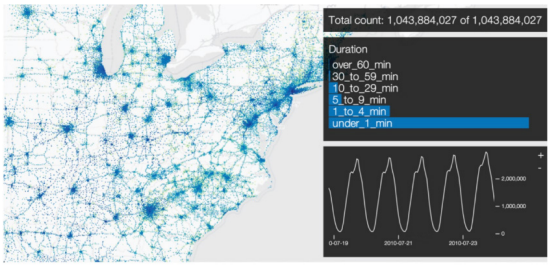
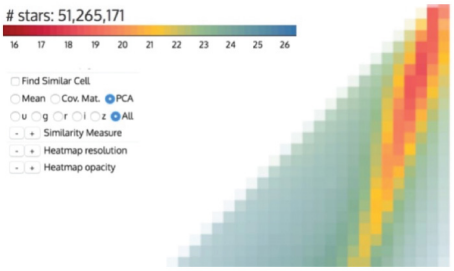
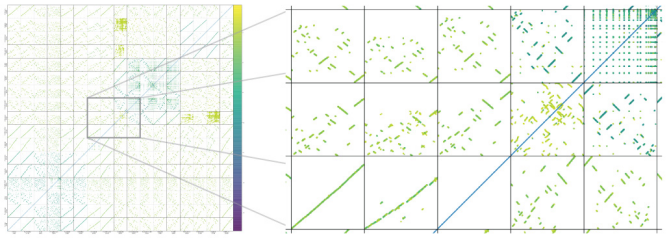


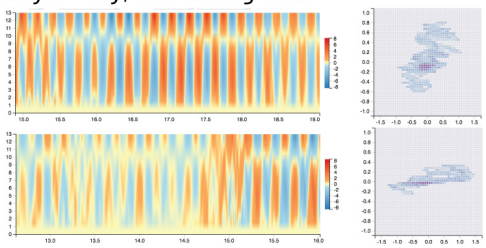
Call Detail Record exploration, from Lins et al.



Star characteristics in the Sloan Digital Sky Survey, from Wang et al.



Syntenic comparison across 17 species of the *Plasmodium* genus, ~10^6 total matches



Exploratory visualization of earthquake simulations, ~10^9 data points

Figure 1: Recent work by the PI has enabled interactive visualizations with billions of elements, from telecommunications records to star characteristics from the Sloan Digital Sky Survey [34, 53], with very quick response times (at around 10ms). Nevertheless, building the data structures takes too long, they do not integrate well with more general data analysis tasks, and they take too much storage; this makes them unsuited for many visualization and analysis tasks, such as ones found in evolutionary biology and civil engineering. We propose novel methods that will make interactive, scalable visualization a central part of the data analysis pipeline.

1 Introduction

We now have the capacity — and the culture — to digitally store vast amounts of information. In the past, data collection was done judiciously and sparely, but the current trend is to gather as much data as possible, and hope to analyze it in the future. As a result, data *exploration* has become a popular path to a quantitative understanding of the world. Data visualization is central to this strategy, with its unmatched power to uncover unforeseen patterns in the data, elicit novel analysis questions, and increase human understanding. Cleveland and McGill's early experiments brought Stevens's psychophysics research program to visual scales [19, 48] and provided a foundation for designing visualizations based on appropriate data semantics [37], creating systems where interaction primitives are sensibly tied to database schemata and queries [54, 55], and building appropriate visual aggregation mechanisms to truthfully and effectively depict large data sets [21].

In contrast, **our computational understanding of exploratory visualization lags behind**. Even though the importance of low-latency in interaction is well-known [35] and some work leveraged the incredible speed of modern graphics processors [38], there has not been nearly as much work aimed at improving our foundational computational understanding of interactive analysis and visualization. Even though it has become a central part of how we try to make sense the enormous data sources we acquire, visualization and interaction currently is implemented largely as an afterthought, past the time where the decisions about algorithms to index and process the data have been made. Consider the following data analysis scenarios, from collaborations carried out by the PI in the last five years:

- An evolutionary biologist studies syntenic relationships (that is, the coincidental localization of gene sequences in different chromosomes) across 17 different *plasmodium* species. The genomic analysis yields about a million syntenic matches, each with a position on a pair of chromosomes, and derived

information such as estimated mutation rates. This data needs to be aggregated and visualized along a variety of axes: positions on chromosomes, species, mutation rates, etc. Their current workflow consists of web-based infrastructure and custom visualization applications.

- An astronomer builds a computational platform for detection of rare, transient events such as supernovae events. They need to compare (for example) the distribution of observed brightness over different bandpasses, detected over hundreds of millions of stars in catalogs, and use this information to build a variety of models and compare them, visually and otherwise. Their workflow consists of a number of Python libraries for data analysis and processing.

- A civil engineer studies the impact of earthquakes in building structures by simulating its effect in a supercomputer. The analysis yields tens of thousands of multivariate, multidimensional time series, which can be aggregated, studied and visualized across physical measurements (shear, moment, acceleration, story height). The simulations are run in a cluster and visualized in Matlab.

- A data scientist is studying fraudulent activity in a large telecommunications network by looking for anomalous spatio-temporal patterns in call volumes. There are about 1 billion calls per day over the continental United States, and each call has a variety of attributes: how long it took, which device was used to call it, the originating location and phone number, etc. The analysis is carried out in a combination of R analysis scripts and custom visualization applications.

All of these scenarios are ripe for disruption from the perspective of interactive, visual data analysis, and **all of them are limited by performance reasons**. The science bottleneck has classically been one of understanding the data, rather than just acquiring it, and we have now reached a point where we cannot make sense of the data because our data analysis infrastructure was not built with interactive performance in mind. There has been some recent work to address this situation. imMens was the first proposal to build a data structure carefully designed for fast visual exploration [36]; essentially in parallel, we developed nanocubes [34], whose indices optimize for the hierarchical nature of data exploration and sparse nature of data in higher dimensions. These two systems offer the capability to build visualizations for a variety of spatiotemporal queries ("show heatmap all geolocated tweets in 2015", or "show histogram of activity throughout the week in California") for large datasets (on the order of a billion elements) in well under a tenth of a second: see Figure 1. These are exciting developments, but as we argue in this proposal, they fall short in how they integrate with the rest of data analysis practice, in how much memory they consume, and how much pre-processing time they require. Similarly, there has been much research and development in data management for big data, from Google's now-folklore MapReduce system [22] to its open-source Hadoop and Hive counterpart [1, 51]. There, the lack of emphasis of *interactive, low-latency, visual* systems is the main shortcoming, as attested by the explosion in popularity of interactive, visual data analysis ecosystems around R [30] and Python [39].

Understanding and exploring data should be as easy as it is to store it. We propose to permeate the data analysis pipeline with computational primitives that are specifically designed to support low-latency exploration and visual analysis. This means rethinking the techniques we use to acquire, process, query, and display data, and taking both data management and visualization research into account. In short, we ask the following: **What does interactive data analysis look like if modern data structures for visualization are an integral part of the pipeline?** A thorough treatment of this question will necessarily go through the matter of adoption: as data analysis becomes ubiquitous, it becomes more important to ensure these advances are well integrated in the data-intensive workflows of researchers.

We focus on a particular class of techniques for acceleration of interactive analysis, namely those based on *data cubes* [26]. It is natural to ask: why take such a restricted point of view? Recent evidence presented by the PI and collaborators (and other teams as well) suggests that data cube variants can offer superior

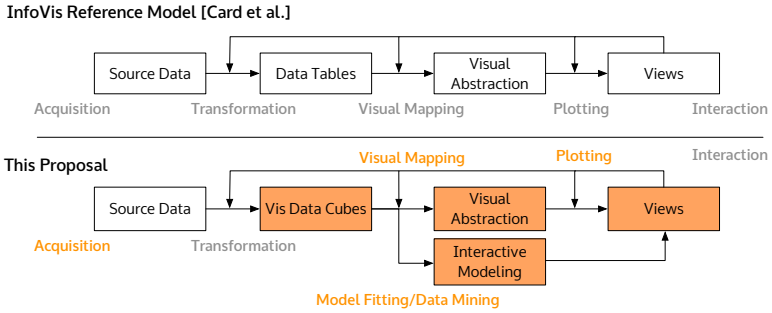


Figure 2: Top: the InfoVis Reference model [14], in a style inspired by Cottam et al. [21]. Note that data tables (and hence linear-time scans) feature centrally, severely limiting the scalability of the model. Bottom: this proposal for an end-to-end scalable visual analysis pipeline, centered around visualization-specific data cubes. The highlighted parts represent aspects of the pipeline touched by this proposal.

performance characteristics for interactive visual analysis in terms of query latency. At the same time, essentially all current proposals come up short in one of more aspects (Section 2). Our preliminary results show promise that these shortcomings are not essential, and that properly designed data cubes can work as general infrastructure for interactive data analysis. In other words, this proposal identifies a single concept — a hierarchical data cube in the style of nanocubes — that we argue should be ubiquitous, and yet it is not. We intend to bridge that gap, through research aims described in Section 3 and in bringing the concepts to a wider audience of data analysts and visualization practitioners; our plan for connecting to existing practice is described in Section 4. At the same time, we recognize the inherent risk in a relatively focused proposal, and will spend some time discussing alternative plans should we discover evidence that hierarchical data cubes are fundamentally flawed. The following list summarizes our goals.

- Section 3.1: **Synthesize the currently-disparate understanding of visualization-specific data cubes**. The many recent proposals for data cube algorithms suited for visualization currently resist direct comparison to each other. We propose to address this by designing a curated benchmark and using the benchmark as the foundation for a unifying review of the current literature.

- Section 3.2: **Investigate space-time tradeoffs**. There are many unexplored regions of the design space of visualization-specific data cubes with significant potential gains in required storage, while preserving low query times. The benchmark we design will serve as a crucial evaluation mechanism in this step.

- Section 3.3: **Design novel interactive analysis techniques based on data cubes**. With a deeper understanding of these novel data cube structures, we propose novel *analysis* techniques which use fast data cube queries (instead of repeated array scans) as their only interface to the data. We propose to design cube-aware versions of classic techniques such as clustering and classification, starting to move the research program to integrating data cubes into the pipeline of interactive analysis.

- Section 3.4: **Integrate with existing analysis infrastructure**. Much of the data analysis done in practice today happens using R and Python ad hoc code, and specifically R's data frames and Python's pandas tables. In order to replace tables that are repeatedly scanned with our proposed techniques, fundamental research into how the integration can happen is required.

Broader Impacts. We recognize that reimagining the data analysis pipeline requires meaningfully engaging with the modern practice of data analysis and data science. If this new perspective stays confined to a small number of researchers and especially trained graduate students, there is a risk for a large wasted opportunity. The last research aim in this proposal, described in Section 3.4, is designed precisely so to

increase the impact of the work in practioners' lives. But, more concretely, we propose to explicitly create integration libraries in R as a proof of concept of the feasibility of the work.

The PI has an extensive track-record of producing usable open-source code from both academia and industry, and dissemination of research results via MIT-licensed, open-source software will be a significant part of this proposal's broader impacts. Finally, in the two years since starting as assistant professor, the PI has a proven track record of recruiting underserved segments of the population to STEM fields. This proposal will provide funds for the PI to continue this trend.

2 Background

Exploratory visualization is an extraordinary way to understand data, and hence the world. As Tukey famously put it, "The greatest value of a picture is when it forces us to notice what we never expected to see" [52]. Consider the concrete example of Anscombe's quartet [6] (Figure 3): even in trivial situations with only 2 dimensions and 11 points, we can easily find datasets that have the same means (in both dimensions), medians, covariances, linear fit parameters and unexplained variances. At the same time, the datasets are clearly quite different from each other, and that is apparent from a simple visualization. These graphical views are among our best tools even for such simple datasets; when we have billions of data points to explore, the ability to get quick, visual answers for a large number of queries is an absolute necessity.

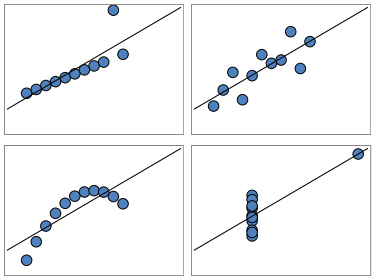


Figure 3: Anscombe's Quartet: these four datasets have many identical summary statistics; visualization elucidates differences that would be otherwise obscured.

2.1 Previous work

Data Management. We are interested in building infrastructure for visual, interactive analysis of data. Researchers in data management have explored somewhat similar avenues in the past, and we review some of the work here. The classic, seminal work in enabling fast, ad-hoc aggregation queries is undoubtedly Gray et al.'s data cube paper [26]. Indeed, this proposal directly builds on this field, and we review the area in more detail in the following paragraph. Hellerstein et al. built the original vision for a responsive RDBMS experience in Online Aggregation [28], from which much of the research on streaming query processing has flowed. For example, in the HCI side, Fisher et al. have explored the extent to which analysts can understand the output of such systems, and also how to design custom views for them [24, 25]. Back in data management, researchers have studied how to increase the convergence rate of such statistical queries, such as shown in BlinkDB [5]. Data management researchers have also recently recognized the need to incorporate analysis tasks into their worldview, as evidenced by the work of Feng, Hellerstein and co-authors in implementing data mining primitives within a relational data base [23, 29]. The focus of this proposal is to enable fast, visual, interactive experiences throughout the entire data analysis process, a goal we argue is pressing, but has not yet been adequately addressed.

Data Cubes. Data cubes [26] formalize the intuition that to understand the behavior of a dataset, there are many different aggregations in which analysts will be interested. Consider the basic example in Figure 4, assuming the table on the left represents a log of sold cars in a dealership. We might be interested in how many Hondas were sold, how many SUVs were sold, or maybe how many total cars were sold. A datacube (on the right subfigure) stores all possible aggregations, representing a summarization over any given column with a special value $\star$. In this example we are using the simplest summary (a "count", representing the total number of rows that match the criteria). But we could consider using, for example, maximum price as the aggregation operation, or the *sum* of the prices sold, which when combined with a count, yields the

average price. More generally, these are different *commutative monoids*. Monoids are simply operations $\oplus$ that are commutative ($a \oplus b = b \oplus a$), associative ($a \oplus (b \oplus c) = (a \oplus b) \oplus c$), and have a neutral element $\emptyset$ ($a \oplus \emptyset = a$). Any commutative monoid will yield a different aggregation, with potentially important analysis applications [16–18, 40, 41, 53]. One clear concern is the explosion in size of the data cube structure: naive representations will grow exponentially fast and quickly become impractical. There has been much work in making data cubes sufficiently small for realistic database sizes. The foundational work in this subarea is Harinarayan et al.'s dynamic-programming algorithm for using the column subset lattice to guide *partial computations* of the data cube [27]. Their main insight, that it's always possible to answer aggregation queries by scanning *some* of the columns, motivates an algorithm to compute the expected running time of data cube queries when some columns are not computed. In terms of minimizing the overall memory size of a data cube structure, the current state of the art is, to the best of our knowledge, the work of Sismanis et al. [47], which exploits a large degree of shared structure in the data cube.

Consider, on the other hand, what would happen if a column in that table contained *geographical position of sales*, or price. These two variables have "structure" in their definition: specifically, there exist very natural aggregation queries that traditional value-at-a-time data cubes do not precompute. Imagine the query "all sales of cars with prices between U$10,000 and U$30,000 within 3,000 miles of Los Angeles". The precomputation structure of a standard data cube does not help: the table might contain the sum of all cars that sold for U$17,000 (say Honda Civics), or all of the cars sold in any given dealership, but in order to combine these to get the overall result, we still need a linear scan over the data cube itself. Worse, in data analysis and exploratory visualization, many (if not most) variables have such "multiresolution" structure: sometimes we're interested in aggregations over the entire Western hemisphere, and sometimes we're interested in queries over local area. Nanocubes, as we will show below, are a variant of data cubes in which the precomputation structures *do* help answer these multiresolution queries quickly. In addition, nanocubes exploit sharing structure much like Dwarf cubes do [47].

Table ...

Make	Style	Trans.
Honda	sedan	auto
BMW	hatch	auto
Ford	SUV	manual
Ford	hatch	manual
BMW	sedan	auto

... Data Cube

Make	Style	Trans.	Count
*	*	*	5
*	*	auto	3
*	*	manual	2
*	sedan	*	2
*	hatch	*	2
*	SUV	*	1
*	sedan	auto	2
*	hatch	auto	1
*	hatch	manual	1
*	SUV	manual	1

Figure 4: On the left, an example table for which we build a (partial) data cube table, on the right. The data cube table holds the "count" preaggregation, making aggregation queries potentially very fast .

Visual Analysis of Big Data. As we previously mentioned, Liu et al.'s imMens was one of the first visualization-specific data cube proposals, taking central advantage of the speed of the GPU to execute some aggregations from data tiles, hitting a favorable point in the space of memory-runtime tradeoffs [36]. Cottam et al. have subsequently defined a general framework to turn previously-computed aggregations into appropriate visual interfaces [21]. Notably, however, they do not engage in a discussion of how to enable *sufficiently-fast* aggregations, which are necessary in order to enable effective, interactive data exploration [35]. In this proposal, we build on some of the work from the PI that shows how changing the data that is aggregated in the data cubes enables a large class of analysis techniques. Although this observation has been made before across the data mining and data management literature [16–18, 41], the research program has not taken into account the particular requirements of visualization systems, nor has it explored the design of general infrastructure for the entire visual analysis pipeline. Finally, we note a recent, exciting development from the data management community: the embracing of perceptual limitations in approximate query processing. As Wu et al. argue [56], instead of answering slowly – but exactly – a database query, we can answer quickly a query, returning a value that is imperceptibly different from the exact one. The main technical innovation

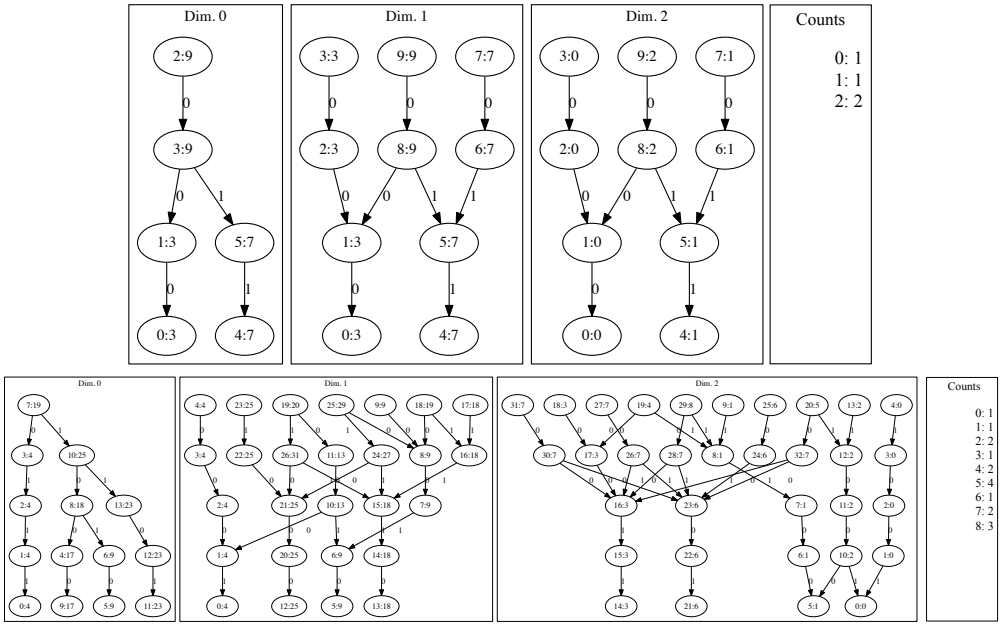


Figure 5: Two simple 3-dimensional nanocubes. The first dimension is always a single tree, and all queries start at the root of that tree. The other dimensions form *shared forests*: each root node has the structure of a binary tree, but some nodes are shared. This sharing provides significant space savings over a naive precomputation scheme and is the main reason nanocubes are reasonably efficient, memory-wise. Note how the amount of sharing (nodes which belong to more than one path) increases as more points are added (bottom example).

here is the incorporation of Cleveland and McGill's perceptual measurements into the physical query planner of data management systems. We see the work in Aim 3.3 as a concrete implementations of this research program, targeting analysis tasks and using the nanocube infrastructure as the computational substrate. In a sense, this work recognizes that data analysis is ultimately driven by humans, and takes human strengths (and limitations) into account. Other ways of incorporating human includes modeling user actions to predict future behavior and hide query latencies [10, 15].

Expressivity and Programmability of Interactive Visualizations. Although interactive visualizations are powerful means to interact with data, actually writing the programs that produce the desired behavior remain a severely limiting factor. Since the beginning of the field there has been considerable effort in encoding the desired relationships and semantics in ways that can be manipulated by programs. MacKinlay's seminal APT work [37] led to Stolte's characterization of data cubes [49] (which eventually led to Tableau, now a multi-billion dollar company) and Bostock et al.'s breakthrough abstractions in d3 [12]. Interactive primitives have received similar attention as well: consider Weaver's characterizations of visual querying metaphors [54, 55], and Satyanarayan et al.'s higher-level Reactive Vega bindings [45]. Achieving aim 3.3 will mean that the field will have to take preaggregated structures into account when designing such DSLs. In other words, here we focus on developing the necessary groundwork, but see connections as natural future opportunities.

2.2 Prior work by the PI

This research plan builds heavily on the award-winning nanocubes work and followup work in which the PI has been a co-author [34, 44, 53]. In order to provide context for the research aims, we provide below a brief overview of the nanocubes data structure.

Imagine a directed graph in which each node represents an aggregation over some subset of the data, and each node has three labeled edges. Two of these edges represent a *partition on the node's data set along some*

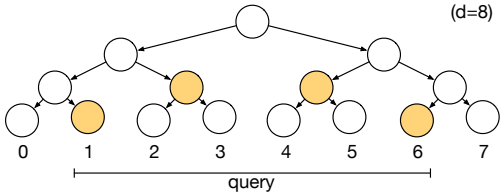


Figure 6: A single-dimensional range query aggregation uses inner nodes to give answers in $O(\log d)$ time for any contiguous range (in this example, the range is 1–6 inclusive). The orange nodes form a *minimal cover* of the region. To build multi-dimensional aggregations, we take the *next* pointer of each of the nodes in the cover, call the algorithm recursively in the following dimension (and *its* following dimension, and so on), and finally combine all of the reported summaries. This gives a $O((\log d)^k)$ running time.

dimension (we call these the "refinement" edges), and the third edge represents a pointer to a node which holds the *same data subset, but whose partition operates in a different dimension* (we call this the "next" edge). This structure can easily represent a data cube, but a naive implementation that progressively splits the dataset at every chance (producing a tree) will contain an exponential number of nodes — with a large base and exponent at that. A nanocube, on the other hand, exploits *shared structure*: it's often possible to prove that data subsets are the same across different nodes. In that case, nodes can be shared by many edges, providing a potentially large savings.

Consider the following simple example of a 3-dimensional nanocube in which values in each of the dimensions range from `000` to `111` (as we will see, thinking of the values are represented in binary will help). Each node in this example is an augmented binary tree node, so along each dimension we get binary trees, and each node has an additional reference to a root node in the next dimension (to represent the idea of "no further refinements along this dimension"). In Figure 5 (top), we show the resulting nanocube after adding the points $(\texttt{000},\texttt{000},\texttt{000})$ and $(\texttt{011},\texttt{011},\texttt{011})$. Each node in each dimension has three pointers: two which subdivide the data within the dimension (denoted by edges in this diagram), and one which points to a node in the next dimension. In these diagrams, we denote these "next" pointers by the node *id* they point to, and so the label for each node has the notation "*id*:*next*". The bottom subfigure shows a slightly more populated structure.

The fundamental query that nanocubes can answer at speed is the *range query aggregation*: the combined aggregation of all values inside a given multidimensional range (see Figure 6). By answering single-dimensional range queries in time and space $O(\log d)$, where d is the cardinality of any dimension, nanocubes can answer the multidimensional query in time and space $O((\log d)^k)$ by accumulating many single-dimensional queries. The principle behind fast single-dimensional range query aggregations is shown in Figure 6. In a sense, **the crucial question connecting the work we propose is: how much of the data analysis pipeline can we encode via range query aggregations, or similar traversals? Which techniques become fast, and what must we give up?**

3 Research Plan

Our research plan starts by organizing the knowledge in the field, and then using this organization to guide future research. This strategy provides the rest of the research community with a valuable resource relatively early, the lack of which we argue is holding the community back. And, although we have confidence in the proposed aims and our preliminary research results, seeking to synthesize the knowledge of the field early on will also allow us to quickly spot unforeseen problems in the research plan and adjust accordingly. This research will also produce a substantial amount of open-source software in the course. Although we

refer the reader to Section 4 and to the Data Management Plan for more details, we note in summary that we will host the source code (and bug trackers, distributions, etc) for the software aspects in this project within our research group's organization in GitHub [50]. We also defer a discussion of this project's timeline to Section 5.

3.1 Aim: synthesize our understanding of vis-specific data cubes

The field of visualization-centric data structures is nascent, and its early developments have shown great promise. Unfortunately, it has so far failed to yield general lessons: the contributions tend to be generally quite different from each other, making appropriate comparisons difficult. In its current state, this lack of structure limits the potential impact of the area. Consider the following two basic, significant questions: Where are the gaps in the space of possible data structures? What are best practices for practitioners, and where are the pitfalls?

We propose to create a pair of artifacts aimed at organizing the field, making it more accessible to new researchers (and ourselves!), and bringing distinctions between current proposals into sharper focus.

Concretely, this aim entails two related activities: the creation of a **unifying scholarly review of vis-specific data cubes** and the design of a **benchmark for vis-specific data cubes**.

3.1.1 A unifying review of vis-specific data cubes

We propose to produce a unifying scholarly review of visualization-specific data cubes. Since the current research in the area is all quite recent and disjoint, there is actual computer science research in identifying unifying principles behind the different proposed techniques. Because of the PI's previous work in designing evaluation mechanisms for three of the state-of-the-art data cube structures [34, 44, 53], we believe that the PI and his research group are uniquely positioned to produce this research. An informal, preliminary effort towards this goal has appeared in a book chapter authored by the PI [46]. We believe, however, that the problem deserves a more thorough treatment:

Research questions. Which visualization systems currently do not benefit from visualization-specific data cubes, and why? Which capabilities are available to all proposed techniques in the literature? Which capabilities are specific to only some of them? Which capabilities exist in interactive visualization systems that are not well-served by visualization-specific data cubes?

This research aim is fundamentally connected to the broader impacts of the proposal. Specifically, we believe that visualization-specific data cube techniques have not been adopted very widely in statistical programming languages. Why is that? A unifying review will help clarify these issues and facilitate future adoption.

3.1.2 A benchmark for vis-specific data cubes

We propose the creation of the benchmark for vis-specific data cubes in the vein of TPC-H, a benchmark for ad hoc database queries designed to stress decision-support capabilities in RDBMS systems [4]. Specifically, we will design, implement, and provide to the research community a *dataset generator* and a *workload generator*. The dataset generator creates (possibly many) sets of data points used to populate the data cube. The workload generator creates a large number of queries that will exercise the various aspects of vis-specific data cubes. This will entail a research publication describing the benchmark methodology and a collection of open-source programs to generate data and queries.

As a source of plausible data, we will intend to use three existing datasets. For small-scale experiments, we will use a dataset of 4.5 million user checkins to Brightkite, a now-defunct location-based social networking website [33]. For medium-scale experiments, we will use the Bureau of Transportation Statistics's On-Time

Statistics dataset, which has collected flight delay data from more than 160 million flights from 80,000 airports and dozens of airlines over a period of 25 years [13]. Finally, for large-scale experiments, the NYC Taxi and Limousine commission has provided trip data for six years, from 2009 to 2015 trip data, comprising more than a billion real-life taxi trips [42]. The PI has experience with these datasets, and has used them in evaluating previous research contributions [34, 44] and in (unfortunately not publicly-available) research demos.

We note that we will also allow the standard solution of "scale factors", which involves inserting multiple copies of the data into the dataset. We provide a wider variety of datasets in order to allow for proper, realistic, variation in the data distribution, since this can have a large impact in the performance of the proposals [34]. Specifically, we ask:

Research questions. How can we build a benchmark that effectively evaluates strengths and weakness of visualization-specific data cubes? Which queries model sufficiently realistic user interactions, while also stressing the computational resources available to the system?

Justification. Since TPC is a well-established organization and TPC-H exists, it is natural to ask: why do we need a new benchmark to begin with? In short, TPC-H does not exercise the types of queries that are generally issued by visualization systems: correlated sequence of overview and drill-down queries that are multiresolution in space in time, both in the stream of queries and in the relational schema [10, 15]. Because of that, using TPC-H to evaluate vis-specific data cubes will cause a mismatch between the reports, unfairly penalize plausible visualization data-cube proposals and unfairly rewarding proposals which do not optimize for the queries which we *do* observe in visualization systems. The present lack a of visualization-specific benchmark, then, makes it impractical to meaningfully compare current and future proposals.

Evaluation. The success of our benchmark proposal will hinge in its ability to capture the capabilities of different techniques — hence the concurrent need for the unifying review. We will consider our benchmark successful to the extent that it highlights the differences we intuitively know to exist in the landscape of vis-specific data structures. To give a necessarily-simplified example, we know from the respective literature that nanocubes take in some cases too much memory [34], that hashedcubes might (even if rarely) yield slow query times [44], and that imMens can have limited resolution in its schema [36]. Because there has been no appropriate benchmark, however, *these systems have actually never been compared directly against each other*. Our benchmark will allow us to rectify this situation.

Previous research results by the PI. In a recently-accepted paper [44], we used interaction logs of live systems (specifically, the nanocubes demos running at `http://nanocubes.net`) to build a set of plausible queries with which to compare HashedCubes and Nanocubes. We will use the same logs as the basis for the query sets to be used in the benchmarks. Note that although this research goal is among the first we intend to pursue (see Section 5 for details on the timeline of this proposal), the benchmark might change slightly to incorporate new details as our understanding of the algorithms improve. These updated versions will be published on our research group's website, which will serve as the central point of access to the materials or other researchers.

3.2 Aim: characterize, more finely, space-time tradeoffs in spatiotemporal data cubes

Although data cubes are typically described in terms of a fully-materialized table (such as the one in Figure 4) from which any query can be answered by a single lookup, we note that the crucial aspect of the data cube is its *API, together with performance guarantees*. As long as a data structure enables fast answers to a wide gamut of aggregation queries, it should sensibly be considered a data cube. In other words, what disqualifies a potential "zero-memory datacube" algorithm that always just issues complete linear scans of the underlying tables is its performance: behavior-wise, it is exactly as valid as the materialized table. This means that there will always be tradeoffs, and these tradeoffs must necessarily be considered in the context of a potential query

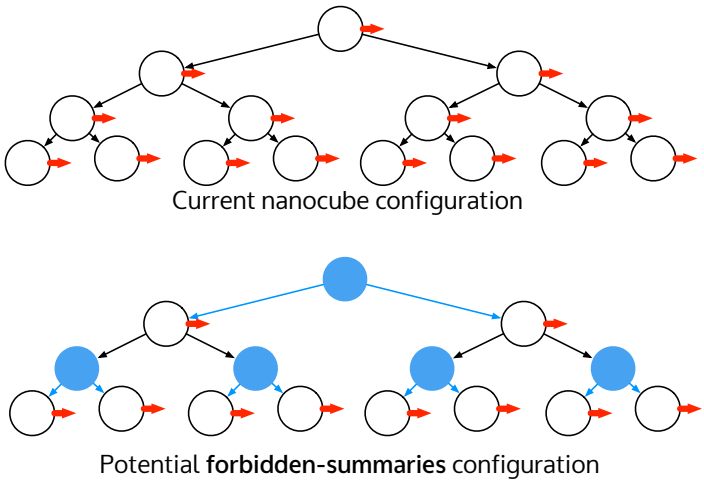


Figure 7: Above, we show an example of a single dimension of a nanocube. Red edges denote to pointers to subgraphs that partition the *next* dimension; we call these *next* edges. In current designs, every dimension node has valid *next* edges, which causes large memory consumption. One possible design is to limit the number of such next-dimension subgraphs: in the *forbidden-summaries* setting, blue nodes lack *next* edges, and so the entire subgraph they point to never needs to be determined. At the same time, aggregations for these nodes can still be computed in constant time by performing the query on both refinement edges and combining the result.

set: what and how much gets precomputed will affect only a certain subset of queries.

Research questions. Are there query sets that cannot all be answered quickly without incurring impractically large storage costs? Where are the storage inefficiencies in the way these indices are presently built?

Justification. Nanocubes offer an attractive point in this design space, but clearly take prohibitive amounts of memory in some situations (on the order of 40GB to index a dataset of 200 million geolocated tweets and four additional dimensions). Worse yet, we currently only know how to improve this memory usage by either not indexing along some columns and reducing the overall resolution of the dimensions. Although the current state-of-the-art in the characterization of space-time tradeoffs in data cubes appears to be the cross-column lattice characterization of Harinarayan et al. [27], the design space is largely unexplored in the context of *within-column* optimizations. We propose here one general idea that merits further exploration, and that has the potential to offer significant memory savings.

Forbidden summaries. In the context of the within-column aggregation that happens in a nanocube, there appears to be an *overabundance* of computed summaries. Every time a node in a dimension has two or more refinement children (we call this an "inner node"), its own *next* edge points to a partially non-shared node. This trivially guarantees that, for any node, it will be possible to compute its aggregation by simply following the chain of *next* edges. However, imagine the case in which *one single inner node* was missing its own chain of *next* edges. In that case, we could simply compute the aggregation for its refinement children (which we know would exist by assumption), and then aggregate these results. We take a constant-factor hit in performance (having to combine a pair of aggregations instead of a single lookup), but the runtime is still $O(1)$. So, as long as we can guarantee that the refinement children have *next* edges of their own, we can prevent some of the inner nodes from precomputing their own *next* edges. We think of these as "forbidden summaries" (see Figure 7). There are at least two promising strategies we will investigate: first, we can forbid summaries uniformly at random. Assuming a small p (so forbidden nodes don't interfere with one another), a back-of-the-envelope calculation gives that the overall structure will take $(1-p)^k$ as little memory, while

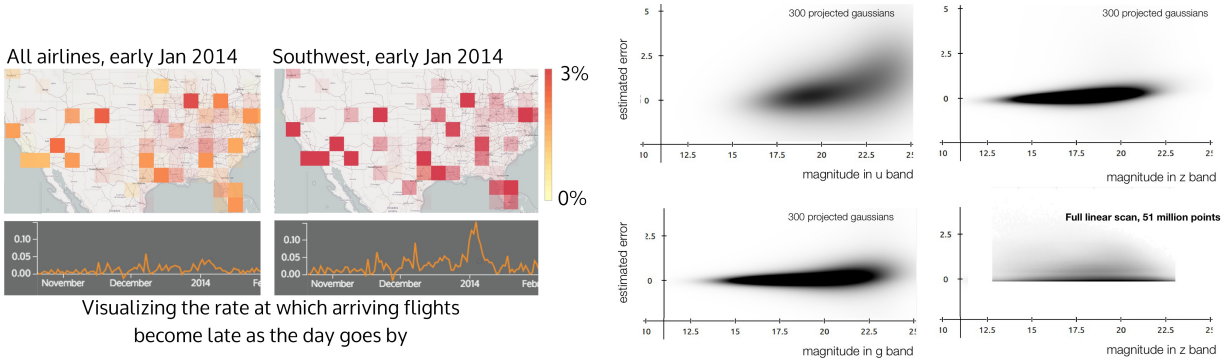


Figure 8: Left: Interactively visualizing model fits over 160 million flights. Live demo available at `http://bit.ly/2a1mTix`. Right: Interactive, approximate KDE plots over 55 million stars of the SDSS Data Release 7 (both figures adapted from Wang et al. [53]). These figures show the potential for incorporating data cubes into other analysis and visualization techniques.

making the queries take $(1-p)^k$ as much time. Since the majority of nanocube queries take well under 1ms [34], a factor of 10 increase in running time would not affect interactivity, but would fundamentally improve the usability of the structure. Another possibility is to take inspiration from tree-labeling systems such as those in red-black trees [20]. A concrete design and analysis of these rules will be carried as part of this research aim, but the general idea is to create node labels and insertion rules to maximize the number of forbidden nodes while guaranteeing the presence of next edges in those nodes.

Evaluation. A successful implementation of this aim will provide users of nanocubes with tunable parameters to control the desired query runtime, and memory usage. In order to evaluate our proposal, we will use the datasets from the benchmark we proposed earlier to measure construction time and memory usage, and the benchmark queries to evaluate query time.

Threats to validity. This is the riskiest aim in the proposal. It is unlikely, although possible, that the outcome of this research aim is shows that it is not practically possible to achieve the results we set out. As such, we are going carry it on early during the proposal, in order to allow us to change direction if necessary. The main way in which we foresee a change in direction is by integrating preaggregation strategies with sampling, such that (for example) the approximate aggregate are obtained from partially-computed summaries, which would take less storage and construction time.

3.3 Aim: derive novel techniques for interactive analysis

In this aim, we seek to use these data cube structures as the single point of access for interactive analysis. Specifically, we envision providing access data access to analysis algorithms only through a nanocube.

Research question. Which data science algorithms can be implemented naturally while only having access to hierarchical data cube aggregations? In this setting, what new techniques become possible? What properties must we give up?

Previous research results, including work by the PI. As it turns out, a large number of important techniques in statistics, machine learning, and data analysis all have *monoids* embedded deep into their structure. This property has been exploited in the past to create variants of data cubes for regression problems, etc. [16–18, 41, 53]. However, the computational consequences of this have only recently been explored at depth by Izbicki [31]. In one class of problems recently tackled by the PI [53], the sufficient statistics to compute multivariate gaussian fits form a monoid: the means and variances can be computed by accumulating sums of second-degree polynomials over variables of interest. From these, it's possible to solve a variety of data analysis problems by accessing only the aggregated value, completely avoiding a linear scan of the

array, while at the same time allowing for the interactive querying that is possible with our data structures. In addition, we have shown the ability to create *visually-approximate kernel-density-estimation plots* by traversing the nanocube data structure adaptively (we refer the reader to Wang et al. for details [53], and Figure 8 for an example).

Justification. Interactive analysis involves more than just linear model fitting. We seek to develop, among other things, aggregation-centric versions of classic techniques such as clustering and supervised classification. As a sketch of a possible implementation of k-means, consider its traditional energy minimization formulation [11], where we seek to minimize $\sum_i \sum_{x \in X} ||x - \mu_i||^2$, where μ_i are the i cluster centers, and x are the data points (we abuse notation slightly by assuming the inner sum implicitly happens over points assigned to the closest cluster center). At its heart, the k-means minimization procedure repeatedly evaluates the derivative of this error term to find locally-optimal cluster assignments. These are linear scans over the data points, but if a potential cluster center is sufficiently far away from a set of values in a given nanocube dimension, then it will be possible to bound the error of the term by simply evaluating a term based on the summaries. This kind of strategy is directly reminiscent of the classic Barnes-Hut many-body $O(n \log n)$ solver [9]. We believe it is possible to use similar techniques for other data-mining procedures, but will focus on clustering as a target problem to begin with.

Evaluation. The main difference in evaluating this research aim, compared to the previous ones, is that the algorithms in this stage are *approximate*. We will need to determine the extent to which the approximations induced by the approximate traversal change the overall result of the clustering procedure. For that, we will use a combination of synthetic and real-world data, again using the data from the benchmark proposed earlier.

Threats to validity. What if turns out that we cannot practically construct data cubes fast enough to enable these algorithms at the quality bounds we seek? Similarly to the threat described above, we propose an alternative course of study in the investigation of the behavior of *sampling* algorithms, as they relate to data science techniques like k-means clustering. One can foresee hybrid variants, in which coarse aggregates are pre-computed at low resolution, and data partitioning is used for sampling within the aggregated regions. This is reminiscent of the strategy used by BlinkDB [5], where the main difference is we are building these coarse "aggregation-and-sampling" structures as nanocubes, respecting the sharing relationships and reusing data partitions.

3.4 Aim: improvements to the core data structure, and integration with existing interactive analysis systems

At this point in the project, we expect to have better control over the performance characteristics of vis data cubes, in addition to a number of implementations of cube-based data analysis techniques. It is important to realize, however, that most practitioners of data analysis use some kind of analytics library on top of Python (like IPython and Pandas [39]), or a specialized language like R [30]. As a result, unless our proposed techniques can easily be incorporated into these existing pieces of infrastructure, adoption and impact will remain low. As we will argue below, this is not only a problem of dissemination: incorporating of these data cubes into a wider range of infrastructures raises specific *research* questions.

Research question. What are the fundamental barriers stopping data cubes from being incorporated into the broader data science pipeline?

Acquisition at scale. The first problem we intend to tackle is that of *data acquisition*. In the case of simple data structures like linear arrays, the process to go from raw data in one machine to a table in a different machine is simple. However, to the best of our knowledge, the current proposals for visualization data cubes are either *in-process databases* and do not support writing and resuming from disk (as in the case of nanocubes), or *cannot support data updates* (in the case of HashedCubes and imMens). As a starting point to this research aim, we note a striking similarity between nanocubes and Bagwell's Array-Mapped Tries and

Hash Trees [7, 8], the basis for Clojure's purely-functional data structures. Purely-functional data structures are typically *mergeable* [43]: two disjoint data structures that can be easily combined into their union. We propose to investigate, then, *array-mapped mergeable nanocubes*: this will allow direct serialization to disk, data updates, *and* parallel acquisition.

Partial transmission of nanocubes. One of the marked advantages of imMens over nanocubes is the ability to send *data tiles* [36]: the imMens data structures decompose naturally in pieces that can be transmitted from server to client, and the query engine can then be implemented by referring entirely to the tiles, vastly reducing network communication. We will investigate *tileable* versions of nanocubes by decomposing the shared forests (see Figure 5 for a reminder) into independent subgraphs. If the data structures are array-mapped as described above, then these tiles will be easily transmissible from one process to another and, as long as typical nanocube queries touch a small number of these graphs, few tiles will be required to answer any one query. Finally, if each subgraph exhibits good locality, then related queries will tend to use related tiles, again reducing the overall network transmission. In order to build these tiles, we will use techniques similar to cache-oblivious mesh layouts and graph partitioning [32, 57, 58].

Evaluation. We note that the combination of the two aims above means that nanocubes will be representable by a collection of arrays, a structure that's very simple and efficiently stored in a variety of programming languages. The success of this aim will be assessed by the extent to which we successfully integrate our infrastructure in a number of different environments. The PI and his research group have extensive experience in interfacing with low-level Python, R, and Javascript software [2, 3], and so these are the languages which we will initially target for integration.

4 Broader Impacts of the Proposed Work

Dissemination: incorporating the research in existing coursework. The PI regularly teaches a senior-year undergraduate course in data visualization, a graduate-level course in data visualization, and a graduate seminar on research advances in large-scale data visualization. If computer science education were not already limited by the number of possible credits in a four-year course (and two to three more years of coursework in a PhD program), a simple dissemination avenue would be to require students to take both a graduate-level database course, a graduate-level data visualization course, and then cover the remaining topics in a graduate seminar. This is effectively how the PI currently trains his current students. However, in the PI's experience, the field of large-scale interactive data analysis, currently, is simply too scattered to cover in a single seminar. Although there is intrinsic intellectual value in the unifying review proposed in Section 3.1.1, one important broader impact of that aim is to simply *allow for more structured dissemination*.

More generally, the PI's current experience with the dilemma of "choose one: performant or pedagogical" was a major driving force behind this proposal. We clearly cannot hope, in the scope of the current proposal, to entirely change how practitioners engage with large-scale interactive data analysis. Nevertheless, it remains the case that a ground-up proposal for a performant, large-scale interactive data analysis pipeline can substantially influence the design of future libraries.

Independent study, and undergraduate research preparation. The Department of Computer Science at the University of Arizona (the PI's home institution) has a strong culture of encouraging undergraduate and graduate students to engage in independent study credits. In the two years since joining the department, the PI has advised five separate projects. One of these was led by a third-year undergraduate student, and culminated in a publication at IEEE VIS 2016. We intend to continue to use these projects to increase the impact of visualization and data analysis. As an example, one current ongoing project involves studying the performance of existing database systems and the impact of performance optimizations, clustering and comparing physical query plans over many thousands of different database configurations. We have used these projects as ways to find research gaps in our field, and increased the reach and impact of the field of

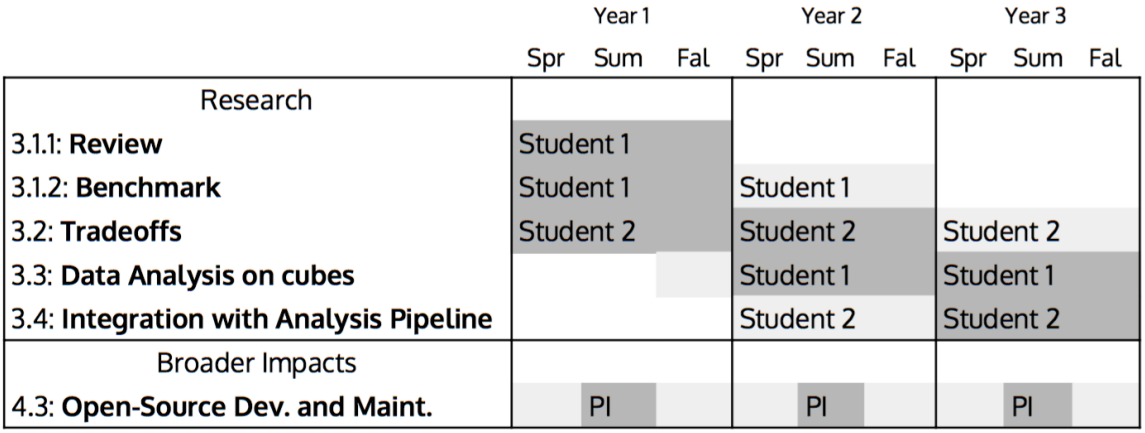


Figure 9: Timeline for activities in this proposal.

interactive visualization: many of these students were first exposed to data analysis and visualization through independent study. As the PI's research group grows, we foresee the value in proposing an elective, one-credit course meant to teach undergraduate students basic research skills. This course would teach best practices in research and methodology: how to read and write research papers; how to run a valid system evaluation; how to communicate with your peers, and so on.

Open Source and Open Science. All software developed as part of this research will be released as open-source, as I (and the groups I have participated in) have done in the past. This is not only an ethical choice, but a practical one, especially for a nascent area. It provides shared infrastructure for other researchers to build on, or even to improve and supersede in their own work. The PI, together with his students, will maintain the open-source software using typical licenses (likely MIT or GPL, depending on software dependencies to be determined in the future), and hosted at the organization's GitHub repository. The PI's research group also has a web page that publicizes the work of the PI and his students, and that serves as a central point of access of research pre-prints[1].

5 Proposal timeline

In this section we offer a rough planned timeline of the proposed activities. Of course, this being a research proposal, the actual timeline is subject to a variety of contingencies. The proposal here is meant to illustrate the general dependency structure of the activities and the pace of the overall project. Later activities are more likely to change based on the unexpected lessons we will inevitably learn from the early activities. The light-gray marks on the Gantt chart below indicate periods of *partial effort*, where we expect the PI and his graduate students to spend some amount of time, but not as a central part of their research agenda. The dark-gray marks indicate *concerted effort*.

6 Results of prior NSF support

PI Scheidegger PI Scheidegger has one active NSF grant: "III: Medium: Collaborative Research: Topological Data Analysis for Large Network Visualization" (Award number 1513651, $268,933.00, Sep. 1st 2015 – Aug. 31st 2019). This research proposes to connect large network visualization with topological data analysis,

[1] `http://hdc.cs.arizona.edu`

using the latter to improve the former. The current grant has partially supported the PI and has supported a PhD student. The support has currently led to four research manuscripts, two of which co-authored by students supported in this grant, and published at the top visualization conference.

Intellectual Merit Although large networks have become ubiquitous in physical, biological and social sciences, it is still hard to effectively make sense of them. Topological data analysis offers a fundamentally new angle of attack, and the combination of network-specific filtrations and algorithms for interactive visualization have already yielded preliminary results which are currently being extended for submission to research venues.

Broader Impact Networks are an exceedingly popular structure for representing relationships between individuals, businesses, telecommunication systems, genomes, and so on. The approaches we propose will ease the challenge of analyzing this large, complex, yet highly relevant data source. We will further disseminate this new perspective of network visualization through the release of the tools and datasets to scientists and the general public. Current graph visualization tools and systems find applications in areas including epidemiology, law enforcement, public policy, marketing, and finance, where the scale and complexity of datasets overwhelm the present state-of-the-art. We expect our alternative approach on the problem to have a transformative effect on the field.