Interactions with Big Data Analytics

Danyel Fisher
Microsoft Research | danyelf@microsoft.com

Rob DeLine
Microsoft Research | rob.deline@microsoft.com

Mary Czerwinski
Microsoft Research | marycz@microsoft.com

Steven Drucker
Microsoft Research | sdrucker@microsoft.com

"When times are mysterious
Serious numbers will speak to us always.
That is why a man with numbers
Can put your mind at ease.
We've got numbers by the trillions
Here and overseas."
—Paul Simon, "When Numbers Get Serious," 1983

Increasingly in the 21st century, our daily lives leave behind a detailed digital record: our shifting thoughts and opinions shared on Twitter, our social relationships, our purchasing habits, our information seeking, our photos and videos—even the movements of our bodies and cars. Naturally, for those interested in human behavior, this bounty of personal data is irresistible. Decision makers of all kinds, from company executives to government agencies to researchers and scientists, would like to base their decisions and actions on this data. In response, a new discipline of *big data analytics* is forming. Fundamentally, big data analytics is a workflow that distills terabytes of low-value data (e.g., every tweet) down to, in some cases, a single bit of high-value data (Should Company X acquire Company Y? Can we reject the null hypothesis?). The goal is to see the big picture from the minutia of our digital lives.

It is no surprise today that big data is useful for HCI researchers and user interface design. As one example, A/B testing is a standard practice in the usability community to help determine relative differences in user performance using different interfaces. For many years, we have used strict laboratory conditions to evaluate interfaces, but more recently we have seen the ability to implement those tests quickly and on a large population by running controlled

experiments on the Web [1]. These experiments allow practitioners to identify causal relationships between changes in design and changes in user-observable behavior on a potentially massive scale. Even more recently, practitioners are starting to identify usability issues by mining query logs for commonly asked questions by users of certain applications [2]. This can help product teams discover large, real-world usability issues while supplementing laboratory techniques that tend to focus on smaller, more isolated problems.

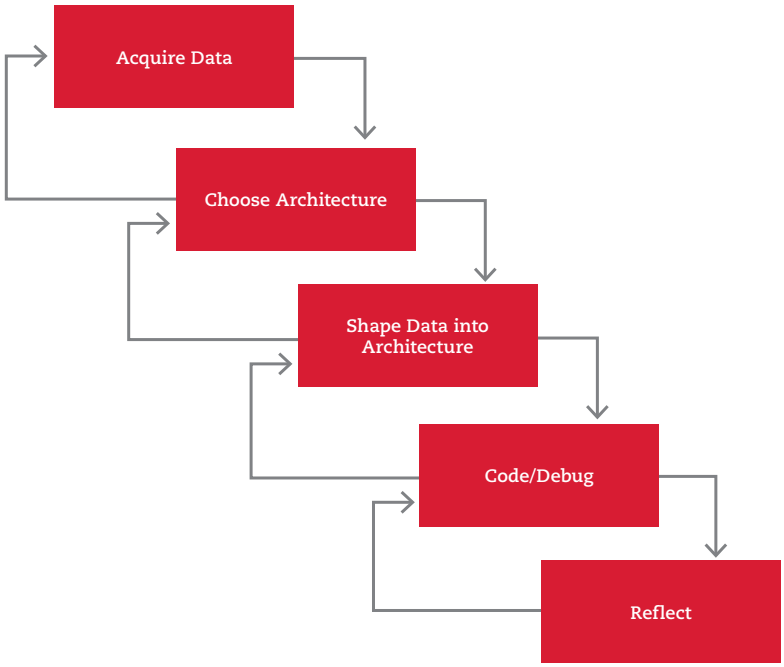
Other companies use the data more directly to modify their offerings. The online game company Zynga creates games and studies data on how its audience plays them in order to update the games immediately. "We're an analytics company masquerading as a games company," said Ken Rudin, a Zynga vice president in charge of its data-analysis team. He continued, "We are totally disrupting the traditional video games industry; a huge portion of that disruption is the ability to use data" [3].

Of course, big data analytics, like any research method, has its limits and pitfalls. Just because analysts have big data to work with doesn't guarantee the sample they need is sufficiently representative of their entire user population (bigger is not better); nor does it mean they have the ground truth around their users' motivations or needs from their behavior logs. For instance, boyd and Crawford argue that working with big data is still subjective and that automated data collection is not self-explanatory—it requires selection and interpretation [4]. They point out that the data sampling and cleaning processes in particular are prone to potential error and bias. So, the challenge for HCI researchers is to leverage the big data that will be increasingly available, but to do so judiciously.

Here we report on the state of the practice of big data analytics, based on a series of interviews we conducted with 16 analysts. While the problems uncovered are pain points for big data analysts (including HCI practitioners), the opportunity for better user experience around each of these areas is vast. It is our hope that HCI researchers will not only turn their attention toward designs that improve the big data research experience, but that they will also cautiously embrace the big data available to them as a converging line of evidence in their iterative design work. The big data user experience challenge will affect every one of us. As Pat Hanrahan, a professor at Stanford, recently said: "The reason big data is impacting every one of us is the data oozing out of everything… It's like electricity flowing throughout an organization—everyone can tap into it on command to answer the individual questions their jobs demand" [5].

The Nature of Analytics Work

The term *analytics* (including its *big data* form) is often used broadly to cover any data-driven decision making. Here, we use the term for two groups: corporate analytics teams and academic research scientists. In the corporate world, an analytics team uses their expertise in statistics, data mining, machine learning, and visualization to answer questions that corporate leaders pose. They draw on data from corporate sources (e.g., customer, sales, or product-usage data) called *business information*, sometimes in combination with data from public sources



▸ Figure 1. The big-data pipeline.

(e.g., tweets or demographics). For example, the CFO of a games company might turn to the analytics department when considering several schemes to sell "extras" to improve a game's revenue. To help this decision making, the analyst's role is to choose informative metrics that can be computed from available data, to perform the necessary computations, and to report the results in a way the CFO can comprehend and act upon. This role is marked by several characteristics:

- The work is exploratory and demand-driven. While some analytics work is routine, analytics teams need flexibility to deal with new and changing data sources, new types of questions, and new techniques and technology.
- The ultimate goal of the work is clear communication, often to an audience with little background in analysis techniques like statistics.
- The work must produce high-confidence results, often under pressure. Reporting the limits of the analysis is often as important as reporting the results. These limits arise both from a poor fit between the available source data and the "ideal" data for answering the question, and from the "lossy" nature of the data transformations. To reduce these limits, the analysts sometimes improve the raw data, for example, by requesting that the engineering team log specific product usage.
- The work creates a strong need to preserve institutional memory, both by tracking the origins of past decisions and by allowing repeatability across analyses. Larger analytics departments often share "tribal knowledge," including the meaning of data values (e.g, how missing values are represented), domain-specific analysis techniques (e.g., how to filter spam bots out of Web requests logs), and general analysis techniques (e.g., when to apply a correction in a statistical test).

In the academic world, research scientists analyze data to test hypotheses and form theories. Though there are undeniable differences with corporate analytics (e.g., scientists typically choose their own research questions, exercise more control over the source data, and report results to knowledgeable peers), the overall analysis workflow is often similar.

This article is an alert and a call to action. It is an alert that big data computing is coming to us all: As we collect data from sensors and stream it from logs, traditional usability testing is being complemented by big data analytics. Increasingly, the usability professional must learn how to do large-scale analytics to provide valuable insights into how millions of users are working on large-scale applications.

It is a call to action in that we highlight important UX challenges in handling large datasets. There is a great opportunity to make the analysis of big data easier to do and faster. While systems researchers are building ever-larger designs, there is a great need to improve the experience of doing analysis with these systems.

Defining Big Data

When does analytics become *big data* analytics? The size that constitutes "big" data has grown according to Moore's Law. In 1975 attendees of the first VLDB (Very Large Databases) conference worried about handling the millions of data points found in U.S census information [6]. In the context of information visualization, Shneiderman describes a dataset as big when it's too big to fit on a screen—at one item per pixel, most desktops would stop at a few million data points today [7].

More often, big data means data that cannot be handled and processed in a straightforward manner. A spreadsheet fits in memory; it is reasonably quick to determine if the data is clean, the values are reasonable, and the results can be computed rapidly. In contrast, a big dataset won't fit in memory, so it will be hard to check whether it is clean. Computations will take a long time. New data may well be constantly streaming in, so that the processing system needs to make decisions about which part of the stream to capture. The dataset may consist of images, natural language text, or heterogeneous data, so it will be hard to predict where the database join keys are.

Finally, a big dataset will probably be so large as to not fit on a single hard drive; as a result, it will be stored on several different disks, and will be processed on a number of cores. Queries will have to be distributed and written to work across a network.

An Old "New" Way of Analyzing Data

In many ways, today's big data analytics is a throwback to an earlier age of mainframe computing. To illustrate this, let's contrast the familiar interactive approach to analyzing data in spreadsheets with the brave new world of big data. If you had a dataset at the turn of the 21st century, you tended to copy your data to your disk, which typically took seconds or less. You would load the dataset into memory and then interactively perform one or more analyses on the data. There was no need to examine preliminary results and iterate, unless you were considering performing different tests or transforming the data. The whole process was very fast, and results would be ready in seconds. You knew right away if you were satisfied with your findings.

In today's world of big data, the luxuries of interactivity, direct manipulation, and fast system response are gone. You can't copy your data to your personal computer in whole; it's more than can be processed in real time, as well as more than can be visualized all at once, and it requires much work to get the right sample and amount of data. In some sense, with existing tools a data scientist may not really know if she got the most appropriate answer or not. Because of these issues, we feel the workflow of computing has taken a giant step backward—back to the punch cards of the 1960s! Most of the cloud systems used for big data analysis feel more like batch jobs, in which you submit a job and go get some coffee, with little insight into what's really going on behind the scenes, how long it will take, or how much it's going to cost.

Challenges Involved in Big Data

To understand the challenges of conducting big data analytics in more detail, we interviewed 16 data analysts at Microsoft. Each of them was working with large datasets and using a variety of cloud and distributed services to process their overwhelming quantity of data. As an example, one of our analysts was a social psychologist who works intimately with Twitter data. He receives the raw Twitter "fire hose" dump of data, often years' worth of it. He then analyzes the feed to study trends such as changing sentiment over time, or how information spreads through the Twitter feed. Several of the other analysts we interviewed do machine learning over very large datasets, for example, over all of the search queries coming out of the Bing search engine.

Although each analyst's workflow varies in specific ways, analyst activities are generally clustered into five steps, shown in Figure 1: 1) acquiring data, 2) choosing an architecture, 3) shaping the data to the architecture, 4) writing and editing code, 5) reflecting and iterating on the results. This sort of workflow is in line with other documented workflows, such as the software-development waterfall model.

These steps differ jarringly from the way data is handled in Excel: Cloud computation is fundamentally different from local computing. Whether Microsoft's Azure, Amazon's EC2, or a Hadoop cluster, parallel computation is a different way of building code. In these highly parallel systems, identical code is run on multiple virtual machines (VMs). Users typically first get their code running on a single instance that runs locally, and then deploy the code to a series of VMs that run remotely on a network. The data is stored across multiple servers to ensure that it can be processed as rapidly as possible; the code itself contains rules that help decide which machines execute the code, and in what sequence. Whereas computing against local data is fast and efficient, computing against data that is stored on a separate machine can be a slow operation.

In a spreadsheet, the user can choose the type of analysis on the fly or can export the dataset into the appropriate tool. Computation doesn't cost much in Excel, so cost planning is not very important. On a VM, each step costs a measurable amount of money and time, and different designs can have substantially different financial or temporal implications.

BIG DATA

A map of the frequency with which people in different places reply to each other on Twitter. The brightness of each arc is proportional to the log of the number of tweets from one place addressing someone in another place, with locations chunked to 20-mile squares. Communication is shown moving clockwise from the person sending the tweet to the person being addressed. Data from Twitter streaming API, May 15 – October 10, 2011.

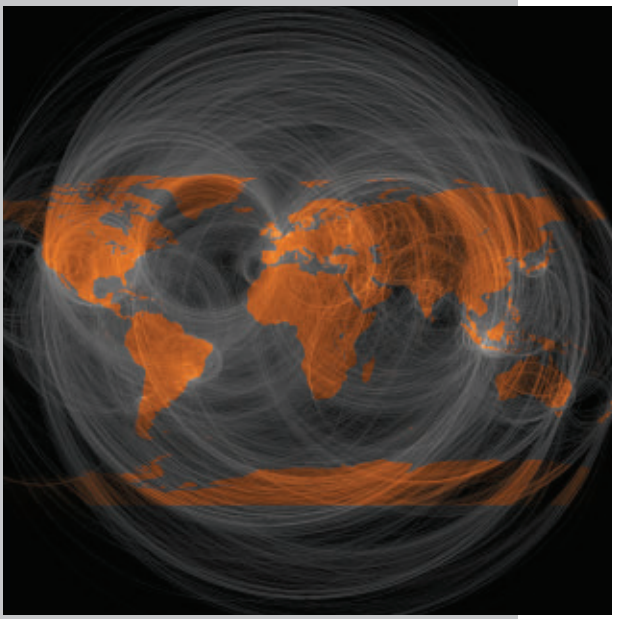


Image by Eric Fischer

See something or say something: Los Angeles. Red dots are locations of Flickr pictures. Blue dots are locations of Twitter tweets. White dots are locations that have been posted to both.

Image by Eric Fischer

This graph charts the frequency of mentions in the *New York Times* of the five U.S. presidents between 1984 and 2009. It also depicts story weights—the darkest lines shows front-page stories, the lighter lines indicate stories buried deeper in the paper.

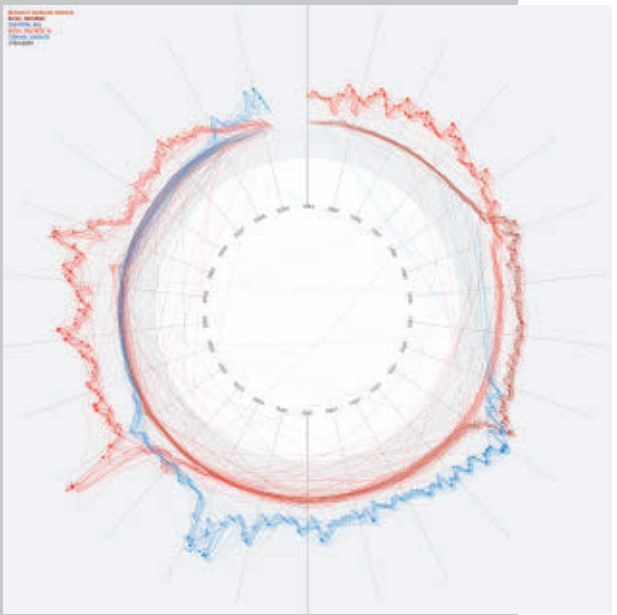


Image by Jer Thorp

We describe the challenges of each of these steps in turn, using examples from our interviewees to illustrate them. Each of these steps provides the HCI practitioner with ample room to improve the user experience.

Acquiring data. The first challenge our analysts identified was determining where the data in their big-data systems came from. How do they discover sources of data? Increasingly, data is available in a wide variety of sources and formats: Online databases of public statistics are provided by the U.S. government (http://data.gov) and the United Nations (http://unstats.org); private companies sell data from data marketplaces, such as Microsoft's Azure Marketplace and Infochimps. Experts ran into many problems with data available online, however. They struggled to figure out what data was available; even when it was available in machine-readable formats, it would often be stored in a schema that made it hard to use. In many of these systems, however, the data is available only after running an aggregation query—or worse, only in PDF files filled with text. Once this data was ready to go online, our analysts would combine it with information they collected themselves from sensor systems. This, in turn, caused new challenges: For example, it could take a third dataset to link the zip codes in a crime database to the area codes in a phone directory.

There are new opportunities for improving standards for announcing data, helping people find data, and formatting data so it can be more easily entered.

Choosing architecture based on cost and performance. Whatever platform the big data analysis is

performed on, the platform organizes the computation around a set of programming abstractions substantially different from those of the normal desktop environment. Analysts trained on the desktop environment have to learn these new abstractions and plan their computation around them, often facing a new set of engineering trade-offs and failure modes.

A completely new part of designing a data analysis for the cloud is planning for the economic impact of the design choices. With cloud computing, nearly every choice about computation, uploading/downloading data, and storage has a direct dollar cost. In addition, each choice has an effect on how long a job will take to execute. Planning and monitoring these costs is unfamiliar and poorly supported for end users, and making mistakes can be quite expensive. Many of these decisions need to be made before the first byte is uploaded to the cloud and before the first line of code is written.

Using cloud computing can support a broad selection of VMs; for more computation, you simply pay more, by buying either more machines or larger ones. Doubling the memory or computation speed of a machine, however, does not double its speed; it can impose non-linear costs as communication overhead, storage, and other aspects change. For example, in certain systems, a developer who takes a larger-scale VM gets access to lower-level systems of the machine and better guarantees of performance; smaller VMs do not get this access.

There is no support for estimating the cost or the duration of a computation before performing it. Programmers end up iteratively re-running their application to adjust the number of VMs, the size of queues, and so on, incurring larger bills while empirically finding the right time-cost balance point. When working with VMs in a shared cloud, their performance characteristics may even vary over repeated experiments.

Current cloud-computing platforms provide little to no support for partitioning an application across VMs. Users must do their own empirical measurements, gauging not only how long a job takes to compute, but also how much overhead is involved in starting a job on a new VM. In our users' projects, this overhead could be substantial, and so the cost of a job was often needed to factor in this long startup time. The time a job takes is affected by both the startup time and the computation time; adding more cores reduces the latter but doesn't affect the former.

Our analysts found it difficult to estimate the effects of these different potential configurations on task and computation time.

Shaping the data to the architecture. Once the analyst has found a dataset and a computing platform, he or she must upload the data into the platform. The analyst must ensure that the data is uploaded in a way that is compatible with how the computation will be structured, and distributed and partitioned appropriately.

Cloud-computing systems use data storage differently than desktop machines. Cloud systems are still evolving models for storage: They offer cloud-based databases (such as Amazon's RDS and Microsoft's SQL Azure), distributed file systems (as in Hadoop), and novel data structures (as in Azure's queues and blobs). These structures are meant to be adaptable to users' coding needs, but they represent a way of thinking about storage and communication that is distinct from what users have previously experienced. These structures hide underlying structure: A queue, for example, is a distributed data structure across a number of machines, and a partitioned table is silently split across multiple machines.

Several of our interviewees commented that moving files back and forth between the cloud and a local machine was extremely common but a huge pain point. When the files are large, efficiency is critical, so it's not simply an issue of finding some way to get files uploaded, but an issue of doing it right. The files need to be organized, partitioned, and prepared before they are uploaded to the cloud. Different techniques for importing can take very different amounts of time; for example, converting files to a binary representation rather than a comma-delimited text file could radically change the ingestion process on the far side.

Currently, tooling is limited: Analysts are used to a rich set of tools for handling local files, including moving and renaming them, looking at their content, and referring to them from applications. Each of these steps is more complex in the cloud. Our experts had difficulty inspecting and working with files once they had been moved up into the cloud.

Once data has been uploaded, it must be cleaned. Cleaning can be a difficult process, requiring multiple people's expertise; as a result, some of our analysts had collaboratively cleaned their dataset. Unfortunately, it was then often difficult to understand who had touched which part of the data. In particular, analysts found them-

selves jury-rigging mechanisms to capture descriptions of what the data cleaner was trying to accomplish and what features were used.

Fairly often, cleaning was a process that occurred only after code had been written and errors hit: They would go back and look for aberrations after a crash, or when a model looked odd. As a result, the analysts also wanted to capture the justifications for cutting aberrant data points: How had they detected the error? What model had found the issue?

Most data tables have the built-in notion that data should be stored and edited in place. Interviewees stressed the expectation that data storage is comparatively cheap: Rather than mutating it in place and losing history, they would prefer to create additional clean versions of columns and new datasets.

Write code. With an architecture selected and the data in place, the analyst begins to select their analysis. In the examples we studied, the analyses were articulated through code, written in C# and Microsoft's SCOPE; outside these environments, analysts might work in languages such as R, Python, or PIG (a database-like language), usually over Hadoop. High-level languages that make it easy for the compiler to support parallelism—such as DryadLINQ or Matlab's matrix-based language—will ultimately help users write cloud-based jobs.

Users must design their code and systems around the idea of separating their work into parallelizable jobs. Algorithms need to be written in new ways in order to do this, and data must be stored differently. For example, some resources might need to be duplicated, one per node. In order to reduce costly communication, an analyst might store a copy of a reference lookup table on each VM.

Abstracting away the cloud. These high-level languages are designed to allow analysts to "abstract away" the cloud. Analysts should spend less effort considering where their data is being processed, and more effort considering the nature of computation. Unfortunately, the abstraction can be leaky. Several of our users complained of times when one process or another was blocked based on transient network issues. The symptoms of these issues can be masked by other failures, as redundant systems and parallel code attempt to recover. As a result, from the analysts' perspective, sometimes the work simply stops, without a clear message; later, the system spins back up and the lag time is not accounted for. So a fine balance between transparency and abstraction seems to be required.

Directly manipulating data versus scripting. Our analysts are accustomed to working in Excel for their smaller data jobs. As a result, they often compared their procedures to working in an Excel-like tool. Needless to say, the cloud was found wanting: Cloud analysis is far more complex than when using desktop tools, even when the type of analysis is similar. They hoped for direct-manipulation environments.

On the other hand, several analysts emphasized the importance of scripting an analysis rather than carrying it out "by hand" through a direct manipulation interface. (R is an example of a scripting interface; SPSS is a direct-manipulation environment that generates a script in the background.) Scripting leaves a log of analysis, which makes it easy to repeat later or to share with teammates. Logs also make it easier to repeat experiments, to try variants when new questions or new data arise, and to fix mistakes in later runs. Finally, scripts document the derivation of high-level information and charts.

Of course, neither of these is available now; instead, analysts write compiled code.

Debugging and iteration. After the execution run is complete, the analyst wants to know if their test worked. This leads to a process of debugging and looking for errors, iteration and changing code to work, and visualization, in order to interpret results.

Code rarely works the first time. While modern programming environments will usually break into a debugger on a crash, a cloud-based computing solution will often be far more difficult to debug. Our users reported that they often found themselves poring through trace files, which reminded some respondents of debugging in the 1980s. While it is simple to write out trace files, a job distributed across a group of VMs means that a single crash might be distributed across multiple VMs, with trace files stored on a variety of machines.

This is compounded by the temporary nature of VMs. If a VM fails or stops responding, the infrastructure cleanly recovers from the fault, moving jobs to a different machine. Unfortunately, the transparency of this process can have the effect of hiding errors when they occur, making it a challenge to diagnose and eliminate them.

Data analytics is inherently exploratory, which makes rapid iteration highly desirable. After a job completes, for example, an analyst may want to tweak

a parameter and try again. In machine learning, for example, a user may want to check different combinations of parameters and feature sets to run after running one iteration suggests different features to try out. As one interviewed analyst put it, "Fast iteration is key but incompatible with the way jobs are submitted and processed in the cloud." It's frustrating to wait for hours only to realize you need a slight tweak to your feature set.

Analysts moved back and forth from local machines to cloud-based systems. For an iterative machine-learning process, some of the feature generation could be done locally, but the raw data then needed to be uploaded to the cloud. If the feature set was not correct or of high enough quality, they needed to go back to the raw data. They would sample data to a local machine, which they would use to both explore features and test models they had generated from a full run-through. This back and forth from local to cloud was poorly supported.

Most, if not all, interviewees stressed the critical role of visualization ("Visualization is huge" was a frequent comment). The common need was that of inspecting data at multiple scales. With large datasets, one must frequently start wide and zoom into very small details. Visual interfaces are extremely well suited to this scenario and can be used in conjunction with statistical analyses (e.g., suggesting to the user a number of clusters to specify in a cluster analysis). A related scenario is the need to see context. Especially in large datasets, getting lost in the data is easy. Visualization provides a way to maintain context by showing data as a subset of a larger part of the data, showing correlated variables, and so on. Visualization is also relevant to data streams that are now common, in that they can help identify patterns over time.

Hope for the Future

This list of complaints and pain points can be exhausting. This new paradigm of computation has placed stumbling blocks before analysts who try to take advantage of the opportunities. Yet it also means new opportunities for better user experiences and improved tools for analysis and investigation. As one frustrated analyst asked, "Can we take a typical Excel user and empower them to become a data scientist?"

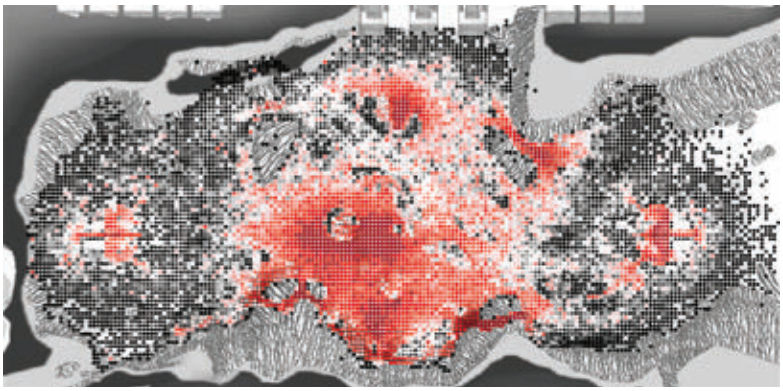
When we picture a system that addresses these issues, we believe this leads to a new way of writing queries and analysis scripts. Here, we imagine a hypothetical system, called an *analytics cloud environment* (ACE), which helps bring together these many ideas under one roof. Constructing this system requires critical contributions from HCI researchers, who can help understand the user experience of interacting with big data and can help figure out what aspects to bring to the fore—in the end, benefitting the process that HCI researchers will eventually need to participate in themselves!

To support rapid iteration on data in the cloud, the environment must be interactive, one in which users iteratively pose queries and see rapid responses. We already know about the power of interactive environments: Tools like R, Python, and Matlab are popular for data analysis. Projects like CONTROL [8] have begun to point the way toward online computation, in which large jobs are broken into small portions whose results are reported incrementally. Fisher et al. [9] point to ways to integrate visualization with online systems, allowing users to get visual feedback on their queries. This rapid iteration over partial results would allow analysts to catch bugs and explore alternatives more quickly.

Most interviewees stressed the critical role of visualization. The common need was that of inspecting data at multiple scales. With large datasets, one must frequently start wide and zoom into very small details.

The ACE would retain the team's institutional memory by allowing the user to save, share, browse, and reuse previous interactions with a data source. This would be richer than a standard R or Python shell—a tool that can introspect on the code it runs. This, in turn, may allow partial computations, letting users drill down into the workflow so they can use only part of a previous iteration.

Part of storing history is remembering where the data came from. The ACE would have an intrinsic notion of provenance. Inspired by tools like VisTrails [10], which today can remember both where a given data item came from and how users have changed their



▸ Figure 2. Heat maps of problem areas in Halo 3 (image courtesy of Microsoft Studios).

scripts, the ACE would store computation histories. A user could click on a result and see where it came from: what operations have been called on it, what queries it has passed through, and perhaps even what rows in the source data caused it to exist.

Contemporary high-level languages are increasingly being adapted for distributed computation, allowing analysts to express their queries as parallelizable entities. The ACE should similarly divide queries transparently among multiple machines. It draws on existing systems, such as Distributed-Computing Matlab, Hadoop's PIG, and DryadLinq, high-level languages that allow parallel computation. The ACE should come with a robust library of algorithms that are known to be efficient in the cloud.

Abstracting away the back-end does not mean hiding all execution details from users, particularly with respect to resources and failures. The ACE should allow users to feel fully in control of the resources and tools that the system is using. It is a challenge to visualization-oriented HCI researchers to figure out what makes for the best combination of tools for tracking and monitoring the states of storage, job execution and CPU utilization, computation rate and queue lengths, and errors for a user. This feedback should be brought in as an integral part of the ACE, so that when a user writes a line of code they can see how many rows per second it can compute—and how much it will cost if it continues to run.

Additionally, just as the back-end can be abstracted, so might the data itself. As stated, analysts we interviewed frequently worked with samples on their local machines before carrying out a full analysis in the cloud. This allowed them both to regain the interactive, exploratory experience of working with small data and insulated them from the complexities and expenses of computing in the cloud. Given the vast differences between the two computing environments, we feel that working locally with samples will continue to have productivity and economic benefits and deserves better support.

The ACE should be able to sample data from the cloud for local experiments and equally be able to push those experiments back up into the cloud. Sampling should also be a first-class activity, allowing users to create samples with desirable properties like statistical distributions or the ability to test corner cases of the scripts. This will steer users away from typical samples of convenience, such as the first 10,000 rows. In addition to sampling, users should also have robust tools for cleaning, previewing, and exploring their data both locally and in the cloud; groups of users working together should be able to distribute and make comments about their cleaning work.

One of the most valuable aspects of the language R is the substantial community that has grown up around it. Our analysts, too, did not work alone, but rather worked with teams. While permission models for cloud computing might still be primitive, the CSCW community has a rich history of supporting complex collaboration. The ACE should encourage collaboration around data, including allowing users to share not just scripts, but also data sources, versions of data sources that have already had a round of computation done on them, and even data runs that are in progress.

The ACE, then, is an umbrella vision for a system that smoothly lives both locally and in the cloud. Users enter commands, but rather than waiting for arduous computation to complete, they get incremental results as soon as they are available. It keeps a rich history of computation and provenance, allowing communities of users to understand how their data is being manipulated and how they are interacting with it. It provides high-level constructs for efficient coding against parallel systems and also allows users to introspect on costs and uses in detail. The challenge for HCI designers is to expose many of these capabilities to users in ways that empower sophisticated users without overwhelming them.

Supporting non-specialists. Universities are quickly creating advanced degree programs in analytics and minting as many new statisticians as they can; nonetheless, in the near term, demand will exceed supply. This shortfall creates the opportunity for tools to allow data-savvy end users to do their own analyses, without expert supervision. On the one hand, this creates the opportunity for rapid discovery, akin to citizen science; on the other hand, this exposes end users to the pitfalls that scientists are trained to avoid. Some pitfalls, like being careless with missing data values, misapplying statistical tests, or overfitting models, are inherent to the workflow, creating the opportunity to develop tools to recognize and mediate them. Other problems are more philosophical, such as confusing correlation with causation or ascribing too much importance to statistical significance. These larger pitfalls are typically avoided through apprenticeship with experts, which is difficult to apply to large numbers of people. If emerging tools allow end users to analyze large datasets without the rigor of scientific practice, the tools could lend the aura of science to otherwise poor conclusions. Societal issues with ample source data, like healthcare and climate change, could be acted on more on the basis of "truthiness" than truth.

Big Data Analytics Is the Future of Interaction Testing and Research

Here we have described big data analytics as an emerging type of knowledge work, with plenty of opportunities for study and productivity improvements. However, even for those who are not interested in this form of knowledge work, big data analytics cannot be ignored: It's an important new avenue to learn about how people interact with computing.

Product teams are already taking advantage of product usage data from tens of thousands to millions of customers. As one example, Bungie Studios recorded the deaths of all players of Halo 3 and produced heat maps to spot problems in game play [11] (see Figure 2). If too many players were dying in a particular game location, they could adjust the game, for example, by moving ammunition to a more obvious location. Another example is the creation of the Ribbon UI in Microsoft Office 2007 [12]. By analyzing the customer experience data from previous releases of the product (about 1.3 billion sessions), the Office team could make informed choices about the most commonly used features. This kind of analysis, which is based on simple tallies of operations, just scratches the surface of what is possible.

ENDNOTES:

1. Kohavi, R., Longbotham, R., Sommerfield. D., and Henne, R.M. Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery 18,* 1 (2008), 140-181; DOI: 10.1007/s10618-008-0114-1

2. Fourney, A., Mann, R., and Terry, M. Characterizing the usability of interactive applications through query log analysis. *Proc. of the 2011 Annual Conference on Human Factors in Computing Systems.* ACM, New York, 2011, 1817-1826; DOI=10.1145/1978942.1979205 http://doi.acm.org/10.1145/1978942.1979205

3. Wingfield, N. Virtual product, real profits: Players spend on Zynga's games, but quality turns some off. *Wall Street Journal* (Sep. 9, 2011); http://online.wsj.com/article/SB10001424053111904823804576502442835413446.html?KEYWORDS=zynga

4. boyd, d. and Crawford, K. Six provocations for big data. A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society (Sept. 2011); http://ssrn.com/abstract=1926431 or http://dx.doi.org/10.2139/ssrn.1926431

5. Woods, D. Tableau Software's Pat Hanrahan on 'What Is a Data Scientist?' *Forbes* (Nov. 30, 2011); http://www.forbes.com/sites/danwoods/2011/11/30/tableau-softwares-pat-hanrahan-on-what-is-a-data-scientist/2/

6. Simonson W. and Alsbrooks, W. A DBMS for the U. S. Bureau of the Census. *Proc. of the 1st International Conference on Very Large Data Bases.* ACM, New York, 1975, 496-498.

7. Shneiderman, B. Extreme visualization: Squeezing a billion datapoints into a million pixels. *Proc. of the ACM SIGMOD International Conference on Management of Data.* ACM, New York, 2008, 3-12; doi:10.1145/1376616.1376618; Key: citeulike:3009411

8. Hellerstein, J.M., Avnur, R., Chou, A., Olston, C., Raman, V., Roth, T., Hidber, C., and Haas, P. Interactive data analysis: The Control Project. *IEEE Computer 32,* 8 (Aug. 1999).

9. Fisher, D., Popov, I., Drucker, S., and schraefel, mc. Trust me, I'm partially right: Incremental visualization lets analysts explore large datsets faster. To appear in *Proc. of CHI 2012.*

10. Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva, C.T., and Vo, H.T. VisTrails: Visualization meets data management. *Proc. of ACM SIGMOD 2006.*

11. Thompson, C. Halo 3: How Microsoft Labs invented a new science of play. *Wired 15,* 9 (Aug. 2007).

12. Harris, J. Inside Deep Thought (why the UI, part 6); http://blogs.msdn.com/b/jensenh/archive/2006/04/05/568947.aspx

ABOUT THE AUTHORS

Danyel Fisher is a researcher at Microsoft Research working on information visualization. In his research he looks for new ways to explore and interact with data. His recent work has focused on applications for groups ranging from information workers to intelligence analysts, and from end users to visualization designers.

Robert DeLine is a principal researcher at Microsoft Research, working at the intersection of software engineering and human-computer interaction. His group designs development tools in a user-centered fashion, conducting studies of development teams to understand their work practice and then prototyping tools to improve that practice.

Mary Czerwinski is a research manager in the Visualization and Interaction for Business and Entertainment (VIBE) research group at Microsoft Research. Her research focuses primarily on novel information visualization and interaction techniques. She also studies information worker task management, multitasking, and awareness systems for individuals and groups.

Steven M. Drucker is a principal researcher and manager of the VUE group at Microsoft Research focusing on human-computer interaction for dealing with large amounts of information. He is also an affiliate professor at the University of Washington Computer Science and Engineering Department.

DOI: 10.1145/2168931.2168943