Stat

The ISI's Journal for the Rapid Dissemination of Statistics Research

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

Large complex data: divide and recombine (D&R) with RHIPE

Saptarshi Guha[a], Ryan Hafen[b], Jeremiah Rounds[c], Jin Xia[d], Jianfu Li[c], Bowei Xi[c] and William S. Cleveland[c,*]

Received 3 August 2012; Accepted 26 August 2012

D&R is a new statistical approach to the analysis of large complex data. The data are divided into subsets. Computationally, each subset is a small dataset. Analytic methods are applied to each of the subsets, and the outputs of each method are recombined to form a result for the entire data. Computations can be run in parallel with no communication among them, making them embarrassingly parallel, the simplest possible parallel processing. Using D&R, a data analyst can apply almost any statistical or visualization method to large complex data. Direct application of most analytic methods to the entire data is either infeasible, or impractical. D&R enables deep analysis: comprehensive analysis, including visualization of the detailed data, that minimizes the risk of losing important information. One of our D&R research thrusts uses statistics to develop "best" division and recombination procedures for analytic methods. Another is a D&R computational environment that has two widely used components, R and Hadoop, and our RHIPE merger of them. Hadoop is a distributed database and parallel compute engine that executes the embarrassingly parallel D&R computations across a cluster. RHIPE allows analysis wholly from within R, making programming with the data very efficient. Copyright © 2012 John Wiley & Sons, Ltd.

Keywords: big data; computer cluster; data visualization; deep analysis; Hadoop; mathematical statistics; statistical computing

1 Introduction

There are two goals in our development of the divide and recombine (D&R) approach to the analysis of large complex data. The first is *deep analysis:* comprehensive analysis, including visualization of the detailed data at their finest granularity, which minimizes the risk of losing important information in the data. The second is a D&R computational environment where an analyst programs exclusively with an interactive language for data analysis (ILDA), making programming with the data very efficient.

We have developed D&R in conjunction with our own analyses of large complex data, especially Internet traffic datasets (Hafen et al., 2009; Xi et al., 2010; Guha et al., 2011; Anderson et al., 2012). We have succeeded in the above two goals. Deep analysis of unprecedentedly large datasets has achieved subject matter firsts: the first validated model for voice over the Internet (VoIP) traffic, allowing simulation study to size VoIP networks; the first algorithm for detecting keystrokes in any Internet connection, a powerful tool for detection of interactive connections where they

[a]**The Mozilla Corporation, San Francisco, CA 94306, USA**
[b]**Pacific Northwest National Labs, Richland, WA 99354, USA**
[c]**Department of Statistics, Purdue University, West Lafayette, IN 47907, USA**
[d]**GE Global Research, Niskayuna, NY 12309, USA**
***Email: wsc@purdue.edu**

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

The ISI's Journal for the Rapid Dissemination of Statistics Research

do not belong; and the first simple, validated model for best-effort Internet traffic that provides a new mathematical foundation for the traffic, and very fast generation for simulation studies for network engineering.

The basics of D&R were first described in Guha et al. (2009), which focuses on visualization, and in Xi et al. (2010), the VoIP analysis mentioned above. The initial breakthrough in our D&R computational environment was RHIPE, first developed by Saptarshi Guha in his PhD thesis at Purdue in the Department of Statistics (Guha, 2010). This merger of R and the Hadoop distributed file system and parallel compute engine allows D&R to be carried out wholly from within the R ILDA. There is now a RHIPE development group working on the GitHub social coding site (RHIPE, 2011a), an R package on GitHub, and a Google discussion group (RHIPE, 2011b).

1.1. D&R basics

Initially, the data analyst divides the data into subsets and writes them to disk. Then each of a collection of analytic methods is applied to each subset. The analytics are *statistical methods* whose output is categorical or numeric, and *visualization methods* whose output is visual. An analytic method is applied to a subset independently, which means without communication with other subset computations. The outputs from the method are then recombined. Thus the computations have three parts: the data are divided into *subsets* by *S computations*; an analytic method is applied to subsets by *W computations* where each is strictly *within* a subset; outputs from W are recombined by *B computations* which are *between* subsets.

Consider a linear regression with the model $Y = X\beta + \epsilon$. Y is $n \times 1$, the dependent variable; X is $n \times p$, p explanatory variables; ϵ is $n \times 1$ with elements that are i.i.d. normal with mean 0; and β is $p \times 1$, the regression coefficients. Suppose $n = rm$ for integer r and m. For $s = 1, \ldots, r$, subset s is X_s $(m \times p)$ and Y_s $(m \times 1)$, where X_s is m rows of X, and Y_s is the corresponding rows of Y. The notational convention for the subsets is

$$X = row_{s=1}^{r}[X_s], \qquad Y = row_{s=1}^{r}[Y_s],$$

where *row* is matrix row concatenation. These subsets are the result of the S computations. For subset s, the optimal least-squares estimate is

$$\dot{\beta}_s = (X_s'X_s)^{-1}X_s'Y_s. \tag{1}$$

These estimates are the results of the W computations. Suppose we recombine by taking the vector mean across subsets s, that is, the means of the elements of $\dot{\beta}_s$. The D&R estimate, the result of the B computations, is

$$\ddot{\beta} = \frac{1}{r}\sum_{s=1}^{r}\dot{\beta}_s = \frac{1}{r}\sum_{s=1}^{r}(X_s'X_s)^{-1}X_s'Y_s. \tag{2}$$

A division is a data structure. For large data and for small data, the choice of the data structure is a major factor in the efficiency of the programming with the data. Analyses of large complex data often consist of a number of different types of analysis that call for different data structures, and therefore different divisions of the entire dataset. The application of analytic methods to the subsets of a single division is a *division analysis thread*. A thread typically has sub-threads. For example, it can be an analysis of just a certain part of each subset, or a further analysis of outputs of W or B computations. The D&R framework for an analysis thread is illustrated in Figure 1. The three classes of recombination procedures are defined in Section 3. Our description and illustration of D&R conveys the fundamental concept, but in practice the S-W-B computations typically interact in more complex ways.

1.2. Exploiting the D&R approach

Consider applying a statistical method directly to an entire large complex dataset. Except for the small number of cases where there is a fast parallel algorithm for the methods, the computation is either infeasible, or impractical

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

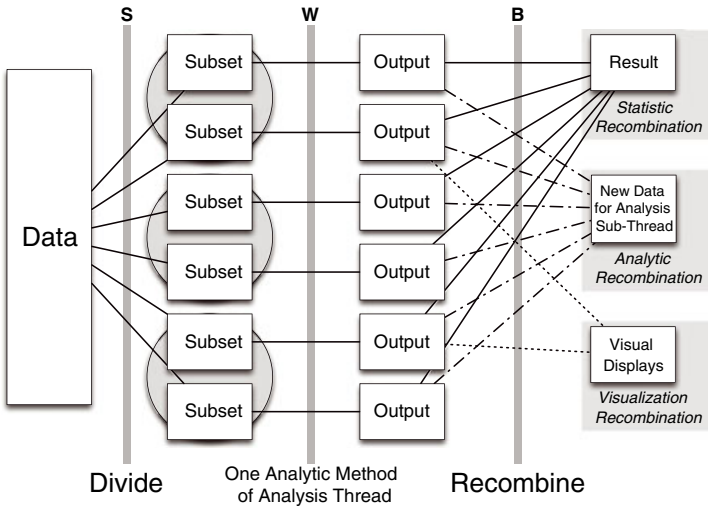

Figure 1. The D&R Statistical and Computational Framework. D&R parallelizes the data. S, W, and B computations carry out D&R. S divides into subsets, W applies an analytic method to each subset, and B applies analytic methods to the outputs of W to achieve recombination. D&R succeeds in achieving deep analysis of complex data because the intensive S, W, and B computations are embarrassingly parallel. From within R, an analyst issues RHIPE R commands to run these computations using Hadoop. Almost all S and W computations, and many B computations, are executed in this way. However, many B computations operate on small data and are done using the standard interactive R computing in the R global environment.

because it takes so long. D&R solves this problem. Intensive parts of the S-B-W computations can run in parallel. In addition, they are the simplest possible parallel computations because there is no communication between them. These *embarrassingly parallel* computations become feasible and run fast on a distributed computational environment designed for them. B computations run across subsets, but certain aspects of them are embarrassingly parallel. Of course, the D&R result for a statistical method is typically not the same as the hypothetical direct all-data result. A major D&R research thrust, described below, seeks "best" D&R procedures, a new framework for research based on statistical thinking and theory.

Consider applying a visualization method to a large complex dataset. Deep analysis requires that we visualize not just summary statistics of the data, but also the detailed data at their finest granularity. D&R enables this too, but by a different mechanism: statistical sampling. Subsets contain detailed data. In D&R, the analyst applies a visualization method to each of a number of subsets, typically not to all because there are far too many to view. Instead, statistical sampling is used to select a limited number. This rigorous form of data reduction uses the same statistical thinking behind survey sampling, but in D&R there is an advantage because all of the data are in hand, which can be exploited in developing a sampling plan.

1.3. Two research thrusts

D&R makes computation feasible, and has some striking validation in practice. Still, for D&R to work statistically as well as possible in practice, much research is needed. Here, we provide an overview of our accomplishments so far. Many new terms must be defined; the first instance of each is put in italics. There are two research thrusts.

The first thrust uses statistical thinking and theory to develop "best" division and recombination procedures for analytic methods. The two procedures used for a method are critical to its success. This research is a new setting for statistical thinking and theory. For both statistical and visualization methods, we have developed general division and

recombination procedures that are described in Section 3, and that we have used in practice. In Section 5 we illustrate the use of statistical theory to study D&R procedures.

The second thrust is our D&R computational environment, which is made up of the R interactive language for data analysis, the Hadoop distributed file system and parallel compute engine, and our RHIPE merger of R and Hadoop. RHIPE allows an analyst to work wholly from within R, making programming with the data very efficient. Sections 4 and 6 describe RHIPE and Hadoop. Development of the environment requires that we run designed experiments to study the many factors that affect performance. This is illustrated in Section 4.

2 D&R illustrated: Internet packet-level traffic

D&R procedures and the D&R computational environment will be illustrated here by our analyses of large complex packet-level Internet traffic data for research in network performance and cyber security (Xi et al., 2010; Guha et al., 2011; Anderson et al., 2012). Each Internet communication is a connection between two hosts (computers). An example is a client host downloading a Web page from a server host. Information sent from one host to the other is broken up into packets of 1460 bytes or less, sent across the Internet, and reassembled at the destination. Attached to each packet are headers that manage the connection and other network processes.

We collect packet-level data in both directions on an Internet link: arrival times and header fields. Each collection has from hundreds of thousands to billions of connections. The number of packets per connection varies from a few to the low hundreds of thousands. In each investigation, typically one analysis thread is modeling by connection. So the division in this case is by connection. Each connection subset is a table. The columns are variables, which are the timestamp and header fields. The rows are the packets, so the number of rows varies by connection.

We focus on a smaller collection, but still in the category of large complex data (Xi et al., 2010). Voice over the Internet (VoIP) was studied. We collected data on the Global Crossing core VoIP network. The hosts were 27 gateways to the network. Each connection is a call, with two directions, each a semi-call. Because we model each direction independently, the division is by semi-call, so each subset is the packet data of one semi-call. The data consist of 1.237 billion packets for 277,540 semi-calls.

Each semi-call has an alternating sequence of transmission intervals with voice packets, and silence intervals with no voice packets. A silence notification packet starts a silence interval and a packet containing voice ends it, which allows the start and end of each interval to be identified. Each semi-call is a stochastic process of alternating intervals. Modeling the process was a critical task in the overall modeling of semi-calls.

3 Division and recombination procedures

3.1. Conditioning-variable division

Conditioning-variable division is a class of division procedures that depend on the subject matter under investigation. Certain variables are selected as *conditioning variables*, and the data are divided into subsets by conditioning on values of them. The conditioning variables become *between subset-variables (BSVs)* with one value per subset. Other variables, *within-subset variables (WSVs)*, vary within each subset. Analyses reveal the relationship of the WSVs in each subset, and how it changes with the BSVs. Raw BSVs and WSVs are those collected directly, and typically other WSVs and BSVs are derived from them. Table I shows the raw BSVs and WSVs for the VoIP semi-call data that come directly from the header fields and timestamp, and the BSVs and WSVs derived from them. A semi-call ID is

The ISI's Journal for the Rapid
Dissemination of Statistics Research

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

Table I. Each semi-call subset for the VoIP packet-level data consist of values of BSVs (between-subset variables) and WSVs (within-subset variables). Raw variables are those that come directly from the measurement process. Derived from them are other variables important to the analysis.

Source	Variable	Type
Raw	1. semi-call identifier	BSV
Raw	2. semi-call source gateway	BSV
Raw	3. semi-call destination gateway	BSV
Raw	4. packet arrival time	WSV
Raw	5. packet flag: voice or silence start	WSV
Derived	6. transmission interval lengths, from 4 and 5	WSV
Derived	7. silence interval lengths, from 4 and 5	WSV
Derived	8. number of transmission intervals, from 6	BSV
Derived	9. number of silence intervals, from 6	BSV
Derived	10. link transmission direction, from 2	BSV
Derived	11. semi-call start time, from 4	BSV
Derived	12. semi-call duration, from 4	BSV
Derived	13. number of packets, from 4	BSV

determined by 4 values from the headers of each packet: source gateway, source port number, destination gateway, destination port number. Formally, the four fields are BSVs, but for the analysis, beyond simply serving as an ID, the two port numbers are not relevant to the analysis.

Conditioning-variable division is not new and is already widely used because it is a powerful mechanism for analysis of data of any size. It is the basis of the trellis display framework for visualization (Becker et al., 1996; Pinheiro & Bates, 2000; Fuentes et al., 2011), which has been implemented in R by the lattice graphics package (Sarkar, 2008).

3.2. Replicate division

Another division class of D&R is *replicate division*. It arises in many different situations. One is when subsets are still too large after conditioning-variable division. For replicate division, the data are n observations of q variables. They are seen as *replicates*, all coming from the same experiment under the same conditions. *Random-replicate (RR) division* uses random sampling of observations without replacement to create subsets. This is attractive because it is computationally fast. But it makes no effort to create subsets each of which is representative of the dataset. *Near-exact-replicate (NER) division* makes the effort. The n observations are broken up into local neighborhoods with approximately the same number of observations; a replicate subset is formed by choosing one point from each neighborhood. Of course, the computation to do this is itself a challenge.

3.3. Statistic recombination

The output of a statistical method applied to each subset is numeric and categorical values. Any function of the data is a statistic, so the recombination of the outputs of a statistical method is *statistic recombination*. Very often the statistic is an estimate of an estimand. In this case, the division procedure and the statistic recombination procedure together become a D&R estimator whose statistical properties we can study. An example is the D&R estimator $\ddot{\beta}$ of the coefficients in the linear regression of Section 1. The statistic recombination is the vector mean of the subset

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

The ISI's Journal for the Rapid
Dissemination of Statistics Research

least-squares estimates $\dot{\beta}_s$. We can pair this with random replicate division to form an estimator, or with near-exact replicate division to form another, and use statistical theory to compare performance. Such a process is used to find "best" D&R estimators. Section 5 illustrates this.

3.4. Analytic recombination

Analytic recombination is simply a continued analysis of the outputs of a W or B computation. This is quite common, and often, when the outputs are a substantial data reduction, they can be treated as small data. Analysis sub-threads with small data are an integral part of D&R; this important matter is discussed further in Sections 4 and 6.

For the modeling of the alternating transmission-silence interval length process, we applied many statistical and visualization methods to just the lengths, which become the detailed data of the alternating process. We concluded the following for each semi-call: the alternating process consists of mutually independent lengths; the transmission lengths are identically distributed; the silence lengths are identically distributed; the square roots of the lengths for each of these two sets of intervals have a marginal distribution that is very well approximated by a gamma distribution; and the two sets of lengths have different shape and scale parameters.

For each semi-call we estimated the shape and scale parameters for silence and for transmission. In addition, for each estimation of the two parameters, we computed 1000 bootstrap samples. Visualization methods applied to the bootstrap samples showed that the bootstrap distribution of the logs of the parameters were very well approximated by a bivariate normal distribution. As a result, the log shape and the log scale became the parameters of the square-root gamma models. We used the bootstrap to estimate the variances and the covariance of the two estimates of the log parameters for each fit. The final output of the analysis for each semi-call is a 12-tuple made up of two 6-tuples, one for silence and one for transmission. Each 6-tuple has 2 log parameter estimates, 2 estimates of their variances, an estimate of their covariance, and the number of lengths.

The 6-tuple for silence is a summary statistic for the silence lengths of a semi-call; a similar statement holds for the transmission 6-tuple. The summary statistics are a validated data reduction, and are small data. They were further analyzed across subsets using a number of statistical and visualization methods, an analytic recombination. One possible model was that the two parameters, log shape and log scale, are the same across all semi-calls for each interval type, a fixed-effects model. A second was that they needed to be modeled by a distribution, a random effects model. We found the latter was needed. For each length type, the 2log parameters were modeled as a bivariate normal; the bivariate normal parameters were found to be different for the two types. So the overall model for the interval length alternating process is hierarchical.

3.5. Visualization recombination

Visualization recombination provides a mechanism for rigorous visualization of the detailed data at their finest granularity (Guha et al., 2009). Subsets contain the detailed data, so we choose a visualization method and apply it to subsets. Section 1 describes the use of statistical sampling to select the visualized subsets. The sampling plan uses BSVs, which makes the process rigorous. Application of the method starts with a statistical W computation on each sampled subset, resulting in numeric and categorical output that are shown on a plot. The visualization recombination is a display design that combines all subset plots.

We have used three general sampling procedures: *representative*, *focused*, and *cognostic*. A representative sample is chosen to cover the joint region of values of a set of BSVs. A focused sample explores a particular sub-region of interest. Cognostics is a general notion of Tukey (Tukey, 1983) that we have tailored to D&R. BSVs are developed that search for certain kinds of statistical behavior in a subset. One application is to find subsets that deviate from a consistent pattern seen in visualization for a representative sample.

The ISI's Journal for the Rapid
Dissemination of Statistics Research

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

In the quantile plot visualization method, empirical quantiles are plotted against those of a mathematical distribution like the normal or gamma (Cleveland, 1993). It and other visualization methods were used in the modeling of the VoIP silence-transmission alternating process. We found the gamma was a good approximation of the marginal of the square root lengths for transmission and for silence, and the bivariate normal was a good approximation of the bootstrap distribution of estimates. An important factor for this model checking was the number of intervals, so it was used as a BSV for the sampling. There was a possibility that the marginal of the lengths changed with the number. The quantile plots showed it did not. Theory suggests that the bivariate bootstrap distribution tends to normal with the number. Quantile plots and other display methods showed this is the case, but it gets there much faster for the logs of the parameters; in fact, the approximation for the logs was adequate even for the smallest numbers. This led to modeling log shape and log scale. Representative sampling by semi-call was used for the visualization sampling. The BSV was the total number of lengths in a semi-call. The sample size was 2000 semi-calls, and the values of the BSV were chosen to range from the smallest number for the semi-calls to the largest, and to be as nearly equally spaced on a log scale as possible. This resulted in 2000 sets of silence intervals and 2000 sets of transmission intervals, meaning 4000 quantile plots for each application of a method.

4 A D&R computational environment

Our D&R computational environment has three open-source software components: R, RHIPE, and Hadoop. The ILDA R (Hornik, 2011) is widely used and highly acclaimed. Its parent, the S language, won the 1999 ACM Software System Award, by far the most prestigious software award. Hadoop, which consists of the the Hadoop Distributed File System (HDFS) and the Map-Reduce distributed compute engine, executes the embarrassingly parallel computing of the S, W, and B computations (White, 2011). Hadoop was inspired by papers written about Google's MapReduce and Google File System (GFS). David Cutting at Yahoo developed the first Hadoop, employing the MapReduce framework with the HDFS. In 2008, the project was passed off to the Apache Software Foundation for worldwide support and distribution. RHIPE, the R and Hadoop Integrated Programming Environment, is our merger of R and Hadoop (Guha, 2010). It means "in a moment" in Greek and is pronounced "hree pay'." Integration of R and Hadoop is accomplished by a set of components written in R and Java. RHIPE communicates with Hadoop to carry out the parallel S, W, and B computations. The analyst does not have to program Hadoop, or attend to details of parallel processing. The analyst simply uses the small number of fundamental computational modes of Hadoop. Two of the modes are discussed in Section 6.

4.1. RHIPE

The data analyst writes R code for S computations that divide the data into subsets, and that create R objects containing the subsets, usually one object per subset. The code is an input to RHIPE R commands that communicate with Hadoop. The subset R objects are distributed by Hadoop across the nodes of the cluster in the HDFS. For the regression example of Section 1, each R object contains one pair, X_s and Y_s. Then the analyst gives R code to RHIPE for W computations that apply an analytic method to each subset, and that create R objects containing the outputs of the method applications. For the regression example, the outputs of the W computations are the r subset regression coefficients $\dot{\beta}_s$. The W outputs are B inputs. The analyst gives R code to RHIPE for B computations that recombine the B inputs, and create R objects containing the B outputs. For the regression example, the output is is a single R object, the vector mean estimate $\ddot{\beta}$.

For the RHIPE-Hadoop computation framework, W computations by their very specification are embarrassingly parallel. Much of the tasking of the S computations are as well. While B computations might appear not to be, certain aspects described in Section 6 are also embarrassingly parallel.

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

The ISI's Journal for the Rapid Dissemination of Statistics Research

RHIPE R commands can have Hadoop write outputs of S, W, or B computations to the HDFS. S output objects are always written because they create division subsets which persist across an analysis thread. B outputs are almost always written to the HDFS because they tend to be either a final answer for a method, or data that are further analyzed to get a final answer. W computations are sometimes written, but are typically not when they are just the means to the recombination end. Whether written or not, the B and W computations can be run simultaneously with W outputs passed to B in memory.

Embarrassingly parallel computations of S, W, or B that are run by Hadoop consist of the same R code being applied to each object in a collection of objects. Hadoop assigns a core to compute on an object. There are typically far more objects than cores. When a core finishes its computation on an object, Hadoop assigns it to a new object. To minimize overall elapsed read/write time when objects are read from the HDFS, the Hadoop scheduling algorithm seeks to assign a core of a node as close as possible to the node on which an object is stored. In other words, Hadoop brings the core to the data, rather than the other way around.

4.2. D&R computation: From elephants to mice

Our R-RHIPE-Hadoop environment has two sets of servers. The RR cluster runs R and RHIPE. The RH cluster runs RHIPE and Hadoop. The analyst logs in to RR, and carries out a traditional R interactive session. RHIPE commands are issued here, but the ensuing S-W-B executions are on RH. Output to be saved is written to the HDFS on RH. These big RH computations are "elephants".

The elephants make D&R deep analysis feasible. However, for a large complex dataset much analysis also occurs on RR in the R global environment, which means computing in the classical R interactive mode. Some of these R commands execute virtually instantaneously, and need an instantaneous response. These are "mice". Analysis in R occurs when outputs of W or B computations are collectively a small dataset to be further studied in an analysis subthread. In our analyses, visualization recombination is in R because sampled subsets are collectively a small dataset. Outputs, even when they come from all subsets, can consist of summary statistics that are collectively also a small dataset. We have seen one example in Section 3 in the modeling of the transmission-silence alternating process for the VoIP data. Analyzing data on RR, or more generally using any ILDA, is a critical part of D&R analysis. We separate the servers running the small-data computations and the servers running the big RHIPE-Hadoop computations to keep the elephants from trampling the mice.

4.3. Designed experiments for optimizing R-RHIPE-Hadoop performance

Factors from a spectrum of sources affect the performance of each distributed R-RHIPE-Hadoop job. At one end are user-specified factors: subset size; number of subsets; and properties of the R commands. At the other end are cluster hardware factors. In between are a large number of RHIPE and Hadoop configuration parameters. The response is the elapsed time of RHIPE R code. The system is very complex, and empirical study through designed experiments is needed to understand the dependence of the response on the factors. While systems knowledge is needed for experimentation, this knowledge by itself cannot provide answers.

One small experiment illustrates performance study. The topic is the effect of the number of subsets on the elapsed time of the R function `glm.fit` to carry out linear logistic regression. We use the same notation as that of the linear regression in Section 1. X is the $n \times p$ matrix of values of p independent variables, and Y is the $n \times 1$ matrix of the response values, in this case 0-1. There are r subsets each of size m, so $n = rm$.

Y and X were generated and $n = 2^{30}$. There were two factors, p and $\log_2(r)$ where $\log_2$ is log base 2. For a first set of runs, $p = 15$ and $\log_2(r)$ took the values 13 to 23 in steps of 1. The corresponding sequence of log subset sizes, $\log_2(m)$, was 17 to 7 in steps of 1. The number of numeric values was 2^{34}, and at 8 bytes per double-precision value

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

in memory, the data size was 2^{37} bytes, about 128 gigabytes. For the second set of runs $p = 127$ and $\log_2(r)$ took the values 18 to 22 in steps of 1. The corresponding sequence for $\log_2(m)$ varied from 12 to 8 in steps of 1. The data size for this set of runs was 2^{40}, about 1 terabyte. For both sets of runs there were three replicates for each combination of p and $\log_2(r)$. Two types of elapsed times were measured for each division. The first was *Method (M)*: in the W computation, applying the R function `glm.fit` for linear logistic regression to the subsets, plus in the B computation, computing the vector mean of the regression coefficients across subsets. The second was *Read-Write (R/W)*: in the W computation, reading the subsets from the HDFS, plus in the B computation, writing the coefficient estimate vector to the HDFS. The cluster on which the timings were run had 11 servers, each with 24 processors, 48 gigabytes of memory, and 8 terabytes of disk. The components of the cluster had average speeds, neither especially fast or slow.

Each panel of Figure 2 has results for one of the three timings: M, R/W, or M + R/W. Log base 2 elapsed time is plotted against $\log_2(r)$. Each plotted point, ∘ or +, is one combination of p and $\log_2(r)$. The elapsed time value plotted is the log of the mean of the three replicates for the combination. The ∘'s are $p = 15$, and the +'s are $p = 127$. Each curve is a loess fit with span = 1 and degree = 2 (Cleveland & Devlin, 1988).

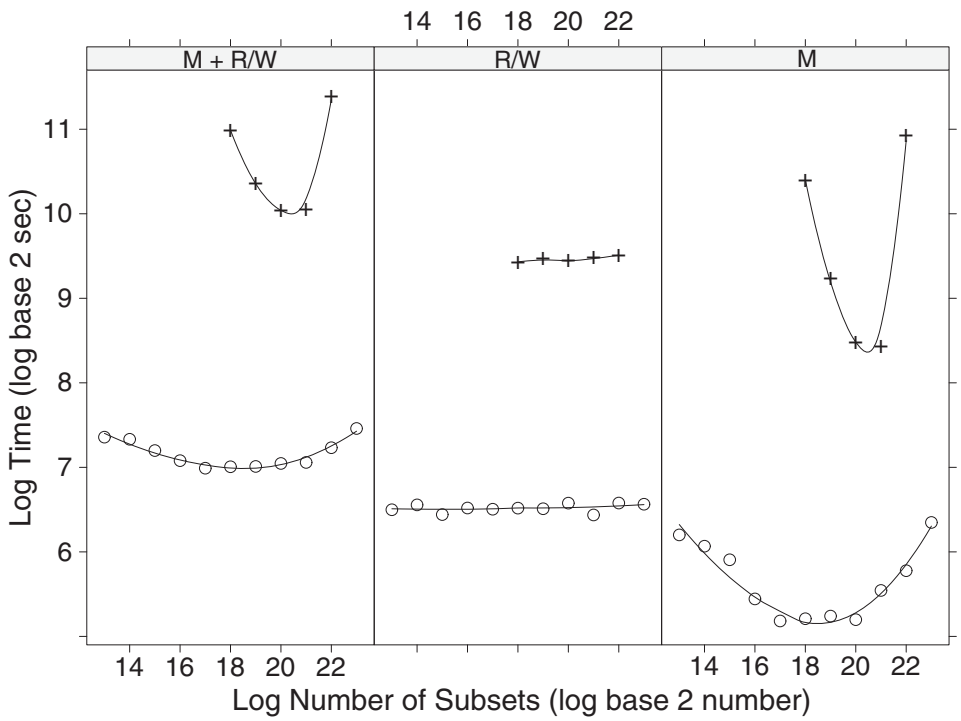

Figure 2. Timings of D&R Estimation of Linear Logistic Regression Coefficients for 128 Gigabyte and 1 Terabyte Datasets. The `glm.fit` program in R was run in the D&R R-RHIPE-Hadoop environment on a cluster with 24 processors, 48 GB of memory, and 8 TB of disk. Two factors were varied. One was p, the number of independent variables. There were 2 values, 15 and 127. The second was r, the number of subsets, which varied for each p. There were 2^{30} observations of the $p + 1$ variables: response and explanatory. The total memory sizes of the two datasets are 128 gigabytes and 1 terabyte. In each panel, log elapsed time is plotted against log r for one type of elapsed time. R/W is reading the subsets from the HDFS plus writing the D&R estimates of coefficients to the HDFS. M is execution of `glm.fit` plus computing the D&R estimate. M + R/W is their sum. ∘'s are times for $p = 15$, and +'s are for $p = 127$. Each elapsed time for one combination of p and r is a mean of three replicate runs. The number of subsets is a major factor for elapsed time. Computation time on this cluster is excellent, and not an impediment to deep analysis.

There is no consistent effect of $\log_2(r)$ on R/W time for each p. This makes sense because the total amount of data read and written does not change with r. Furthermore, the R/W time for $p = 127$ is about 8 times that for $p = 15$ because the data size of the first is 8 times that for the second. There is a substantial effect of $\log_2(r)$ on M time for both values of p, a decrease to a minimum and then an increase. For $p = 15$, the minimum occurs at values of $\log_2(r)$ equal to 18–19; the minimum M + R/W elapsed time at these values is about 128 sec = 2.1 min. For $p = 127$, the minimum is at values of $\log_2(r)$ equal to 20–21; the minimum M + R/W elapsed time at these values is about 1056 sec = 17.6 min. Again, the second is about 8 times that of the first.

The decrease and increase of the M time with $\log_2(r)$ has two causes. For smaller values of $\log_2(r)$, subset object sizes $\log_2(m)$ are larger, so the requests for memory are larger, and can cause a bottleneck. For larger values of $\log_2(r)$, there are more subsets, and Hadoop overhead in handling them goes up and can cause a bottleneck. The values of $\log_2(r)$ where the minima occur balance the two causes.

For the cluster measured, the best elapsed times are excellent, and would not interfere with deep analysis. However, the hardware specifications are a major factor for elapsed time. The disk, bus, and controller speeds have a major impact on R/W time. For the above experiment, R/W is about 70% of M + R/W at the minima. Increasing the above speeds substantially would reduce the elapsed time substantially.

5 Statistical theory for D&R estimation recombination

The D&R estimator $\ddot{\beta}$ from Equation 2 in Section 1 is

$$\ddot{\beta} = \frac{1}{r}\sum_{s=1}^{r}(X_s'X_s)^{-1}X_s'Y_s.$$

The direct entire-data least-squares estimator of β can be written as

$$\hat{\beta} = \left(\sum_{s=1}^{r}X_s'X_s\right)^{-1}\sum_{s=1}^{r}X_s'Y_s. \tag{3}$$

For the normal model, $\hat{\beta}$ is optimal, a maximum of statistical accuracy. Except for certain special cases, $\ddot{\beta} \neq \hat{\beta}$.

The accuracy of a D&R estimator from a statistic recombination or a statistical method depends on the statistical division and recombination procedures that are used. We use statistical thinking and theory to develop procedures that lead to "best" estimators. As part of this, we can let the size of the dataset increase and see how results change with the size. However, this needs to be done in keeping with the conceptual framework of D&R to make it relevant. For D&R to work, subset sizes need to be limited, so we fix subset size and let the number of subsets increase. Of course, there has been a lot of work on combining data for estimation. One example is meta-analysis, a name first used by Glass (1976). Methods developed in this and other such areas might be useful for D&R.

One of the few cases where a statistical method has an embarrassingly parallel algorithm for direct application to the entire data is this least-squares estimation for the linear regression model. The terms $X_s'X_s$ and $X_s'Y_s$ in Equation 3 are computed in parallel across a cluster, and the recombination sums them and then computes the algebraic expression of the equation. The algorithm can be implemented very readily in the R-RHIPE-Hadoop computational environment. However, theoretical study of D&R estimation for the regression model is of great interest because the mathematics is

The ISI's Journal for the Rapid
Dissemination of Statistics Research

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

tractable, making it an instructive leading case for intuition about other methods and models for response-explanatory analysis that do not have such direct entire-data algorithms.

We illustrate study of D&R estimation for a very simple case of the regression model. We take $p = 1$, so there is just one regression coefficient without intercept. X in this case is an $n \times 1$ matrix. We take the variance of the error terms to be 1. Suppose $m > 2$ and $r > 1$, the first to keep certain variances finite, and the second to not include the direct entire-data case as part of D&R. The division procedure will be random replicate, and the statistic recombination will be the mean of the subset estimates, the above $\ddot{\beta}$.

The variances of $\hat{\beta}$, $\dot{\beta}_s$, and $\ddot{\beta}$ are

$$\sigma^2(\hat{\beta}) = (X'X)^{-1}, \quad \sigma^2(\dot{\beta}_s) = (X_s'X_s)^{-1}, \quad \text{and} \quad \sigma^2(\ddot{\beta}) = \frac{1}{r^2}\sum_{s=1}^{r}(X_s'X_s)^{-1}.$$

From the theory of linear regression with normal i.i.d. errors, $\sigma^2(\hat{\beta}) < \sigma^2(\ddot{\beta})$ except for special cases of X. We study how much less.

In any specific application of the regression model, X is fixed and known, but to investigate relative sizes of the variances we take X to be random and look at the distributions of the variances. Suppose the elements of X are i.i.d. normal with mean 0 and variance 1. Since the division procedure is random-replicate, r subsets each of size m are chosen randomly. Thus $\sigma^2(\dot{\beta}_s)$ for $s = 1, \ldots, r$ are i.i.d. ratio of the variances given X. Consider the

$$\rho = \frac{\sigma^2(\ddot{\beta})}{\sigma^2(\hat{\beta})} = \frac{1}{r^2}\sum_{s=1}^{r}\frac{X'X}{X_s'X_s} = \frac{1}{r^2}\sum_{s=1}^{r}\left(1 + \sum_{t\neq s}\frac{X_t'X_t}{X_s'X_s}\right) = \frac{1}{r^2}\sum_{s=1}^{r}\left(1 + \sum_{t\neq s}\frac{X_t'X_t/m}{X_s'X_s/m}\right).$$

Now, unconditional on X, each of the $r-1$ terms in the last sum over $t \neq s$ in this equation has an F-distribution with degrees of freedom m and m, so the expected value of each term is $m/(m-2)$, which means

$$\mu = \mathrm{E}\left(\frac{\sigma^2(\ddot{\beta})}{\sigma^2(\hat{\beta})}\right) = \frac{m}{m-2} - \frac{2}{r(m-2)} = \frac{n-2}{n-2r}.$$

A simple derivation shows

$$\tau = \mathrm{Var}\left(\frac{\sigma^2(\ddot{\beta})}{\sigma^2(\hat{\beta})}\right) = \frac{8(m-2r^{-1})(1-r^{-1})}{r(m-2)^2(m-4)} = \frac{8(n-2)(r-1)}{(n-2r)^2(n-4r)}.$$

This bodes well for random-replicate division. As r increases for fixed m, μ increases monotonically to $m/(m-2)$, and τ decreases monotonically to 0, both to order r^{-1}. In addition, an increase in m for fixed r, up to its limit, increases the mean further toward 1, and the variance toward 0.

We could get exact numerical information about the distribution of ρ, but a simple bound gives convincing results. From Cantelli's inequality, $\Pr(\rho \geq \mu + c\sqrt{\tau}) \leq (1+c^2)^{-1}$. We take $c = \sqrt{99}$ so that $(1+c^2)^{-1} = 0.01$. For each fixed n, we find the r for which

$$\mu + c\sqrt{\tau} = 1.01, \tag{4}$$

not restricting r to be an integer. For this r, the probability of the D&R variance being more than 1% greater than the variance of the entire-data least-squares estimate is 0.01. (We can tighten the argument by taking n to be certain values, say powers of 2, and round r down to the nearest power of 2. Decreasing r for fixed n decreases μ toward 1 and τ toward 0. The resulting $\mu + c\sqrt{\tau}$ is smaller than 1.01. Doing this yields the same conclusion.)

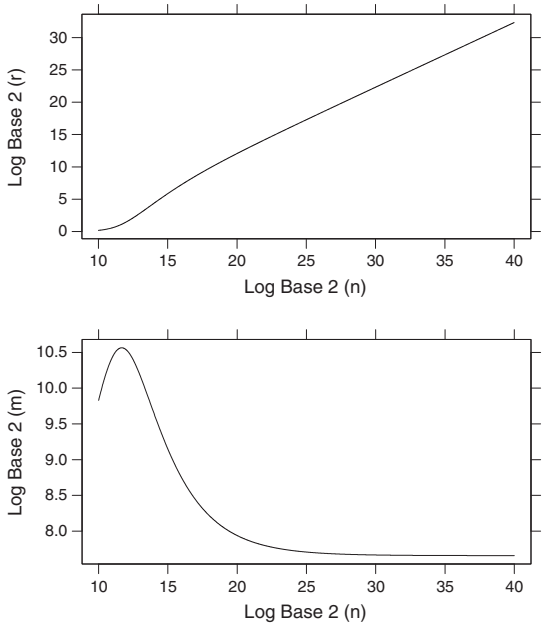

Figure 3. Theoretical Study of D&R Estimator vs. Direct Entire-Data Estimator for Regression with One Coefficient. The setting is linear regression with one explanatory variable X and no intercept, so there is just one regression coefficient. The error variance is 1. The division procedure is random-replicate. The estimation recombination procedure is the mean of the subset coefficient estimates. The D&R estimate is compared with the direct entire-data estimate. To compare the variances we take elements of X to be i.i.d normal with mean 0 and variance 1, and then study the distributions of the variances. The sample size n varies from 2^{10} to 2^{40}. For each n we find the number of subsets, r, for which the D&R variance is very close to that of the direct estimate with high probability. In the top panel, $\log_2(r)$ is plotted against $\log_2(n)$. In the bottom panel, the resulting $\log_2(m) = \log_2(n/r)$ is plotted against $\log_2(n)$. r increases with n, guaranteeing big computational gains for n as small as 2^{15}, and m stays modest in size, guaranteeing computation is feasible. This means random replicate division does extremely well even for sample sizes dramatically smaller than those that would need D&R.

In Figure 3, values of $\log_2(n)$ on the horizontal axes of the two plots range from 10 to 40 in steps of 0.01. In the top plot the vertical axis is values of r for each n from solving Equation 4. In the bottom plot the vertical axis is simply $m = n/r$, the resulting values of m arising from the solution for r. We can see that even for quite small values of n, such as $n = 2^{15}$, the D&R estimate gives results that are very close to those of least-squares, yet r is large, thereby giving a substantial computational saving. Note that throughout, m is small, in fact, well below what it needs to be. The conclusion is that random-replicate division becomes nearly optimal for this theoretical setting, starting at values of n much smaller than those where D&R needs to be used. Of course, one might be able to do even better with recombination methods that weight subset estimates by values proportional to the estimate variances or, for multiple regression, to the covariance matrices.

6 MapReduce for D&R

In this section, MapReduce is illustrated with examples. Map and Reduce are two modes of Hadoop computation. Map is an embarrassingly parallel computation mode, and does not compute across subsets. It executes part of the W computations. Reduce can compute across subsets, and executes some B computations and some of the W computations. Typically, both are involved in S computations.

The ISI's Journal for the Rapid
Dissemination of Statistics Research

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

6.1. Example: The timed logistic regression

Timings for a logistic regression model were studied in Section 4. Notation there is based on that of the regression example in Section 1. Suppose the subsets, X_s and Y_s, have been written to the HDFS. Using MapReduce terminology, inputs to Map are key-value pairs. For the example, the pairs have {key = subset ID for subset s, value = R object containing the subset s data, Y_s and X_s}. Map outputs are also key-value pairs. For the example, there is one pair for each input pair. The output values have {value = $\dot{\beta}_s$, subset s estimate of coefficients}. The output keys depend on the recombination that is intended. This will be illustrated next by two recombination procedures.

Suppose there is in addition to the explanatory variables X, another explanatory variable C that is categorical. The subset data are now X_s, Y_s, and C_s. Suppose C has categories c_i for $i = 1, \ldots, 2^{10}$; C is constant within each subset; and the c_i appear the same number of times, 2^{20}. Suppose the number of subsets is $r = 2^{\ell}$, where ℓ would be around 20 for the cluster described in Section 4 in view of the timings. Suppose the recombination is to estimate the coefficients for c_i by the vector mean of the $2^{\ell-10}$ estimates $\dot{\beta}_s$ for subsets with $C = c_i$. This is an interaction between C and each explanatory variable of X. Then the Map output key-value pairs would have {key = c_i, value = $\dot{\beta}_s$ for s with category c_i}. Reduce also has inputs and outputs that are key-value pairs. For the example, the Reduce input key-value pairs are the output key-value pairs of the Map. The first step of Reduce is to assemble input key-value pairs into groups by the key, c_i. Then it does an embarrassingly parallel computation across groups to get the vector means for the c_i. The Reduce output key-value pairs have {key = c_i, value = coefficient vector-mean estimate for c_i}.

Suppose instead that the recombination is that actually done for the timing in Section 4. The vector mean is taken across all $\dot{\beta}_s$. For this recombination, the Map output key-value pairs now have {key = constant, value = $\dot{\beta}_s$}, and Reduce only has one output with {key = constant, value = $\ddot{\beta}$}.

6.2. Example: Modeling the transmission-silence process

Our next example is the estimation of the gamma shape and scale parameters for the VoIP transmission-silence alternating process discussed in Section 3. Estimation for each semi-call resulted in a 12-tuple made up of two 6-tuples, one for silence and one for transmission. Suppose the VoIP subsets, the WSVs and BSVs of semi-calls, have been created and written to the HDFS, with each subset as a single R object. The estimation is a Map W computation. The input is {key = semi-call ID, value = semi-call R object}. The output is {key = semi-call ID, value = R object containing the 12-tuple for the semi-call}. The result is a very large number of 12-tuple R objects, very inconvenient as an overall data structure for analytic recombination. Instead, we can change the Map output and use Reduce to create more effective data structures: one R object with silence 6-tuples, and another with transmission 6-tuples. To do this, we have Map, after computation of the values of a semi-call 12-tuple, create a silence 6-tuple R object and a transmission 6-tuple R object, and then emit two key-value pairs: {key = "silence", value = silence 6-tuple} and {key = "transmission", value = transmission 6-tuple}. The Map outputs are the inputs to the Reduce, which assembles the key-value pairs into two groups, silence and transmission; creates the two data structures; and writes them to the HDFS. The two Reduce output key-value pairs have {key = silence, value = R object with silence 6-tuples} and {key = transmission, value = R object with transmission 6-tuples}.

One outcome to note is that while Reduce has been used, it was for a W computation. There has not yet been a recombination, which is a statistical analytic concept. Reduce created effective data structures for the recombination lying ahead. As described in Section 3, the recombination was analytic, and since the two 6-tuple data structures were small data, the recombination was carried out in R, with no Map or Reduce.

6.3. Example: The connection-level division of internet packet-level data

The S computations for our connection modeling of packet-level traffic employ both Map and Reduce, each applied a number of times. We describe one aspect. Packets in both directions on a link are collected in the order of their arrival on the link. Packets of different connections are intermingled, or "statistically multiplexed", in this arrival process. Processing into connection objects requires demultiplexing. Using Map, packet data objects are read into R from text files with one packet per line. The lines are input key-value pairs: {key = line identifier, value = data for one packet from the line}. The Map output is {key = connection ID formed from 4 fields in the packet data, value = R object for data of one packet}. Reduce receives them, assembles packets by connection ID, forms the R connection object, and writes it to the HDFS. The output is {key = connection ID, value = connection R object}. In this example, the embarrassingly parallel computation of the Reduce in forming the connection R objects is a major computational gain.

Acknowledgements

Adrian Baddeley and Nicholas Fisher provided detailed comments on an early version of this paper that had a substantial impact on how we ultimately described our research in D&R. We are very grateful for the insights they provided. In addition, an anonymous reviewer made several useful comments.

This work was supported in part by the Army Research Office MURI Program under award W911NF-08-1-0238, the National Science Foundation FODAVA Program under award CCF-0937123, and the National Science Foundation SCREMS Program under Award DMS-0532217.

References

Anderson, D, Xi, B & Cleveland, WS (2012). 'Multifractal and Gaussian fractional sum-difference models for internet traffic', *Technical Report*, Department of Statistics, Purdue University.

Becker, RA, Cleveland, WS & Shyu, MJ (1996), 'The design and control of trellis display', *Journal of Computational and Statistical Graphics*, **5**, 123–155.

Cleveland, WS (1993), *Visualizing Data*, *Hobart Press*, Chicago.

Cleveland, WS & Devlin, SJ (1988), 'Locally-weighted regression: An approach to regression analysis by local fitting', *Journal of the American Statistical Association*, **83**, 596–610.

Fuentes, M, Xi, B & Cleveland, WS (2011), 'Trellis display for modeling data from designed experiments', *Statistical Analysis and Data Mining*, **4**, 133–145.

Glass, GV (1976), 'Primary, secondary, and meta-analysis of research', *Educational Researcher*, **5**, 3–8.

Guha, S (2010),'Computing environment for the statistical analysis of large and complex data', *Ph.D. Thesis*, Purdue University Department of Statistics.

Guha, S, Hafen, RP, Kidwell, P & Cleveland, WS (2009), 'Visualization databases for the analysis of large complex datasets', *Journal of Machine Learning Research*, **5**, 193–200.

Guha, S, Kidwell, P, Barthur, A, Cleveland, WS, Gerth, J & Bullard, C. (2011). 'A streaming statistical algorithm for detection of SSH keystroke packets in TCP connections', In *Operations Research, Computing,*

(wileyonlinelibrary.com) DOI: 10.1002/sta4.7

and Homeland Defense, Wood, RK & Dell, RF (eds.), Institute for Operations Research and Management Sciences, Hanover, Maryland, USA.

Hafen, RP, Anderson, DE, Cleveland, WS, Maciejewski, R, Ebert, DS, Abusalah, A, Yakout, M, Ouzzani, SM & Grannis, S (2009), 'Syndromic surveillance: STL for modeling, visualizing, and monitoring disease counts', *BMC Medical Informatics and Decision Making*, **9:21**, doi:10.1186/1472–6947–9–21.

Hornik, K (2011), 'The R FAQ', World Wide Web, http://CRAN.R-project.org/doc/FAQ/R-FAQ.html [Accessed on September 1, 2012].

Pinheiro, JC & Bates, DM (2000), *Mixed-Effects Models in S and S-Plus*, *Springer*, New York.

RHIPE (2011a), 'Core Development Group', World Wide Web, http://github.com/saptarshiguha/rhipe/.

RHIPE (2011b), 'Google Discussion Group', World Wide Web, http://groups.google.com/group/rhipe.

Sarkar, D (2008), *Lattice: Multivariate Data Visualization with R*, *Springer*, New York.

Tukey, JW. (1983). 'Another look at the future', In *Computer Science and Statistics: Proceedings of the 14th Symposium on the Interface*, Heiner, KW, Sacher, RS & Wilkinson, JW (eds.), Springer, New York, 2–8.

White, T (2011), *Hadoop: The Definitive Guide*, second edition, *O'Reilly*, Sebastopol, CA.

Xi, B, Chen, H, Cleveland, WS & Telkamp, T (2010), 'Statistical analysis and modeling of internet VoIP traffic for network engineering', *Electronic Journal of Statistics*, **4**, 58–116.