization is required to send aid to where it can be most effective. A progressive visual analysis system supports Greg in this critical decision. Calls and other real-time information are aggregated and visualized in an interactive map. This map also shows detailed topographical information and population density data since lower and crowded areas are more likely to need attention by search teams. Due the large volume of static and incoming data the visualization and the underlying fluid dynamics models that predict water movement suffer from a large computational delay of over fifteen minutes.

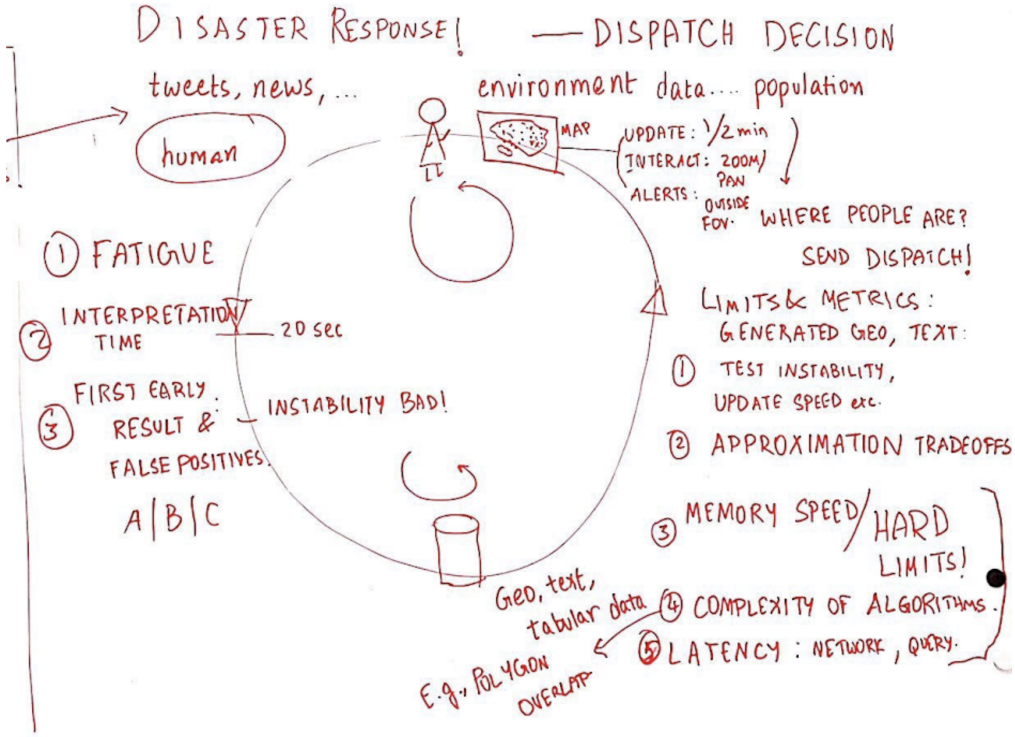


Figure 1 An example evaluation scenario.

The measures that can be used to evaluate the effectiveness of this visualization system can be (1) human fatigue resulted from monitoring and (2) time before decisions can be made. A first round of user study experimentation reveals that the computational delay of 15 minutes is not acceptable in this disaster response scenario, requiring an approximate approach in computation. At the same time, avoiding user fatigue and confusion dictates a large degree of visual stability in the displayed information.

Based on these requirements, a progressive algorithm is chosen that iteratively improves result while providing early intermediate results for early and continuously improving visualization (i.e., V2A reflection). This is made possible by limiting iterations through the fluid dynamics model and using an approximate polygon overlap algorithm to integrate topographic information. Unfortunately, this introduces a fair bit of uncertainty and volatility in the early responses from the system. It is found that sometimes false positives are created.

The possible false positives threaten the utility of the entire system. If operators cannot trust the displayed incidents, they may be reluctant to dispatch aid. This limitation from false-positive cases can be handled by visualizing the uncertainty and allowing a weighing of displayed incidents w.r.t. their certainty and their severity. For example, an incident involving one person with 100% certainty might be weighed against an incident involving 100 people with 80% certainty. As another V2A reflection cycle, the algorithm is extended to provide the uncertainty of the intermediate results.

The design of visualization is modified to show the uncertainty computed by the algorithm (i.e., the A2V reflection). This raises another aspect for evaluation such as how well the user can interpret the uncertainty and how correct the decision made during analysis is.

We have described a framework to reconcile the competing and sometimes incompatible goals between progressive visualization and the required data processing systems. We have shown that through iterating over their respective metrics and limits we can derive

both a compromise in their requirements as well as usable bounds for their evaluation and benchmarking. Regardless, open questions remain concerning the more general questions of how various metrics can be quantified and qualified (e.g., through experimental protocols concerning the question of trust). Similarly, the implications of "steering" a computation through interaction are not clear yet, while this could be simulated through restarts, already computed information might be re-used to speed up computation. Immediate next steps would be to investigate how visualization and data processing metrics are connected to each other concretely, what choices exist to achieve trade-offs between them, and how perform various different types of studies in the evaluation loop, e.g., formative or summative studies in HCI.

4.5 Uncertainty in Progressive Data Analysis

Sriram Karthik Badam (University of Maryland – College Park, US), Marco Angelini (Sapienza University of Rome, IT), Jean-Daniel Fekete (INRIA Saclay – Orsay, FR), Barbara Hammer (Universität Bielefeld, DE), Christopher M. Jermaine (Rice University – Houston, US), Hannes Mühleisen (CWI – Amsterdam, NL), Cagatay Turkay (City – University of London, GB), Frank van Ham (IBM Netherlands – Weert, NL), Anna Vilanova (TU Delft, NL), and Yunhai Wang (Shandong University – Jinan, CN)

4.5.1 Sources of uncertainty

In the progressive visual analytics pipeline, we can find different sources of uncertainty on the different steps of the pipeline. Initially, there is uncertainty in the data itself (e.g., measuring errors, sampled population). The algorithm can produce uncertain results independently of being progressive (e.g., local minima tSNE embedding). There is also a source of uncertainty introduced when an algorithm is made progressive. Furthermore, the visual encoding and visualization introduce uncertainty given, for example, the representation chosen or the resolution of the screen. There is also uncertainty associated to the perception and interpretation bias of the user on the visualization. The task to develop might also be uncertain. Finally, if uncertainty is estimated, there is also uncertainty in the uncertainty estimate, e.g., assumptions needed for the estimation.

All sources of uncertainty are relevant for the final decision-making process of the user. For this report, we focus on the uncertainty specific to progressive VA algorithms, and not present to other pipelines. We focus, therefore, on the uncertainty introduced due to making progressive the underlying algorithm. There is substantial amount of research studying each one of the sources of uncertainty named above. All uncertainties should be considered in combination with the progressive uncertainty, it is important for the final decision making process of the user. For example, it makes no sense to very accurately calculate a result for a data set that is very uncertain.

We can consider that there are two main ways to make an algorithm or method progressive:

- **Data based** uses the full data but rather a subsets or an aggregated version.
  - Sampling approximates the data fed to the method, for example by a sampling strategy.
  - Aggregation approximates by building aggregation data structures to facilitate the calculations, e.g., hierarchies.

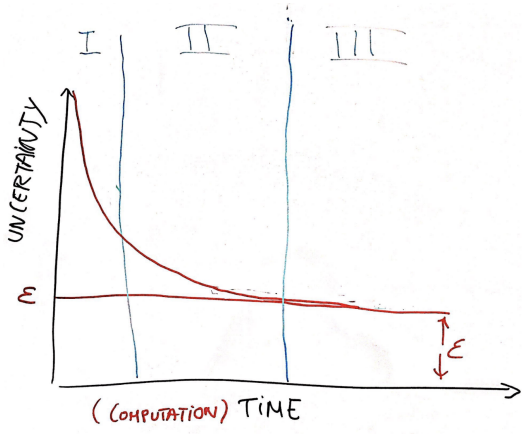


Figure 2 The three phases in progression.

- **Computationally based** approximates the computations themselves, for example, using heuristics.

4.5.2 Model of uncertainty in progression (Phases)

The stage of progression has a say over the uncertainty and how the user responds to it. To model uncertainty, we considered three phases during the progressive analysis process. Our current characterization is based upon certain assumptions about the data as well as the analytical process.

Assumptions: Our first assumption is that the progressive analysis system works with fixed data. This implies the world of the system is closed and there is no new data added during the analytical process. This assumption implies that in an ideal situation, the uncertainty decreases with the computational progress and converges to a minimum.

Beyond this, we focus on hypothesis generation for modeling the uncertainty. During hypothesis generation, the analyst is still open and undirected–exploring patterns and outliers that pop up from the data. In contrast, hypothesis confirmation creates more challenges as the analysts need to collate all the steps of the progression to actually confirm or reject their hypothesis with a certain confidence.

Based on these assumptions, we characterize three phases (Figure 2) in progressive analysis that showcase different effects of uncertainty on the user.

1. Phase I: The first phase represents an early part of the progressive analysis, where the estimates are improving rapidly. This phase is very uncertain as the system is just beginning to process the data either in batches or through iterations of an algorithm. This phase should help the analyst evaluate the methods to find major mistakes and fix parameters to adapt the analysis or restart it. Therefore this is the phase of **estimating suitability**. This is not the time to make concrete insights or even decisions from the data.
2. Phase II: During this phase, the analyst sees clear patterns in the progression capturing some artifacts within the data. The certainty or confidence increases and the analysts can develop early insights. Therefore this is the phase of **early response**. However at this stage, there is a tradeoff between time and value: the longer an analyst waits the more certain they can be. It is the responsibility of the system to balance the time and value to fit the constraints of the analyst.

3. Phase III: This is the phase of **convergence**. The uncertainty has stabilized to its minimum, in some cases it can approach zero at the end. The added value of watching the progression in this phase is little as the changes are minimal and the analyst already has enough understanding to make decisions.

This uncertainty evolution model represents the ideal world–what we hope to achieve. However, quantifying uncertainty from the system to represent such evolution is a complex problem. In fact, this problem needs to be resolved both from the data end–by creating measures that are driven by the user tasks–and from the visualization end–by creating visual representations that can capture the uncertainty of the intermediate estimates in these phases. In reality, there can be a synthesis of these two methods to convey a concrete picture of uncertainty to the user.

4.5.3 Measures of uncertainty

Given the considerations made on the model of uncertainty in progression, in this section we explain the different ways in which uncertainty can be computed. The definition of progressive uncertainty requires to consider the characteristics of the dataset on which the progression is applied. Starting from this characteristics a hierarchy of ways to compute uncertainty can be defined, reported in the following list of measures that are relevant to the decision making in progressive visual analytics:

1. Quality: For obtaining a measure of uncertainty it is needed that, for the original data some statistical properties are known, e.g., distribution of the data (provided by the DB in which they are stored), or assumptions on the statistical properties of the data (e.g., independent identically distributed, i.i.d., that allows to estimate a distribution for the data. In this case the quality at the various stages of the progression can be measured comparing the processed data with the distribution, obtaining confidence intervals.
2. Stability: if the distribution of the data is not known, it is not possible to measure quantitatively uncertainty, so a different indicator must be used based on the data computed up to the actual stage of progression. This indicator can be stability, that effectively estimate when a process is globally in a stable state; Stability measures are based on the computation and aggregation of difference between the values assigned to visual elements in two consecutive stages of the progression. If this measure is minimized under a certain threshold defined by the user/system for a certain sequence of stages of length N, the process can be considered stable.
3. Progress: Stability can be always computed independently from the task or dataset, but in the case in which the progressive computation never converges toward a stable state it can be used only as an evidence that the result is still uncertain. In this case the definition of uncertainty is again relaxed toward the concept of progress of the progressive computation. The computation of the progress is based on the assumption that the end of the progressive process can be computed or estimated.
   In order to allow this estimation we defined two methods:
   - When the problem is data-bounded (e.g., computation of average value) the progress is estimated with respect to the percentage of the total data that has been processed up to the actual stage of the computation.
   - When the problem is process-bounded (e.g., clustering) we make the assumption that the complexity bound of the particular algorithm used is used to estimate its termination time.

Given the availability of this information it is possible to estimate at every stage of the progressive computation how much it lasts to its natural end, obtaining the actual progress.

As an additional consideration it is important to note that each measure of this hierarchy can be evaluated in a Global fashion (on the whole progressive process) or in a local fashion based on the selection of areas of interest for the user. So while in order to use a level of the hierarchy as uncertainty measure all the relative assumptions must be valid for both global/local fashions, the decision-making process of the user could be based on different granularity. As an example, if it is possible to define the stability concept, it would be possible to have a progressive computation that at a generic stage m is globally not stable, while it is locally stable in the area of interest of the user.

4.5.4 Visual Encoding Characteristics

Uncertainty visualization is an active field of research within the visualization community. Visualization of uncertainty has as main challenge to identify effective visual encodings in a usually already cluttered display. Several visual encodings exist (e.g., textures, blurring, box plots) and their uncertainty perception effectiveness are being studied. An effective visual encoding is also data and task dependent. We will not deal with this general issues in uncertainty visualization here, we will focus on the specific characteristics that a visual encoding of progressive uncertainty should address. The main identified characteristics are as follows:

- The encoding should consider the notion of change which is an essential part of progressive methods. For example, enhance deviation in consecutive results, and provide visual indications of stability.
- In a general setting it is interesting to be able to identify differences of uncertainty in different areas of the data (i.e., locally), as well as having an overview of the overall uncertainty (i.e., global).
- The decision-making process needs to be able to combine progress, stability and quality. Therefore, the visual encoding should include this three aspects of the uncertainty.
- Bounded estimations will not always be possible, visual encodings that are flexible and allow to include soft-bounded representations for progression,quality and stability are relevant.
- The visual encoding based on animation/evolution should be taken with care to not mislead the user. It can also distract the user towards fast small changing areas which might blind from slow large changes.

4.5.5 Use Cases

In this section different scenarios are reported with respect to the characteristics of the uncertainty modelled in the previous sections.

4.5.5.1 (1) Information about distribution of data are available.

(progress – stability – quality) Dataset is composed of sales data of a company, and the task is to compute the Top 3 products that sold the most. Given the knowledge of the distribution, the progressive system computes data in chunks, and at each stage it produces a top 3 for which it is measured the relative quality/uncertainty with a confidence interval. At phase 1 the confidence interval will still be too large, so the user wait for the computation to unfold

more. The confidence interval will shrinks at every stage to the point in which at some stage in phase 2 it becomes acceptable for the user, Eventually the user will decide if it even needs it to stabilize or not (still in phase 2) for her final decision.

4.5.5.2 (2) Information about distribution of data are not available.

(progress – stability) The dataset and task are the same of Scenario 1, but this time the distribution of the data is not available. The progressive computation is still started on chunks of data and enter in phase 1. At this point no measure of quality/uncertainty can be computed, and the Stability and Progress are the only two measures available. The user, depending on their trends, choose to base her decision on the stability of the expected Top 3 (if it does not vary after several stages) and progress (proportion of data left to be computed). If the process does not become stable, user uses the Progress to make the decision.

4.5.5.3 (3) Stability is not definable.

(progress – quality) The dataset is based on a timely data processing coming from taxi data in New York City. The task is to identifying the taxi stops that earns more money during a month. In these conditions it is probable that even the identification of stable elements through the various stages of the progression could originate from natural skewness in the data (timestamps) and the stability measure would not result in a reliable indicator for decision-making. At that point the progress (quantity of processed data) is the only indicator that can be used to support the decision-making process.

4.5.5.4 (4) Identify groups in a dataset without knowledge of the groups or the data.

(progress, but contains visual indicators of stability) This scenario represents an exploratory analysis application in which the progression contributes to finding groups within a dataset. This is driven by a computation process that progressively outputs clusters from data (e.g., similarity of tweets) or classifies the data in presence of ground truth (e.g., identifying digits in MNIST). These groups are represented in a progressive visualization that highlights them. In terms of the three metrics described before, the progress can be concretely captured by following the iterations of the computations or the number of the batches processed within the data. This holds by assuming the developer bounded the number of iterations, as common for many ML toolkits. The stability and quality of the computation can be hard to quantify or even computationally expensive in some cases within this scenario. For example, computing the inter-cluster distance for k-means as the quality measure is computationally expensive. Therefore, stability and quality can only be visually assessed in this situation. It becomes the responsibility of the progressive visualization to indicate the changes and convey a sense of stability to the user through the global and local highlights.

4.5.5.5 (5) Pattern search in a high dimensional space

(progress only) Pattern search tasks are particularly challenging for progressive analysis. An example scenario for this task is identifying particular patterns of event sequences from a large event data. When dealing with pattern search, measuring progress is still possible when the event sequences are processed in batches. The progress can be conveyed to the user. However, the stability and quality of the search is hard to quantify. In simple terms, it can be hard to know how many more sequences may match the expected pattern without iterating through the rest of the data. Quality is also not possible to access as search represents a concrete outcome – hit or miss.

4.5.6 Open Questions

1. How to define the scheme in a general setting of progressiveness?
2. How best to convey all the uncertainty heuristics to the user to help decision making?
3. User and task dependency on the influence of uncertainty in the decision making process?

4.6 Threats to Research in Progressive Analytics and Visualization

Dominik Moritz (University of Washington – Seattle, US), Christopher M. Jermaine (Rice University – Houston, US), and Emanuel Zgraggen (MIT – Cambridge, US)

4.6.1 Introduction

Progressive analytics provides users with increasingly useful results with the results eventually converging to the precise answer. The claimed benefit is that users can catch errors early and make decisions based on the approximate results while not having to a priori decide on a necessary accuracy. Users can watch the results change until they are confident enough in the results. In this document, we describe threats to this ideologically pure vision of progressive analytics and visualization. These threats are meant as warnings against blindly working on this vision, and are designed to encourage critical reflection on what is needed to benefit users and not accidentally lead to confusion or false discoveries.

There are three main groups of threats that we identified. First, designing and implementing progressive systems has costs, and using them has additional overhead in terms of both computation and user effort. These costs may not outweigh the benefits, and there is a threat of producing over-engineered systems. Second, communicating uncertainty of approximations is challenging. Continuously-changing visualizations (aka. "dancing bars") can add an additional mental burden to users, and can lead to incorrect interpretations of uncertainty. Third, many variations on the idea of progressiveness have been well-studied by prior research, and methods exist that can address some of the concerns that lead to the idea of designing progressive systems in the first place. Hence, there is a danger that the development of progressive systems will be dismissed by academia or industry as unnecessary or not novel.

The next sections describe the three main threats to research on progressive analytics and visualization.

4.6.2 Cost vs. Benefit

We argue that a potential threat to the progressive analytics research agenda presented in this report is that cost to achieve it could outweigh the benefits.

Creating "pure" progressive systems, where users are continually presented with updated and refined results, are expensive to research, build, and run. The architecture of data management systems and query languages has to be fundamentally changed and machine learning, data mining and other algorithms need to be redesigned to be usable in a progressive setting. Furthermore, adding support for progressive results can increase the overall runtime, and therefore computation cost, of algorithms. For example, a blocking join implementation can use only a single hash table and thus use less memory and computer resources. Implementing

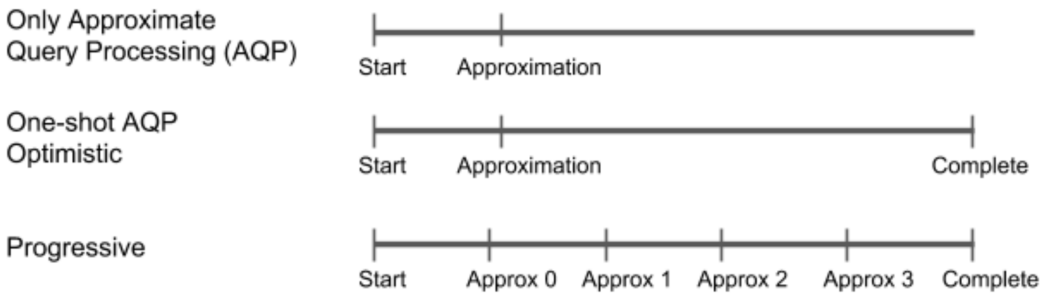


Figure 3 *Only Approximate Query Processing and One-Shot AQP Optimistic* are both approaches that can be implemented on top of existing databases, hence their development cost is low. A pure Progressive system requires more substantial changes, and therefore higher cost. We need to carefully analyze in which cases paying this cost is justified.

queries in a way so that they progressively converge to the true result adds overhead over approximation that does not have to make this guarantee.

Additionally, the progressive paradigm inflicts a cost on users: they need to learn how to use, interpret and interact with these systems. Progressive systems require users to understand and correctly interpret uncertainty, and progressively updating results are potentially distracting and increase the cognitive workload of users.

We urge caution before addressing these complex topics and creating engineering and research agendas around them. We believe that we must first gain a better understanding when progressiveness is really needed, critical, and clearly provides added value over perhaps simpler or existing solutions.

For example, one-shot sampling (i.e., approximation over a single sample) is already heavily used by data scientists to debug, test, and tune their machine learning pipelines before running them on the entire dataset. Such sampling techniques are much simpler to implement on top of existing systems without fundamental redesigns and therefore impose almost no additional cost. For a data scientist described in this example: does progressive analytics provide enough of an additional benefit to justify the redesign of widely used ML packages? As a research community it is important for us to find use cases and scenarios where progressiveness provably justifies its cost and compare and contrast it to other approaches.

4.6.3 Uncertainty

The vast majority of proposed methods for facilitating progressive computations utilize some sort of randomness or statistical inference, and hence they may result in uncertainty as to the final answer to the computation. Various statistical measures of uncertainty such as the standard error, bias, and variance, of the error distribution are computed to describe how the estimate relates to the correct answer. Since these are abstract measures of uncertainty that are often ill-understood by users, the standard approach to date as been to translate these metrics into confidence bounds on the answer to the computation, and present those bounds to the user as a measure of the uncertainty of the current guess as to the result of the computation. A confidence bound is a statement of the form, "With a probability of p%, the real answer is in the range x +- ϵ." Confidence bounds are appealing because they have a natural, graphical representation and can easily be animated, showing shrinking or converging bounds when visualizing a progressive computation.

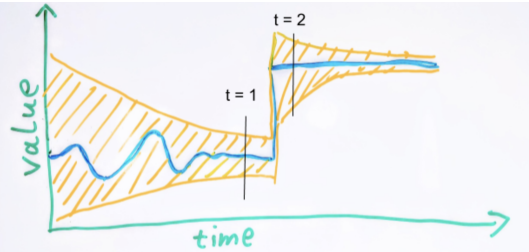


Figure 4 Estimation of a value (blue) over time with confidence bounds (yellow).

However, there are significant concerns regarding the use of confidence bounds in a progressive computation, relating to the fact that in an animation, a sequence of confidence bounds are presented to the user, and not just a single bound.

The first concern is that displaying an animation results in an instance of the so-called *multiple hypothesis testing* (MHT) problem. When ignored, MHT results in so-called "p-hacking." Imagine that a user wants to answer the question: "Were the profits in Europe in 2017 less than $2.8 million?" She watches a visualization with 95% confidence bounds that progressively shrink over time, until the entire confidence region is above $2.8 million, and she stops the computation. The problem is that the user may have been shown 100 different 95% confidence bounds in sequence, only the last of which excluded the $2.8 million cutoff, and concluded that there was only a 5% chance that the profits were under $2.8 million. In fact, rejecting the hypothesis with a p-value of 0.05 is not implied by the observed bounds, as the user has tested the "less than $2.8 million" hypotheses 100 separate times, and found it to be rejected only one time. Depending upon the correlation between the various estimates, the "real" p-value may be much, much higher than 0.05.

A second problem regarding showing a series of estimates and bounds to a user is that the user may view the sequence of estimates and bounds as data to be interpreted, when there is no statistical basis for doing so. Consider the picture in Figure 4, showing an estimate and 95% confidence bounds as a function of time. A user may watch a visualization of the bounds and conclude at time t = 1 that they have converged. After all, at that time, the bounds have focused in on a particular value for some time, and now appear stable. However, there is no statistical basis for this conclusion. No matter how stable the bounds appear, the bounds still allow for a 5% chance that there will be a jump as shown at time t = 2. In effect, the animation itself has made it easy for a user to incorrectly draw conclusions about the accuracy of the computation.

A final problem with uncertainty in the context of progressiveness is that there is "uncertainty in the uncertainty." That is, statistical measures of uncertainty such as confidence bounds almost always depend upon estimating the variance of an error distribution, or else using a simulation-based method such as the bootstrap (typically, estimating the variance of an error distribution is as difficult as estimating the answer to the computation, and so computing it exactly is not feasible). Thus, the bounds themselves are actually an estimate, and may be incorrect. In practice, displayed bounds can shrink for some time, only to get wider (showing more uncertainty) over time as more data are processed. This may happen even though the variance or standard error of the underlying computation can be formally shown to monotonically decrease over time (this is commonly observed when using the standard, ripple-join style estimator for aggregate queries over relational joins). The concern is that a user may make a decision based upon confidence bounds that are tight only by virtue of themselves being a high-variance estimate for the "real" bounds.

4.6.4 Novelty

The three main use cases for progressive analytics and visualizations are massive datasets where full computation takes too long, monitoring and incremental computation of algorithms, and illustrating algorithms step by step. Some potential consumers of progressive technology may argue that these usage scenarios can all be addressed with existing ideas and research efforts (such as approximate query processing and anytime algorithms) and do not require a new research topic called "progressiveness."

For example, in exploratory analysis, users create visualizations, interpret the results, and then decide to refine their query or ask a new question. Speed is crucial to maintain attention and to ensure that the analyst will not lose their train of thought. However, while speed is crucial, perfect accuracy is often not necessary. For example, an analyst may have queried for the sum of values and only upon seeing the result notices that they actually wanted an average. With quick results, they can easily correct this mistake. However, instead of progressiveness, users can just use an approximation based on a sample of the dataset. Approximate query processing (AQP) is an established research area in databases. Data scientists already typically develop their analysis pipelines on a sample before running it on the full dataset, and may question the need for a fully progressive system.

For another example, monitoring machine learning algorithms allows model creators to see when their model converges, and to determine whether there are potential problems with the data or the model parameters. The machine learning community has long investigated anytime algorithms where users can "pull" results at any time, and monitoring is built into most machine learning frameworks. Incremental computation to update a model when new data arrives has also been an area of extensive research in the algorithms and machine learning communities. There is concern that these problems have already been studied, and gathering these ideas under a new umbrella called "progressive computation" does not add additional value.

Finally, the developing visualizations of computations that evolve over time, with the goal of giving people insight into the inner workings of a computation or to potentially steer the computation, is not new. In fact, it is a well-known idea, generally referred to as "algorithm visualization." Again, this may lead to questions regarding the necessity of branding a class of related ideas as "progressive computations."

4.6.5 Conclusion

In this document, we describe three threats to the area of progressive analytics and visualization. We do not see these threats as existential threats but as questions that will come up in grant reviews and challenges that should be considered. We hope that considering these threats encourages the community to investigate the design space between static and progressive. We should investigate how exposure to incorrect approximate results influences decision making even when users see the precise results later (e.g., anchoring effect). Working on these challenges may help us to overcome the threats outlines in this document.

Since we are lacking practical and usable progressive systems today, we should focus carefully on what is already possible with existing systems, and what cannot be done with them. Along the way, we may find "killer use cases" that convincingly show that alternatives like just approximation are not sufficient and proof the need for progressiveness. Before we strive for the "pure" progressiveness of continuously updating results, we should investigate better tool support for users today. Analysts already use samples and there is a need for better tools to communicate uncertainty, help users make decisions in the face of uncertainty, select appropriate samples for selective queries, and better support for incremental update to models when only one parameter changes.

4.7 Progressive Data Science: Potential and Challenges

Cagatay Turkay (City – University of London, GB), Carsten Binnig (TU Darmstadt, DE), Jean-Daniel Fekete (INRIA Saclay – Orsay, FR), Barbara Hammer (Universität Bielefeld, DE), Daniel A. Keim (Universität Konstanz, DE), Themis Palpanas (Paris Descartes University, FR), Nicola Pezzotti (TU Delft, NL), Florin Rusu (University of California – Merced, US), Hendrik Strobelt (IBM TJ Watson Research Center – Cambridge, US), and Yunhai Wang (Shandong University – Jinan, CN)

Data science often requires time-consuming iterative manual activities. In particular, the early activities including data selection, preprocessing, transformation and mining, highly depend on iterative trial-and-error processes that could be sped up significantly by quick feedback on the impact of changes. The idea of progressive data science is to compute the results of changes in a progressive manner, returning a first approximation of the results quickly with iterative refinements until converging to the final result. Enabling the user to interact with the intermediate results allows an early detection of wrong or suboptimal choices, the guided definition of modifications to the pipeline and their quick assessment. In this paper, we discuss the progressiveness challenges arising in the different steps of the data science pipeline. We describe how changes in each step of the pipeline affect the following steps and outline why a progressive data science process will help to make the process more effective. Computing progressive approximations of the results resulting from changes creates numerous research challenges, especially if the changes are made in the early steps of the pipeline. We discuss these challenges and then outline first steps towards more progressiveness in the data science process, which will ultimately help to significantly speed-up the data science process. An extended version of this report is available as [1].

References

1 Cagatay Turkay, Nicola Pezzotti, Carsten Binnig, Hendrik Strobelt, Barbara Hammer, Daniel A. Keim, Jean-Daniel Fekete, Themis Palpanas, Yunhai Wang, and Florin Rusu. Progressive Data Science: Potential and Challenges. *arXiv.org e-print*, 812.08032, 2019.

Participants

- Marco Angelini
Sapienza University of Rome, IT
- Michael Aupetit
QCRI – Doha, QA
- Sriram Karthik Badam
University of Maryland – College Park, US
- Carsten Binnig
TU Darmstadt, DE
- Remco Chang
Tufts University – Medford, US
- Jean-Daniel Fekete
INRIA Saclay – Orsay, FR
- Danyel Fisher
Honeycomb – San Francisco, US
- Hans Hagen
TU Kaiserslautern, DE
- Barbara Hammer
Universität Bielefeld, DE
- Christopher M. Jermaine
Rice University – Houston, US
- Jaemin Jo
Seoul National University, KR
- Daniel A. Keim
Universität Konstanz, DE
- Jörn Kohlhammer
Fraunhofer IGD – Darmstadt, DE
- Stefan Manegold
CWI – Amsterdam, NL
- Luana Micallef
University of Copenhagen, DK
- Dominik Moritz
University of Washington – Seattle, US
- Thomas Mühlbacher
VRVis – Wien, AT
- Hannes Mühleisen
CWI – Amsterdam, NL
- Themis Palpanas
Paris Descartes University, FR
- Adam Perer
Carnegie Mellon University – Pittsburgh, US
- Nicola Pezzotti
TU Delft, NL
- Gaëlle Richer
University of Bordeaux, FR
- Florin Rusu
University of California – Merced, US
- Giuseppe Santucci
Sapienza University of Rome, IT
- Hans-Jörg Schulz
Aarhus University, DK
- Michael Sedlmair
Universität Stuttgart, DE
- Charles D. Stolper
Google, US
- Hendrik Strobelt
IBM TJ Watson Research Center – Cambridge, US
- Cagatay Turkay
City – University of London, GB
- Frank van Ham
IBM Netherlands – Weert, NL
- Anna Vilanova
TU Delft, NL
- Yunhai Wang
Shandong University – Jinan, CN
- Chris Weaver
University of Oklahoma – Norman, US
- Emanuel Zgraggen
MIT – Cambridge, US