Stat
The ISI's Journal for the Rapid
Dissemination of Statistics Research
(wileyonlinelibrary.com) DOI: 10.100X/sta.0000

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], William S. Cleveland [c]

Received 00 Month 2012; Accepted 00 Month 2012

Today, small data can be analyzed deeply. This means comprehensive, detailed analysis that does not lose important information in the data. Analysis of small data can be carried out using an interactive language for data analysis (ILDA) such as R. This makes programming with the data much more efficient compared with programming with a lower-level language, saving the analyst much time. D&R extends to large complex data, deep analysis using an ILDA.

Keywords: big data; data visualization; Hadoop; statistical computing; computer cluster

1. Introduction

In D&R, the data are divided into subsets by the data analyst and written to disk. These are *S computations* because they create subsets. Then analytic methods are each applied to each subset. There are statistical methods, whose output is categorical or numeric, and visualization methods, whose output is visual. Each analytic method is applied to the subsets independently, which means without communication between the subset computations. These are *W computations* because they are within subsets. For each method, the outputs of the W computations are recombined across subsets. These are *B computations* because they are between subsets. Divide, recombine, and the S-W-B computations are illustrated in Figure 1. The three computations interact in practice in more complex ways, but the figure captures the fundamental concept. Work in D&R has two distinct areas — (1) *D&R Computational Environment:* Carries out the S, W, and B computations; (2) *Statistical Theory and Procedures:* Development of D&R division and recombination procedures.

Consider a linear regression with the model $Y = X\beta + \epsilon$. Y is $n \times 1$; X is $n \times p$; 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 $n = rm$. We divide the n observations of $p + 1$ variables into r subsets, each with m observations. X_s for $s = 1, \ldots, r$ is m rows of X, and the $m \times 1$ Y_s is

The Mozilla Foundation, San Francisco, CA USA [a] Pacific Northwest National Labs, Richland WA USA [b] Purdue University, West Lafayette, IN USA [c] GE Global Research, Albany, NY USA [d]
[†]Email: wsc@purdue.edu

Prepared using staauth.cls [Version: 2012/05/12 v1.00]

the corresponding m observations of Y. Our notational convention for the subsets will be that

$$X = \mathrm{row}_{s=1}^{r}[X_s], \quad Y = \mathrm{row}_{s=1}^{r}[Y_s],$$

where row is matrix row concatenation. These are the S computations. For each subset, the optimal least-squares estimate is

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

These estimations are the W computations. Suppose we recombine by taking the vector mean, that is, the means of the elements of $\dot{\beta}_s$ across s. The D&R estimator is a B computation,

$$\ddot{\beta} = \frac{1}{r} \sum_{s=1}^{r} \dot{\beta}_s. \tag{1}$$

D&R enables two goals for large complex data: (1) *Deep Analysis*, which means comprehensive, detailed analysis that does not lose important information in the data; (2) *Analysis Using an Interactive Language for Data Analysis (ILDA)*, such as R, which makes programming with the data very efficient, saving the analyst much time. D&R achieves the goals because the heaviest of the S, W, and B computations can be carried out using the simplest possible parallel computation: no communication between the different computational processes. In the parlance of parallel processing, the computations are "embarrassingly parallel". The outcome is that almost any analytic method of statistics, machine learning, and visualization can be applied to large complex data; this includes visualization, not just of summary statistics, but of the detailed data in its finest granularity.

2. Division and Recombination Procedures

Two major categories of division procedures are *conditioning-variable* and *replicate*. A fundamental division typically persists through much of an analysis thread, which consists of the application of many analytic methods to the subsets of the division and to the outputs of the W computations. Recombination is far broader because it varies with each method of analysis. Three categories that account for much usage are *estimation*, *visualization*, and *analytic*.

2.1. D&R Illustrated

We will illustrate D&R procedures with our own analyses of large complex packet-level Internet traffic data (Xi et al., 2010; Guha et al., 2011; Anderson et al., 2012). Each Internet communication consists of a connection between two hosts (computers). An example is a client host downloading a Web page from a server host. Information to be 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 from both directions of an Internet link: arrival times and header fields. We do this for research in network performance and in cyber security. 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. Often, at least one analysis thread of each investigation is modeling by connection. So the fundamental division in this case is by connection. Each connection subset is a table. The columns are variables: the timestamp and header fields. The rows are the packets, so the number of rows varies by connection.

Here, we focus on one collection, a smaller one, but still in the category of large complex data (Xi et al., 2010). This was an investigation of the Voice over the Internet Protocol (VoIP). We collected data on the Global Crossing core VoIP network. The hosts in this case are 27 gateways to the VoIP network. Each connection has two directions, or semi-calls, of a call. However, because we model each direction independently in studying VoIP, each subset is a

semi-call. Collection was on a link connected to one gateway in Newark, NJ USA. The source host or the destination host of each semi-call is Newark, and the other host is one of the other 26 gateways. The data consist of 1.237 billion packets for 277,540 semi-calls.

Those packets of a semi-call carrying voice occur in transmission intervals that are separated by silence intervals. So a semi-call is a stochastic process of alternating transmission and silence intervals. A silence notification packet starts a silence interval and a packet containing voice ends it. This allows the start and end of each interval to be identified. The numbers of silence and transmission intervals for a semi-call differ by no more than one.

2.2. Conditioning-Variable Division

Conditioning-variable division is a fundamental division procedure that is heavily dependent on the subject matter under investigation. Certain variables important to the analysis are selected as *conditioning variables*. The data are divided into subsets by conditioning on values of the selected variables. The conditioning variables become *between subset-variables (BSVs)* with one value per subset. Other variables, *within-subset variables (WSVs)*, vary within each subset. We study the relationship of the WSVs in each subset and how it changes with the BSVs. In addition, other BSVs and WSVs can be derived from the starting WSVs and BSVs.

For the VoIP data, the semi-call ID is 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. Table 1 shows the raw BSVs and WSVs that come directly from the header fields and timestamp, and BSVs and WSVs derived from the raw variables.

Conditioning-variable division is not new and is already widely used for small and moderate data because it is a powerful mechanism for analysis of data whatever the size, from small to large. 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). So conditioning-variable division is a statistical best practice generally, but for large complex data, it also contributes substantially to computational feasibility.

2.3. Replicate Division

Another fundamental division procedure of D&R is *replicate division*. The need for this arises in many different situations. One is when subsets are still too large after conditioning-variable 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 without replacement to create the subsets. This is attractive because it is computationally fast. But it makes no effort to create subsets where each is representative of the data as a whole. *Near-exact-replicate (NER) division* makes the effort. The n observations are broken up into local neighborhoods, each with the same number of observations or close to it; each replicate subset is formed by choosing one point from each neighborhood. Of course, this takes some computing, which is itself a challenge.

2.4. Estimation Recombination

Estimation recombination is a procedure that is an estimator when a statistical method is used for analysis. An example is the D&R estimator $\ddot{\beta}$ of the coefficients in the linear regression of Section 1. A D&R estimator uses all of the data, but is typically not the same as the estimator based on direct computation on all of the data. Theoretical issues of D&R estimation are discussed in Section 5.

2.5. Analytic Recombination

Analytic recombination is simply a continued analysis of the output of a W computation. This is quite common, and often, when the output is 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 3 and 6.

The semi-call division of the VoIP data is a fundamental division for the semi-call modeling. A critical task was the modeling of the alternating transmission-silence interval length process. To carry this out, we applied many statistical and visualization methods to just the lengths, which become the detailed data of the alternating process. From this analysis, we concluded the following: the alternating process consists of mutually independent lengths; the lengths for each set are identically distributed; the square roots of the lengths for each set 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 set of lengths, 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 log scale became the parameters of the models. We used the bootstrap to estimate the variances and the covariance of the estimates of these parameters. 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 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 2 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.

2.6. Visualization Recombination

Visualization recombination provides a mechanism for rigorous visualization of the detailed data at their finest granularity (Guha et al., 2009). We choose a visualization method and apply it to certain subsets. Application of the method starts with a statistical W computation on each subset, resulting in numeric and categorical output. The output of each subset is plotted. The visualization recombination is a display that combines all subset plots.

Typically, it is feasible to apply a statistical method to all subsets of a division. The difficulty with a visualization method is that almost always there are too many subsets to view plots of them all. So we use statistical sampling methods to choose a sample of subsets to visualize. The sampling is guided by BSVs, some computed just for the sampling. The approach is exactly that used for survey sampling, but here we have great power because the data are in hand, and we can readily get BSVs for guidance. True, the sampling is a data reduction, but one that is systematic.

We have used three methods of sampling for visualization: *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 subregion 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 visualizations for a representative sample.

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 methods were used in the modeling of the silence-transmission process in two ways. It showed the gamma was a good approximation of the marginal of the square root lengths, and that the bivariate normal was a good approximation of the bootstrap distribution. An important factor for this model checking was the number of intervals. For the lengths, we had to determine if a changing number in a semi-call changed the marginal. It did not. We expected the bootstrap distribution would tend to normal with the number. It did so, but much faster for the logs of the parameters, in fact, was even a reasonable approximation for the smallest numbers. That let to modeling log shape and log scale. We used representative sampling for the quantile plot method and sampled by semi-call. The BSV was the total number of lengths in a semi-call, and the sample size was 2000 semi-calls. Semi-call subsets were chosen to range from the smallest number of intervals to the largest number, and to be as nearly equally spaced on a log scale as possible. This resulted in 2000 sets of silence intervals and 2000 sets for transmission, which was 4000 quantile plots for each application of the method.

3. A D&R Computational Environment

3.1. R-RHIPE-Hadoop

Our D&R computational environment has three open-source software components: (1) the R ILDA (Hornik, 2011); (2) the Hadoop distributed computational environment (White, 2011); (3) the RHIPE merger of R and Hadoop (Guha, 2010). R is a widely used and highly acclaimed ILDA. Its parent, the S language, won the 1999 ACM Software System Award, by far the most prestigious software award. Hadoop is a distributed software system for computing in parallel on clusters of commodity hardware. It consists of the Hadoop Distributed File System (HDFS) and the MapReduce parallel compute engine. Hadoop carries out those S, W, and B computations that are embarrassingly parallel. RHIPE enables an analyst to use D&R to analyze large complex data wholly from within R.

3.2. RHIPE

RHIPE is the R and Hadoop Integrated Programming Environment. It means "in a moment" in Greek and is pronounced "hree pay'." It was first developed in 2010 by Saptarshi Guha in his PhD thesis at Purdue in the Department of Statistics (Guha, 2010). There is now a development group working on the GitHub social coding site (RHIPE, 2011a), an R package on GitHub, and a Google discussion group (RHIPE, 2011b). Integration of R and Hadoop is accomplished by a set of components written in R and Java. The components handle the passing of information between R and Hadoop.

The data analyst writes R code to specify the subsets, which is given to RHIPE R commands that pass them on to Hadoop for execution, much of it in parallel across the cluster. Hadoop divides the data into subsets and distributes subset R objects across the nodes of the cluster in the HDFS. For the regression example of Section 1, an object contains X_s and Y_s. The analyst specifies the R commands for the W computation and gives them to RHIPE, and Hadoop carries out the computation in parallel across the cluster. For the regression example, the output of the W computation is the r subset regression coefficients $\dot{\beta}_s$. The analyst specifies the R commands for the B computation, which utilize the output of the W computation, and again gives them to RHIPE for computation by Hadoop. But now the extent of the parallel computation, which can be across subsets, depends on the recombination procedure. This is discussed in Section 6. For the regression example, there is no parallelism, just one computation of the vector mean of the $\dot{\beta}_s$.

R RHIPE commands can be used to write outputs of S, W, B computations to the HDFS. In the regression example, subsets created are written to the HDFS. This can be done for the W computation of the $\dot{\beta}_s$ if desired, but alternatively, the B computations can be run in conjunction with the W computations with W output passed on to B input in memory. Once in the HDFS, RHIPE R commands can be used to write objects to the R global environment.

A parallel computation for S, W, or B run by Hadoop consists of the same R code being applied to each object in a collection of objects. The computations are scheduled in the sense that 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, it is assigned to a new object. To minimize overall elapsed read/write time when objects need to be 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 the object is stored. In other words, Hadoop brings the core to the data, rather than the other way around.

3.3. D&R Computation in the R Global Environment: From Elephants to Mice

The big parallel R-RHIPE-Hadoop computations are "elephants". However, if output objects of the W or B computations constitute collectively small data, then the analyst can use RHIPE R commands to read them from the HDFS, and write them into the R global environment. Analysis of the output continues using R in the traditional way with interactive R commands, some of them as small as "mice" that compute virtually instantaneously. D&R analysis of a large complex dataset typically involves much of this. This is not a forced data reduction to reduce size, but rather a validated one in which a parsimonious description is found to characterize behavior in the data. We have seen one example in Section 2 in the modeling of the transmission-silence alternating process for the VoIP data. Computing in the traditional interactive R environment is as critical a part of D&R computation as the elephants. The design of the D&R environment must separate servers running R and those running RHIPE-Hadoop, to keep the elephants from trampling on the mice.

3.4. Factors and Responses for the Complex System

Factors from a spectrum of sources affect the performance of each distributed R-RHIPE 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, the knowledge on its own cannot provide answers.

We describe one small experiment to illustrate. The topic is the effect of the number of subsets on elapsed time. The response in the experiment was elapsed time of the R function `glm.fit` to carry out linear logistic regression. We use notation to describe the regression variables like that of the linear regression with i.i.d. normal errors in Section 1. X is the $n \times p$ matrix of values of p explanatory variables, and Y is the $n \times 1$ matrix of the 0-1 response values. We generated Y and X, which were numeric. We use subset notation as before as well. There are r subsets each of size m so $n = rm$.

We took $n = 2^{30}$. For a first set of runs, $p = 15$. There were 11 values of the factor $\log_2(r)$, — $13, 14, \ldots 23$ — where $\log_2(r)$ is log base 2. The log subset sizes were $\log_2(m) = 17, 16, \ldots, 7$. The number of numeric values was 2^{34}, and at 8 bytes per double-precision value in memory, the data size was 2^{37} bytes, about 128 gigabytes. For the second set of runs $p = 127$, $\log_2(r) = 18, 19, \ldots 22$, and $\log_2(m) = 12, 13, \ldots, 8$. The data size was now 2^{40}, about 1 terabyte. For both sets there were three replicate runs for each value of $\log_2(r)$. Two types of elapsed times were measured for each division. (1) *Method (M)*: In the W computation, applying the R function `glm.fit` for linear logistic regression to each subset, plus in the B computation, computing the vector mean across subsets of the regression coefficients. (2) *Read-Write (R/W)*: In the W computation, reading the subsets from the HDFS, plus in the B computation, writing the coefficient estimate vectors to the HDFS. The cluster on which the timings were run has 11 servers each with 24 processors, 48 GB of memory, and 8 terabytes of disk. The components of the cluster have average speeds, neither especially fast or slow.

The ∘'s in Figure 2 graph elapsed time for $p = 15$ against $\log_2(r)$ for M, R/W, and M + R/W. Each time is a mean of the three replicates for each $\log_2(r)$. The solid curve is a loess fit to the ∘'s of each panel set with a span = 1 and degree = 2. The +'s and dashed curves show the results for $p = 127$ in the same way.

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 is at $\log_2(r) = 18$–19, and the minimum M + R/W time is about 128 sec = 2.1 min. For $p = 127$, the minimum is at $\log_2(r) = 20$–21, and the minimum M + R/W time is about 1056 sec = 17.6 min. Again, the second is about 8 times that of the first.

The decrease and increase of the time results from two causes. For smaller r, subset object sizes are larger, so the requests for memory are larger, and can cause a bottleneck. For larger r, the number of subsets is larger, and Hadoop overhead in handling them goes up and can cause a bottleneck. So the $\log_2(r)$ where the minima occur balance the two causes.

One important matter in interpreting the absolute timings is that the hardware specifications are a very major factor. In this case, the disk, bus, and controller speeds have a major impact on R/W time. For the above experiment, about 70% of the minimum elapsed time is R/W. Faster disk could substantially improve the elapsed time.

4. Why D&R?

4.1. D&R Estimation

We return to the linear regression example in Section 1. The D&R estimator $\ddot{\beta}$ from Equation 1 is

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

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

$$\hat{\beta} = \left(\sum_{s=1}^{r}X_s'X_s\right)^{-1}\sum_{s=1}^{r}X_s'Y_s. \tag{2}$$

For the normal model, $\hat{\beta}$ is optimal, a maximum of statistical accuracy. Except for certain special cases, $\ddot{\beta} \neq \hat{\beta}$. D&R estimation is generally less accurate than the direct all-data estimator. However, for the analysis of large complex data, a direct all-data estimator is typically either infeasible or is impractical because computing takes so long. Because S-W-B computation is feasible, D&R enables use of almost any statistical method on large complex data. The division procedure and the recombination procedure of a D&R estimator for a statistical method determine the accuracy. We can use statistical theory to find "best" D&R estimators.

For some statistical methods, it is possible to develop a parallel external memory algorithm (PEMA) that can provide fast computation of the direct all-data estimator (Vitter, 2001). A PEMA also begins with division and computes on the subsets. In fact, one example is the above linear regression for p small or moderate. The terms $X_s'X_s$ and $X_s'Y_s$ in Equation 2 are computed in parallel across a cluster, and the recombination sums them and then computes the algebraic expression of the equation. In this case, the computation is even embarrassingly parallel. It can be implemented very readily by the R-RHIPE-Hadoop computational environment. Embarrassingly parallel PEMAs are rare. If we allow communication between the parallel computations, PEMAs are feasible for more methods. For some, development is straightforward. For most, it ranges from difficult to intractable. For the successes, computation is more complex than

D&R. Even if PEMAs for statistical methods were feasible, developing and implementing them soon for the 1000s of available statistical methods is infeasible. The R CRAN package repository now has about 4000 packages.

D&R is very cost effective because it can employ a heterogeneous cluster. Much less powerful nodes, within limits of course, do less work yet still contribute by reducing computation time. This contrasts with algorithms that have communication between parallel tasks, where a much less powerful node can add to computation time.

4.2. D&R Visualization

Estimators and statistical inferences based on them are one aspect of data analysis. Just as critical are data exploration and model building, which are necessary for deep analysis. Both require extensive visualization. Summaries of the detailed data need to be visualized. Detailed data at their finest granularity also need to be visualized. This was established in the 1960s by the pioneers of model building (Anscombe & Tukey, 1963; Daniel & Wood, 1971; Tukey, 1977). It is true of data of any size, from small to large and complex.

D&R enables visualization of the detailed data through the statistical sampling described in Section 2. The sampling is in fact a data reduction, but one which is rigorous. Even for those limited number of statistical methods for which a PEMA exists for estimation recombination, D&R is still necessary for data exploration and model building. For the above linear regression, there is a wide variety of visualization methods for model diagnostics (Cook & Weisberg, 1982; D. A. Belsley, 2005). D&R enables this.

5. Statistical Theory for D&R Estimation Recombination

The accuracy of D&R estimation for a statistical method depends on the statistical division and recombination procedures that are used. D&R statistics research uses statistical theory to study accuracy and to find best D&R estimators. As part of this, we can let the sample size of all of the data increase. 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 size and let the number of subsets increase.

We illustrate such study by continuing the example of linear regression with i.i.d. normal errors treated in previous sections. Even though there is a fast PEMA, it is very instructive to study this case theoretically because the mathematics is tractable, making it an instructive leading case for intuition about other methods and models for the dependence of a response on explanatory variables. We illustrate by an investigation of the performance of random-replicate division for one $p = 1$, one explanatory variable, so X in this case is an $n \times 1$ matrix with no intercept. 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 all-data case as part of D&R. 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. Suppose the division procedure is random-replicate division, so r subsets each of size m are chosen randomly. Then $\sigma^2(\dot{\beta}_s)$ for $s = 1, \ldots, r$ are i.i.d. The ratio of the variances given X is

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

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

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

A simple derivation shows

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

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 $\mu + c\sqrt{\tau} = 1.01$, not restricting r to an integer. (We could tighten the argument by taking n to be certain values, say powers of 2, and rounding 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.) For this r, the probability of the D&R variance being more than 1% greater than the variance of the all-data least-squares estimate is 0.01. We took values of $\log_2(n)$ ranging from 10 to 40 in steps of 0.01. Figure 3 graphs the resulting values of $\log_2(r)$ and its accompanying value of $\log_2(m)$ against $\log_2(n)$ for n ranging from about a thousand to a trillion. 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.

6. MapReduce for D&R

Map and Reduce are two modes of Hadoop computation. Each has input key-value pairs, and output key-value pairs. Map does embarrassingly parallel computations of the same R code for each of the input key-value pairs. Map is heavily used in S and W computations. Reduce can compute across key-value pairs, and does embarrassingly parallel computation for certain of its tasks. It is heavily used in B computations, but plays a role in many S and W computations.

6.1. Example: The Timed Logistic Regression

Suppose the subsets of the logistic regression timing of Section 3 are in place in the HDFS. Map executes the W computation: user specified R code, passed to Hadoop by a RHIPE R command, that runs `glm.fit` and extracts the coefficients from the R object returned by the fit. The input key-value pairs have {key = subset ID, value = R object containing the subset data, Y_s and X_s}. The output values of the Map are {value = $\dot{\beta}_s$, subset estimates of coefficients}. The output keys depend on the coming recombination. To explain, we will depart for a moment from what was actually done. Suppose there is a categorical variable with 2^{10} categories, the values for each subset are the same category, and there are 2^{r-10} subsets for each category. Suppose the recombination is to estimate the coefficients separately for each category by the $\dot{\beta}_s$ means for the category. Then the Map output has {key = category for $\dot{\beta}_s$}. The Map output key-value pairs are the input key-value pairs for the Reduce, whose first step is to assemble them into groups by category. Then it does an embarrassingly parallel computation to compute means. The 2^{r-10} Reduce

output key-value pairs have {key = category, value = vector mean of β_s for category}. Now we go back to the actual computation in which the vector mean of all β_s were computed by Reduce. The Map output now has {key = constant, value = $\hat{\beta}_s$}. With a single key, the Reduce does one vector mean computation across all $\hat{\beta}_s$. The output is {key = constant, value = vector mean}.

6.2. Example: Modeling the Transmission-Silence Process

Suppose each VoIP subset, the WSVs and BSVs of a semi-call, has been created and written to the HDFS as an R object. For the D&R estimation in the transmission-silence interval length modeling, Map executes the W computations to get the 12-tuples. The Map 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}. The result then is a very large number of R objects, very inconvenient as an overall data structure. We use Reduce to create two data structures: an R object containing the silence 6-tuples, and another with the transmission 6-tuples. The Map output is {key = "silence" or "transmission", value = 6-tuple}. This is passed to Reduce which assembles the key-value pairs into two groups, silence and transmission; computes the two data structures; and writes them to the HDFS. The two output key-value pairs have {key = "silence" or "transmission", value = R data structure with 6-tuples}. The critical thing to note is that while Reduce has been used, there has not yet been a recombination, which is statistical analytic concept. All that has been done is to create effective data structures for the recombination lying ahead.

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". 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 = one packet data}. The Map output is {key = connection ID formed from 4 fields in the packet data, value = R object for one packet data}. 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.

7. Discussion: The Roles of MapReduce and D&R

Map and Reduce are brilliant computational tools that along with the HDFS enable D&R. Map and Reduce do not correspond to divide and recombine or even to the S, W, and B computations in a rules-based way. The examples show that Map and Reduce are used for S and W computations. Reduce is important for recombination in many circumstances, but not all. It is typically not needed for visualization recombination beyond forming data structures. When outputs of W or B are small and in need of analytical recombination, they are read into R and analyzed there without Hadoop at all.

D&R addresses division and recombination procedures, which have an immense impact on the effectiveness of the analysis: how much we learn. Map and Reduce and the associated computational technology do not provide this. D&R utilizes basic statistical thinking and statistical theory to develop high-performance D&R procedures. D&R also addresses the programming environment of the analyst. RHIPE puts the analysis in the R environment. This makes analysis more efficient, which in turn contributes much to the effectiveness of the analysis. RHIPE provides access to Hadoop. Without such access, Hadoop by itself could not succeed in providing deep analysis of large complex data in which almost any method of statistics, machine learning, and visualization could be applied.

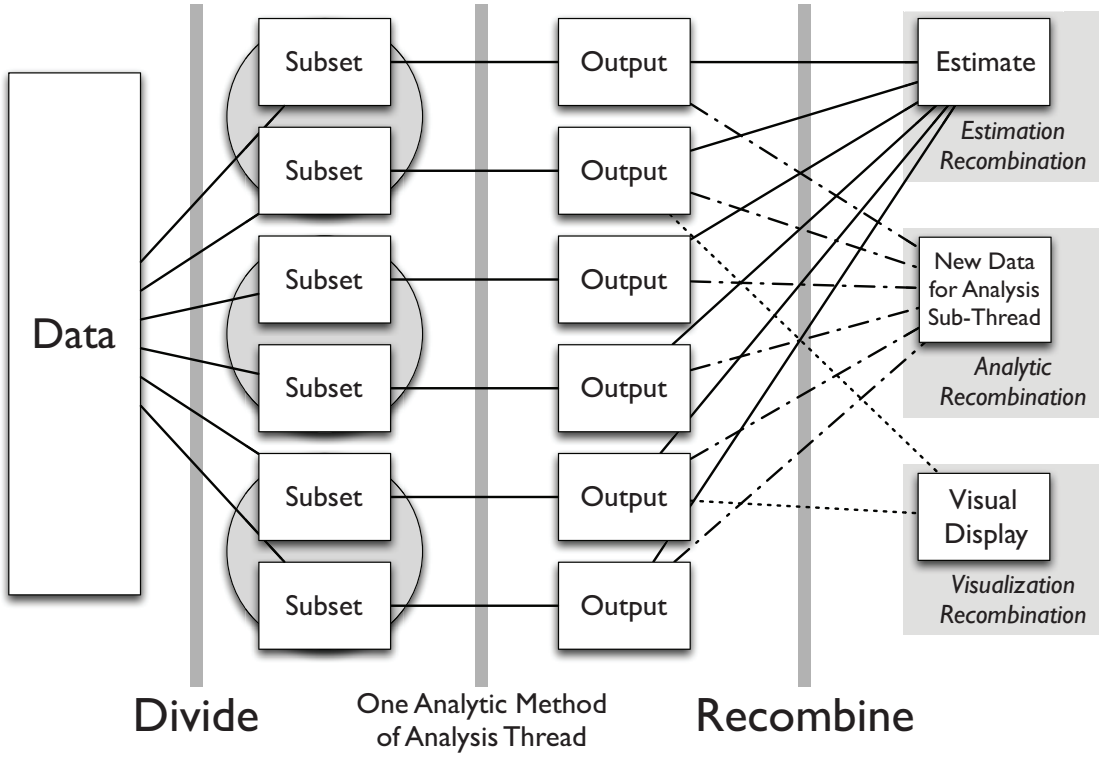

Figure 1. D&R parallelizes the data. S, W, and B computations carry out D&R, and the heaviest computing is embarrassingly parallel. S divides into subsets, W applies an analytic method to each subset, and B applies analytic methods to the outputs of W to achieve recombination. Three recombination methods are estimation, visualization, and further analysis. The parallel computations of S, W, and R are carried out on a cluster by the Hadoop distributed file system and compute engine. RHIPE, the R and Hadoop Integrated Programming Environment allows the data analyst to carry out D&R wholly from within R, which makes programming with the data very efficient. The figure gets basic ideas across, but actual analyses of large complex data usually combine multiple S, W, and B computations. Typically there are a number of analysis threads, each with its own fundamental division, and each thread has many sub-threads.

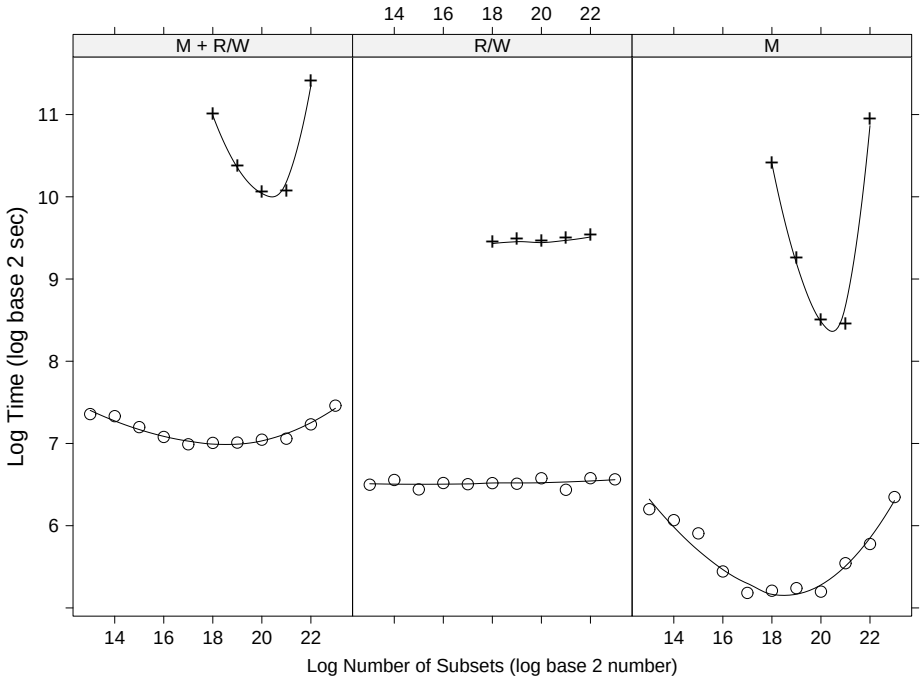

Figure 2. Timings of D&R estimation for logistic regression using the `glm.fit` program in R were carried out for 2^{30} observations of a response with $p = 15$ numeric explanatory variables in one set of runs, and $p = 127$ in another set. The memory size of the two datasets are 128 gigabytes and 1 terabyte, respectively. The number of subsets, r, was varied. In each panel, log elapsed time is plotted against $\log_s(r)$ ($'\circ' = 15$; $'+' = 127$). Loess curves are fitted to each set for each time. M is execution of `glm.fit` plus computing the mean across subsets of the subset estimates of each of the p coefficients. R/W is reading the subsets from the HDFS, and writing the coefficient means to the HDFS. The cluster has 11 nodes, each with 24 processors, 48 GB of memory, and 8 TB of disk. The number of subsets is a major factor for elapsed time. Computation time on this cluster is excellent, and not an impediment to analysis.

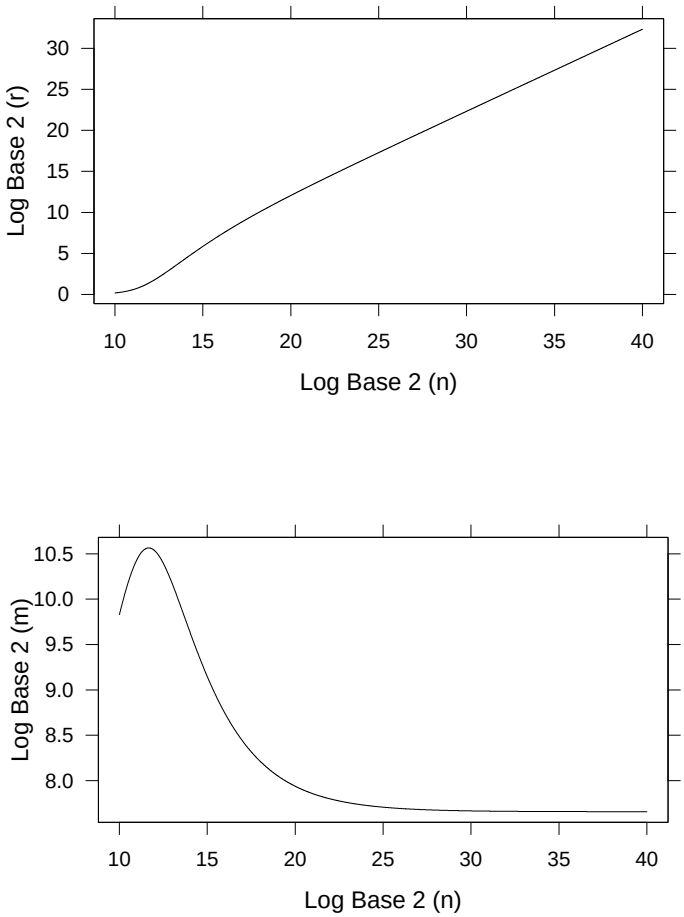

Figure 3. The setting is linear regression with one explanatory variable X and no intercept, so there is just one regression coefficient. The error variance is 1. The division procedure is random-replicate. The estimation recombination procedure is the mean of the subset coefficient estimates. The D&R estimate is compared with the direct all-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 $\log_2(m)$ 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.

Table 1. The table shows the BSVs (between-subset variables) and the WSVs (within-subset variables) of the semi-call subsets for the VoIP packet-level data. Raw variables are those that come directly from the timestamping and the packet headers. 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

References

Anderson, D, Xi, B & Cleveland, WS (2012), 'Multifractal and Gaussian Fractional Sum-Difference Models for Internet Traffic,' Tech. rep., Department of Statistics, Purdue University.

Anscombe, FJ & Tukey, JW (1963), 'The Examination and Analysis of Residuals,' *Technometrics*, **5**, pp. 141–160.

Becker, RA, Cleveland, WS & Shyu, MJ (1996), 'The Design and Control of Trellis Display,' *Journal of Computational and Statistical Graphics*, **5**, pp. 123–155.

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

Cook, R & Weisberg, S (1982), *Residuals and Influence in Regression*, Chapman and Hall.

D. A. Belsley, REW, E. Kuh (2005), *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*, Wiley, New York.

Daniel, C & Wood, F (1971), *Fitting Equations to Data*, Wiley, New York.

Fuentes, M, Xi, B & Cleveland, WS (2011), 'Trellis Display for Modeling Data from Designed Experiments,' *Statistical Analysis and Data Mining*, **4**, pp. 133–145.

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**, pp. 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 Wood, RK & Dell, RF (eds.), *Operations Research, Computing, and Homeland Defense*, Institute for Operations Research and Management Sciences.

Hornik, K (2011), 'The R FAQ,' World Wide Web, http://CRAN.R-project.org/doc/FAQ/R-FAQ.html.

Pinheiro, J & Bates, D (2000), *Mixed-Effects Models in S and S-PLUS*, Springer.

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 (1977), *Exploratory Data Analysis*, Addison-Wesley, Reading, Massachusetts.

Tukey, JW (1983), 'Another look at the future,' in K. W. Heiner, RSS & Wilkinson, JW (eds.), *Computer Science and Statistics: Proc. 14th Sy mposium on the Interface*, Springer, New York, pp. 2–8.

Vitter, JS (2001), 'External Memory Algorithms and Data Structures: Dealing with Massive Data,' *ACM Computing Surveys*, **33**, pp. 209–271.

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**, pp. 58–116.