What Use is Statistics for Massive Data?

BY DIANE LAMBERT

Bell Labs, Lucent Technologies

Statistics in the broad sense is about extracting information from data. The common view of statistics is much narrower, though. Often it is seen only as a set of cookbook methods that are designed for small sets of data that are obtained according to a known design or sampling plan. The massive dynamic sets of data with tens or hundreds of gigabytes or even terabytes of data that are increasingly common in business, manufacturing, environmental sciences, astronomy, data networking and many other areas are felt to be beyond the domain of statistics. Moreover, the most visible challenges for massive data involve computing, which can lead to the view that computer science is more appropriate for understanding massive data than statistics is. This paper, however, argues that the discipline of statistics and thoughtful practitioners and researchers are still essential for extracting information from really big sets of data.

1. What is Statistics? Most people are first exposed to statistics in a required undergraduate course that is filled with a hundred or more other students who are also required to be there. Partially in response to demands from other departments, the introductory course focuses on traditional methods for the kinds of small experiments, studies and surveys that are part of the curriculum for other departments. Typically, nearly all, if not all, the datasets considered have fewer than 100 observations with only a few variables on each, and the objective is to apply simple methods for finding means, variances, confidence intervals and p-values and for fitting linear regression models. There is no discussion of computing, probably both for lack of time and because sophisticated computing is not needed. There is some discussion of mathematical properties of techniques, but usually not in ways that are intuitive. Inference centers on confidence intervals and p-values, which are simple to compute but subtle to explain. What most people take away from the course is that statistics is neither particularly hard nor interesting (except for some peculiar use of language), it is only useful for simple data and simple questions (if at all), and any one can apply it (although it is best avoided), but hardly anyone can understand it.

But statistics is much different, and not just broader or deeper, from what is taught in undergraduate courses. Simply stated, statistics is about extracting information from data that are noisy or uncertain. The unstated position is that all data are noisy. Twenty measurements from a small experiment are noisy, and zillions of transaction records in a data warehouse are noisy. The twenty observations from an experiment are noisy because measurement errors are unavoidable. Transactions have noise because the individuals generating them vary, and transactions for even just one individual vary. In statisticians' terms, transactions vary across individuals and vary within any one individual over time and space. If the data arise by sampling, so not all individuals are included, then errors and variability are introduced

by the process that determines which individuals or population units are included in the sample. Additionally, the data may have errors or be incomplete (even in the U.S. Census). Statistics, then, is about overcoming the uncertainty, errors and variability (noise) in the data to reveal the information hidden within.

How information is extracted from data depends on the goals. Sometimes, the goal is only to summarize the data at hand and not to make inferences about future experiments, future transactions, or the process that generated the data at hand. Even summarization may be difficult and require specialized algorithms if the data are massive. A more ambitious, and perhaps more common, goal is inference about unknown parameters and relationships. For example, the goal may be to understand how an underlying, unobservable quantity, such as power output, that cannot be measured without error varies as a function of other quantities, such as temperature and current. That is, the problem is to infer the form of the relationship and its parameters from experimental data. Or, the goal may be to estimate the fraction of current mortgages that will eventually default. Predicting future values, rather than inferring underlying relationships and parameters, is another common goal. For example, the goal may be to predict whether a particular applicant for a mortgage will default if the mortgage is granted, so data on current mortgages are only a noisy surrogate for the future data of interest. Both inference and prediction require data analysis, not just data summarization.

Statisticians take on many roles to analyze data. Some design graphics that bring out the structure of both the random and non-random components of data. Data visualization from the perspective of a statistician is discussed in [23], [10] and [11]. Interactive, dynamic data visualization for the purpose of statistical analysis is discussed in [20], with the ideas available in the package GGobi, for example. Statisticians have also designed flexible programming environments for exploring, analyzing and visualizing data. Two popular languages for statistical computing are S [5] and Lisp-Stat [22]. The 1999 ACM Software Systems Award was given to S because it "has forever altered how we analyze, visualize and manipulate data." Previous winners of this prestigious award include Unix, Teχ and the World-Wide Web. The ideas of S programming have been developed in the commercial package S-Plus from Insightful, Inc. and in the open source package R. Recently, a global collaboration has begun to build a statistical computing environment called Omegahat that is based on distributed components ([6] and [21]; also see the Internet site www.omegahat.org). The environment allows statisticians to access a rich set of tools, including visualization packages like GGobi, statistical computing languages like R, and database management systems like MySQL, from within one environment, removing the need to escape to different applications and to import and export data from one system to another to analyze data with task-optimized tools.

Most statisticians, though, spend most of their time working with data, breaking it down into deterministic and random noise components. Usually the first step is informal. The data are plotted in many different ways – with different filters, transformations, and views selected each time – to discover patterns and structure informally. Then those patterns are set aside, often by subtraction, and the residuals of the data from the pattern are examined visually in many different ways to reveal

any remaining structure or patterns in the data. Then that structure is subtracted, and the fit and subtract steps are repeated on the new residuals, on and on (with some backstepping, perhaps, to correct missteps in modeling) until no structure can be found. In this way, a statistical model is built empirically.

The basic statistical model is simply

$$data = mean + noise,$$

but both the mean, which is a deterministic representation of the data, and the noise, which represents the variability and noise around the deterministic model, can be complicated. The mean might be parametric, in which case it could be a linear, nonlinear, additive, or multiplicative function with only a small set of parameters (coefficients, for example.) Or, the mean can be nonparametric; *e.g.*, a step function (as in a tree), an assignment function (as in classification or clustering) or smooth as a function of a set of explanatory variables. The noise describes the variation in the data, which affects prediction and the reliability of estimates. The noise might be independently normal, correlated, long-tailed, weighted to account for non-random sampling, length-biased, spatial, censored or hierarchically structured, for example. If the noise does not affect the data additively, a more appropriate model is

$$data \sim F_{\boldsymbol{\theta}} \quad mean(data) = g(\boldsymbol{\theta}),$$

where $F_{\boldsymbol{\theta}}$ is a distribution that describes the random variability of the data around the deterministic model $g(\boldsymbol{\theta})$ for the data. $F_{\boldsymbol{\theta}}$ can be a well-known distribution, like the Bernoulli in the case of logistic regression. It can include correlation among observations, mixing to accommodate outliers, weighting to account for non-random (but probabilistic) sampling and length biasing, for example. The mean parameter $\boldsymbol{\theta}$ may depend on a set of explanatory variables or predictors, as well as on parameters such as unknown coefficients in linear or nonlinear models, and it can be as complicated for an arbitrary distribution $F_{\boldsymbol{\theta}}$ as it is for normal noise. Many kinds of models are possible, each representing both the structure and variation in the data (see [19], for example). The more complicated the process that generated the data, the more complicated the statistical model may need to be.

A basic premise that drives much of statistics is that modeling the noise component well is as important as modeling the mean structure well. The noise component is needed for prediction, to understand what is likely and what is not. It is also needed to understand how reliable the estimated mean or prediction is. The noisier the data, the less reliable the estimate. A basic tenet of statistics is that using an estimate or prediction without some sense of its reliability is downright dangerous. Although the range and nature of statistical models may seem bewildering to non-statisticians, a wealth of models is felt to be crucial for representing data when the goal is reliable inference or prediction.

Most statisticians have not dealt with massive data, though. Most do not have the necessary computing or database management skills; their skills lie elsewhere. But, it is possible to apply statistics to massive data. Section 2 describes what a statistician sees when looking at massive data and how that differs from what a computer scientist who works with massive data, often called a *data miner*, sees. Section 3 then takes a closer look at applying statistics to fraud detection, which

is a problem that has been tackled by many data miners from computer science. Section 4 offers final thoughts on the role of statistics.

2. Massive Data

2.1. *A Database View* Statisticians find probabilistic models a natural way to think about data, but not everyone looks at data that way. Another view, which is common in computer science, holds that the data in a database are not random or uncertain but an accounting of everything that has happened. Thus, the need from a data mining perspective is typically not to make inferences about an underlying process that is observed imprecisely, or to estimate deterministic and random models for the data, or to account for noise and uncertainty in the data to make inferences and predictions. Instead, from the data mining perspective, the need is to tabulate and process the data.

Statistics might then be used to approximate answers to queries that in principle can be known exactly, given enough computing power and time, but in practice are too costly to obtain (*e.g.*, [17]). How much was spent last month? How many people bought those two products in Florida last weekend? If all the data cannot be processed to answer the questions of interest, then the data can be sampled (not necessarily randomly) and knowledge of the estimating procedure and sampling scheme can be used to provide probabilistic bounds on the difference between the approximate answer and the answer that would have been obtained if all the data had been processed.

The notion that the data are not noisy, or at least noise does not have to be explicitly incorporated into models, is strange and naive to statisticians but not to computer scientists. Many if not most of the data mining tools that computer scientists use were not designed with noise in mind. Instead, a common goal of computer scientists in data mining is to find interesting patterns, associations and rules from massive sets of data without considering the process that generated the data at hand. For example, an interesting rule might be "People who buy beer between 7 p.m. and 9 p.m. buy diapers at the same time," or $X \wedge Y \implies Z$. The value of a rule is measured by its *support* and *confidence*, where support is the fraction of database elements that satisfy conditions X, Y, and Z and confidence is $support(X, Y, Z)/support(X, Y)$. This has the flavor of probability, in the sense that $support = P(X \wedge Y \wedge Z)$ and $confidence = P(Z|X \wedge Y)$ if each database record or item is given equal probability. The resemblance is only superficial, however, because there is no sense of randomness or uncertainty and the goal is to reason only about the data in the database, not about a larger population, judging performance solely by the fraction of a database that satisfies the rule exactly. Finding a good rule then reduces to finding frequent itemsets (which is a hard computational problem).

In contrast, a statistical approach would consider the data in a database as a sample (not necessarily random) from a larger population, such as all people who buy at convenience stores, or a random process, such as disease infection. A statistician would then define "interesting" in terms of a probability model, interpreting "interesting" as either the mean under the probability model or as current or future observations that are rare under the model, for example. This leads to two

important differences with a computer science based approach. First, what is interesting can be either rare (outliers) or common (the mean). Second, and perhaps more importantly, the statistical model can be used to evaluate rules that have not been observed in the data. That is, a statistician uses a model to *estimate* $P(Z|X = x, Y = y)$, even at unobserved values of (x, y, z), while the computer scientist relies on partitioning a database and *counting* records with $X \wedge Y$ and $X \wedge Y \wedge Z$.

The view that the data have no noise and extracting information from data amounts to finding frequent itemsets is appropriate in some contexts, but not all. It is not enough when the unit of analysis is small and the answer cannot be computed from the data with certainty, even if all the data are analyzed. Will this customer switch to another wireless service provider? How much will be spent on those two products in that convenience store in Florida next weekend? Noise is also important in dynamic modeling when older data are less relevant than newer data. In that case, a statistical approach allows gradations of uncertainty rather than just binary relevant/irrelevant decisions. Finally, relying solely on counting instances in a database violates the principle of "borrowing strength" or combining many small, different but related problems to make larger inferences. Questions that require going beyond counting instances are the ones that most interest statisticians.

But, processing and counting massive data is so difficult that it can overshadow data analysis. The forward to *Advances in Knowledge Discovery and Data Mining* states that

> Finding new phenomena or enhancing our knowledge about them has a greater long-range value than optimizing production processes or inventories and is second only to tasks that preserve our world and our environment. It is not surprising that it is also one of the most difficult computing challenges to do well. [24]

This quote often takes statisticians by surprise because, from their perspective, the challenges in knowledge discovery go far beyond processing, scaling up algorithms or using the right database management system. Traditional issues of data visualization and analysis are as challenging as computing is because the data can have complex structure and sources of errors and variability.

2.2. *A Hard Example: Web Analysis* Suppose the goal is to analyze the usability of a website and the data consist of the clickstreams of all visitors to the website over a period of time. The simplest approach is to tally the data, counting page hits broken down by time period or geography or referral engine, for example. Counting does not reflect how people use the website, though. Usability requires monitoring how a user traverses pages at the site and how the user changes when the website is revisited. Does a new visitor keep returning to the home page, perhaps because starting over seems the only way to get to the relevant location? Do people look at one page briefly and then leave the site? Do people who are referred from one search engine browse differently from people who are referred by a different search engine? Do people who visit a site frequently differ from those who visit once? How much revenue is generated during a visit to an online store? These questions require

thinking of the *visitor* as the unit of measurement, not the entry in a log file. This leads to thorny questions, such as what is a visit? Understanding the bias in the data is also important. Which visitors are of interest? If uninteresting visitors, like robots that grab all web pages, dominate the data then treating all visitors equally introduces bias. The fact that most people look at the home page and then leave may be important, but keeping these shallow visitors in all stages of the analysis may only obscure how the website is used by those who use it at all. Identifying the interesting visitors is difficult not just because there are many visitors to consider but also because "interesting" is a hidden label that changes as websites evolve and has to be deduced by considering many visitors. Finally, ignoring the tree structure of pages and paths, which are part of the context of the problem, can easily produce misleading results.

The first step in any statistical analysis is to look at the data in innumerable ways. Visualizing the behavior of visitors at a website in a way that respects the structure of the data is difficult, but some progress has been made. For example, Mark Hansen and Wim Sweldens of Bell Labs have introduced *synchronized browsing*, which allows an analyst to browse web pages and simultaneously see the results of analyses in the browser, tying the structure of the analysis to the structure of the website. As the content provider clicks through the website, results of an analysis of visitors to the page, such as where they came from, what they did on the page, how long they stayed, and where they went next, are displayed. This gives the content provider a sense of the structure of the data that can hint at more formal models of the "average" user, the "typical" way the website is used, and the range of likely departures from the typical and average. Thus, the visualization captures the context, the mean, and the noise of the data.

Although there is a massive amount of data in a web log file, we may still be data-poor when the dynamic tree structure of the data is taken into account. There may be many more ways to traverse a website than there are visitors, and the possible paths change continually as pages are added, deleted and edited. Still, if a statistical model can be used to describe a visit, then it is possible to make inferences about paths that have never been observed. Even if a complete path has never been observed, visitors may have used parts of the path. With a statistical model, the data can be used to estimate probabilities of the partial paths which can then be combined to estimate the probability of a complete, unobserved path.

Traditional data mining problems, like clustering pages or users, that computer scientists have focused on are also important, but these can be much richer than often assumed. For example, the goal may be not just to segment users into homogeneous groups but to understand how user segments are changing and how the changes relate to changes in the website. A static measure of performance, such as an overall misclassification rate, that gives one summary of performance can then be seriously wrong about current performance. Generally, good performance measures cannot be developed without close collaboration with subject matter experts.

The fact that it is dangerous to analyze data without knowledge of the context has been underscored by many people involved in analysis of data. But it is worth repeating because ignoring it leads to silliness which can discredit all of statistics and data mining. An April 1998 article in Forbes entitled "Diaper-Beer Syndrome"

gives a long list of failed attempts to build data warehouses for data mining and an equally long list of unrealistic claims, including some that grew out of the often-cited case that found a correlation between buying beer and diapers in one small convenience store in early evening hours. Several of the papers in [13] also emphasize that a set of general tools applied in a vacuum without taking the context into account is likely to lead to useless results (for example, [1] and [7]).

In principle, then, statistics does have a role to play with massive data. But, has statistics had any success in analyzing massive data? The rest of this paper considers that question in the context of fraud detection, a topic that has previously been discussed in the data mining and knowledge discovery literature (e.g., [3] and [12]). More information on fraud detection is given in [4].

3. Statistical Fraud Detection There are many kinds of telecommunications fraud. In calling card fraud, a stolen credit card is used to place a call. In wireless fraud, a cellular phone may be cloned. In wireline fraud, a hacker may break into a university's telecommunications network and route calls over the network until the fraud is detected. In subscription fraud, a customer initiates service without intending to pay. Our goal is to find such fraud as quickly as possible by scoring each call for fraud while it is active or as soon as it has ended and then updating a fraud score for the account by the new call score.

If we could predict the next legitimate call on an account precisely, we could score calls for fraud by comparing the observed call with the predicted call. The new call would be scored zero if it matched the predicted call exactly, and scored one if not, and the cumulative call score would measure the severity of the fraud. Of course, calls cannot be predicted without uncertainty, so the natural alternative is to score calls against a probabilistic or predictive model instead of against an exact prediction. The predictive model has to accommodate the variability in the calling pattern, so calls within the usual range of variability do not falsely trigger a fraud alarm. Consequently, fraud detection is as much about describing customer-specific noise as it is about describing a customer-specific average.

Probabilistic call scoring is not easy. First, there are many, many calls, and scoring must be fast enough to keep up with the data flow. So any score must be simple to compute and any prediction model must be simple to maintain. Second, there can be millions of callers, with wildly diverse calling patterns. Some customers make a few calls a week; others make a few thousand calls a day. Some customers never make calls out of business hours; some never make calls in business hours. Some never call China; some make many calls to China. Scoring must be appropriate for *any* caller, so a prediction model must be maintained for *each* customer. Third, the customer base is often volatile, with new customers making calls each day, so the prediction model must be easy to initialize meaningfully for a caller with no previous history. In statistical terms, we are forced to "borrow strength" from our knowledge of previous customers to initialize the distribution for a new customer quickly. Finally, the prediction model has to adapt as more is learned about the customer, without intervention or off-line processing, but the model should adapt only to legitimate behavior, not fraud. Thus, finding a good algorithm for fraud detection involves much more than training a classifier on legitimate and fraudulent calls or

legitimate and fraudulent account summaries; it involves modeling legitimate and fraudulent calls probabilistically and scoring calls against both models.

From a statistical perspective, a call can be represented by a random vector $\boldsymbol{X} = (X_1, \ldots, X_k)$, where X_1 might be call duration, X_2 call timing (day-of-week or hour-of-day), X_3 call rate, X_4 geography of the called number (hierarchically organized by country, region, city, exchange, for example), and so on until all the information that can be gleaned from a calling record is included. A legitimate caller i at the time of its call n has a multivariate distribution $C_{i,n}$ on $\boldsymbol{X}_{i,n}$. A fraudster has a, hopefully different, multivariate distribution F. The distributions $C_{i,n}$ and F are high-dimensional, with possibly complex interactions among the variables. For example, call duration may depend on the time-of-day, so considering only the one-dimensional, marginal distribution of call duration, ignoring the differences in peak and off-peak hours, would be misleading. In any case, the statistical problem is to score a call according to whether it is more likely to be fraud (have come from F) or to be legitimate (have come from $C_{i,n}$), so the first step is to estimate $C_{i,n}$ for each caller i and the fraud distribution F as well as possible, given the constraints on space and processing time.

An ideal estimator of $C_{i,n}$ or F would be

- **nonparametric**, to capture the full set of behaviors over the customer base,
- **short**, to meet constraints on storage space,
- **sufficient**, so the $C_{i,n}$s and F represent all the data relevant to detecting fraud,
- **multivariate**, to capture important dependencies among calling variables,
- **dynamic**, to enable updating $C_{i,n}$ with every unsuspicious call that the customer makes,
- **easily initialized**, so a reasonable starting estimate can be assigned to a new customer, and
- **"optimal"**, to predict most customers well and a significant fraction very well.

Histograms (an array of bins and corresponding relative frequencies) are simple nonparametric estimates that have the required properties (see [9] for details). Either standard fixed-width histograms with fixed bin endpoints and variable bin heights or fixed-height histograms with variable endpoints and fixed heights (*i.e.*, a set of quantiles) can be used. Choosing the set of histograms to monitor amounts to choosing the set of marginal and conditional distributions to be monitored. For example, should durations of peak and nonpeak calls be monitored separately or together? A complication is that the same set of distributions has to be monitored for all customers to simplify processing.

A statistical approach to designing $C_{i,n}$ is described in [18]. It assumes that there is a set of "legitimate" historical calls for tens or hundreds of thousands

of customers over several months, where legitimate means that the customers experienced no fraud during the period or so little fraud that it was not detected. The set of conditional distributions to track is then chosen by applying χ^2 tests of independence to each customer separately, and combining the resulting p-values across accounts to find the conditioning variables that are needed to model most customers well. Because the procedure is based on statistical testing, it explicitly accommodates uncertainty and limits the structure of the model to that which can be supported by the data. It is also feasible for large databases of customers because each χ^2 test is quick to compute. The resulting structure or set of histograms is called a *signature*. Each customer i at the time of its call n has its own signature $C_{i,n}$.

The next task is to define a way to re-estimate or incrementally update a signature with every new unsuspicious call that the customer makes. Call-by-call updating restricts processing to the subset of accounts that have new calls and avoids the need to retrieve past calls from a database, which is inevitably slow. It also enables call-by-call fraud detection. In the statistics literature, incremental updating is called sequential estimation, and it has a long history. Perhaps the most common sequential estimation method, one that is equally familiar to engineers, is exponentially weighted moving averaging (EWMA). If a standard, fixed-width histogram has estimated bin probabilities $\boldsymbol{p}_{i,n}$ after call n, and call $n+1$ is represented by a vector $\boldsymbol{Z}_{i,n+1}$ that is zero in every bin except the one that contains the observed value for call $n+1$, then the EWMA updated bin probabilities are

$$\boldsymbol{p}_{i,n+1} = (1-w)\boldsymbol{p}_{i,n} + w\boldsymbol{Z}_{i,n+1},$$

where $0 < w < 1$. The weight w controls how fast old calls are aged out of the signature because the current call has weight w and the k^{th} earlier call has weight $w(1-w)^k$. With larger w, the estimate adapts more quickly to changes in a customer's calling pattern, but it is also more variable because the effective sample size or number of calls that have non-negligible weight is smaller. Variants of exponentially weighted moving averages can be used to update fixed depth histograms [8] and top-seller kinds of histograms that try to retain only the bins with the highest frequencies [17].

Sequential estimators require initial values to start, so signatures require initial values; statistics can be used to find them. Our method assigns each new customer to several customer segments, one for each component of the signature, on the basis of information about its first few calls. Each customer segment has an average histogram that is used to initialize the signature component of any customer assigned to that segment. These customer segments partition the space of all possible customers, but there are two major differences between standard database partitioning and our approach. First, we initialize each component of the signature separately, because the signature is a product of its components by design. This gives a huge number of possible customer segments or products of signature components, some of which were not even observed in the training set. The second difference is that each customer can diverge from its initial assignment and move to its own "segment of one" over time because the initial signatures evolve through exponential updating. Details are given in [9].

The statistical problems in fraud detection do not end with algorithm design. Evaluating a fraud detection algorithm is as challenging as designing one. For example, fraud management centers track the fraction of *investigated* accounts that are mislabeled as fraud, while researchers track the fraction of *all* accounts that are mislabeled as fraud. Thus, the researcher has a larger denominator, a much smaller false alarm probability, and a more optimistic view of performance. Yet changing to the fraud analyst view is hard because that requires modeling the process for deciding which suspicious accounts to investigate first. Moreover, because fraud detection is ongoing, performance has to be tracked over time, not just summarized at the end of a test period. There are also many facets to performance, such as the time to detect fraud, the number of false alarms, and the losses from fraud. Simulating a fraud management center by running the proposed algorithms for scoring calls and accounts and prioritizing high scoring accounts on a huge set of calls may be the only way to obtain valid estimates of the dynamic performance of a fraud detection system. But then statistics can again be applied to design the simulation and its evaluation.

4. Final Thoughts Computing is critical to the analysis of massive data, and statisticians have much to learn from computer scientists about computing with massive data. At the same time, statistics puts new demands on computer science. For example, there is a need for database management systems that allow more than rudimentary ways to explore data. Many statisticians manage data by extracting what they need from a database, applying tools like `awk` and `perl` to the selected data to reduce or aggregate the data further, and then analyzing the reduced or partially analyzed data in a statistical environment like S. This works, but it is not convenient and probably far from the state of the art. Scalable algorithms are needed, too, but the ability to explore data and fitted models conveniently is more fundamental.

There are also aspects of data mining in computer science that statisticians should adopt. A statistical model is a principled way to reason about data, but it may not be the best way to start exploring data. Partitioning the data, especially when there is a lot of data, may be better, and it has led to some new powerful techniques of model building, such as boosting ([14] and [15]). Even with procedures like boosting, though, a model helps to understand why the procedure works, and thus where it works best and where it fails. (See, for example, [16] and [2].)

Massive data raise many questions beyond analysis that could be much better addressed by statisticians and computer scientists together than by either group working along. For example, there is growing concern about protecting the confidentiality of the massive data collected in log files on the internet. Confidentiality has been discussed in the statistics literature for decades, but most of the proposals that statisticians have made are not feasible for large sets of data. Data cleaning is another area of common interest.

Statistics has traditionally had strong ties to mathematics, both for designing new methodology and evaluating the performance of methods. Mathematics continues to be important to the analysis of massive sets of data. Probability is fundamental to every step in statistical modeling. Take away probability, and there is

not much left to the approach to fraud detection described in Section 3. Probability also plays a role in computing. It is basic to the design and performance of MCMC (Markov Chain Monte Carlo), for example, which is the workhorse of Bayesian model fitting. Other branches of mathematics that are important to understanding massive data include optimization, Bayesian networks, coding, compression, approximation theory, functional analysis, and algebra. So, just as extracting information is not just computing, it is not just computing together with statistics. Mathematics continues to provide the principles and foundations of much of statistics.

In short, there is much to learn from many disciplines before analysis of massive data will be no harder than analysis of more modest, traditional sets of data. And, it is perfectly clear that the discipline of statistics and statistical thinkers who will have the high standards and wisdom of Jack Hall will be essential for extracting information from massive data.

5. Acknowledgments Special thanks go to John Chambers, Bill Cleveland, Mark Hansen, José Pinheiro, and Don Sun of Bell Labs, Rick Becker of AT&T Labs, and an anonymous referee.

REFERENCES

[1] R. J. Brachman and T. Anand. The process of knowledge discovery in databases. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 37 – 57. AAAI Press, Menlo Park, CA, 1996.

[2] P. Buhlmann and B. Yu. Analyzing bagging. *Annals of Statistics*, 2002. To appear.

[3] P. Burge and P. Shawe-Taylor. Detecting cellular fraud using adaptive prototypes. In T. Fawcett, I. Haimowitz, F. Provost, and S. Stolfo, editors, *AI Approaches to Fraud Detection and Risk Management*. AAAI Press, Menlo Park, CA, 1997.

[4] M. Cahill, F. Chen, D. Lambert, J. C. Pinheiro, and D. X. Sun. Detecting fraud in the real world. In J. Abello, P. Pardalos, and M. Resende, editors, *Handbook of Massive Datasets*, pages 911 – 933. Kluwer Press, New York, 2002.

[5] J. M. Chambers. *Programming with Data*. Springer, New York, 1998.

[6] J. M. Chambers. Users, programmers and statistical software. *Journal of Computational and Graphical Statistics*, 9:404 – 422, 2000.

[7] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153 – 180. AAAI Press, Menlo Park, CA, 1996.

[8] F. Chen, D. Lambert, and J. C. Pinheiro. Incremental quantile estimation for massive tracking. In J. Abello, P.M. Pardalos, and M.G. Resende, editors, *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2000.

[9] F. Chen, D. Lambert, J. C. Pinheiro, and D. X. Sun. Reducing transaction databases, without lagging behind the data or losing information. Technical report, Bell Labs, Lucent Technologies, 2000.

[10] W. S. Cleveland. *The Elements of Graphing Data*. Hobart Press, Summit, NJ, 1985.

[11] W. S. Cleveland. *Visualizing Data*. Hobart Press, Summit NJ, 1993.

[12] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1:291–316, 1997.

[13] U. M. Fayyad, G. Piatstsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA, 1996.

[14] Y. Freund. Boosting a weak learning algorithm. *Information and Computation*, 121:256 – 285, 1995.

[15] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In L. Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148 – 156. Morgan Kauffman, San Francisco, 1995.

[16] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:333 – 407, 2000.

[17] P. B. Gibbons, Y. E. Ioannidis, and V. Poosala. Fast incremental maintenance of approximate histograms. In M. Jarke, editor, *Proceedings of the Twenty-third International Conference on Very Large Data Bases*, pages 466–475, 1997.

[18] D. Lambert and J. C. Pinheiro. Mining a stream of transactions for customer patterns. In *Proceedings of the ACM SIGKDD2001*, 2001.

[19] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, New York, 1989.

[20] D. F. Swayne, D. Cook, and A. Buja. Xgobi: interactive dynamic data visualization in the x window system. *Journal of Computational and Graphical Statistics*, 7:113 – 130, 1998.

[21] D. Temple Lang. The Omega project: new possibilities for statistical software. *Journal of Computational and Graphical Statistics*, 9:423 – 451, 2000.

[22] L. Tierney. *LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics.* J. Wiley and Sons, New York, 1991.

[23] E.R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, 1983.

[24] G. Wiederhold. On the barriers and future of knowledge discovery. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, 1996.

600 Mountain Avenue
Murray Hill, NJ 07974
dl@bell-labs.com