A Survey of Statistical Methods and Computing for Big Data

Chun Wang, Ming-Hui Chen, Elizabeth Schifano, Jing Wu, and Jun Yan

March 2, 2015

Abstract

Big data are data on a massive scale in terms of volume, intensity, and complexity that exceed the capacity of standard software tools. They present opportunities as well as challenges to statisticians. The role of computational statisticians in scientific discovery from big data analyses has been under-recognized even by peer statisticians. This article reviews recent methodological and software developments in statistics that address the big data challenges. Methodologies are grouped into three classes: subsampling-based, divide and conquer, and sequential updating for stream data. Software review focuses on the open source R and R packages, covering recent tools that help break the barriers of computer memory and computing power. Some of the tools are illustrated in a case study with a logistic regression for the chance of airline delay.

Keywords: bootstrap, divide and conquer, external memory algorithm, high performance computing, online update, sampling, software

1 Introduction

A 2011 McKinsey report predicted shortage of talent necessary for organizations to take advantage of big data (Manyika *et al.*, 2011). Data now stream from daily life thanks to technological advances, and big data has indeed become a big deal (e.g., Shaw, 2014). In the President's Corner of the June 2013 issue of AMStat News, the three presidents (elect, current, and past) of the American Statistical Association (ASA) wrote an article titled "The ASA and Big Data" (Schenker *et al.*, 2013). This article echos the June 2012 column of Rodriguez (2012) on the recent media focus on big data, and discusses on what the statistics profession needs to do in response to the fact that statistics and statisticians are missing from big data discussions. In the followup July 2013 column, president Marie Davidian further raised the issues of statistics not being recognized as data science and mainstream academic statisticians being left behind by the rise of big data (Davidian, 2013). A white paper prepared by a working group of the ASA called for more ambitious efforts from statisticians to work together with researchers in other fields on national research priorities in order to achieve better science more quickly (Rudin *et al.*, 2014). The same concern was expressed in a 2014 president's address of the Institute of Mathematical Statistics (IMS) (Yu, 2014). President Bin Yu of the IMS called for statisticians to own Data Science by working on real problems such as those from genomics, neuroscience, astronomy, nanoscience, computational social science, personalized medicine/healthcare, finance, and government; relevant methodology/theory will follow naturally.

Big data in the media or the business world may mean differently than what are familiar to academic statisticians (Jordan & Lin, 2014). Big data are data on a massive scale in terms of volume, intensity, and complexity that exceed the ability of standard software tools to manage and analyze (e.g., Snijders *et al.*, 2012). The origin of the term "big data" as it is understood today has been traced back in a recent study (Diebold, 2012) to lunch-table conversations at Silicon Graphics in the mid-1990s, in which John Mashey figured prominently (Mashey, 1998). Big data are generated by countless online interactions among people, transactions between people and systems, and sensor-enabled machinery. Internet search engines (e.g., Google and YouTube) and social network tools (e.g., Facebook and Twitter) generate billions of activity data per day. Rather than Gigabytes and Terabytes,

nowadays, the data produced are estimated by zettabytes, and are growing 40% every day (Fan & Bifet, 2013). In the big data analytics world, a 3V definition by Laney (2001) is widely accepted: volume (amount of data), velocity (speed of data in and out), and variety (range of data types and sources). High variety brings nontraditional or even unstructured data types, such as social network sentiments and internet map usage, which calls for new, creative ways to understand the structure of data and even to ask intelligent research questions (e.g., Jordan & Lin, 2014). High volume and high velocity may bring noise accumulation, spurious correlation and incidental homogeneity, creating issues in computational feasibility and algorithmic stability (Fan et al., 2013).

Notwithstanding that new statistical thinking and methods are needed for the high variety aspect of big data, our focus is on fitting standard statistical models to big data whose size exceeds the capacity of a single computer from its high volume and high velocity. There are two computational barriers for big data analysis: 1) the data can be too big to hold in a computer's memory; and 2) the computing task can take too long to wait for the results. These barriers can be approached either with newly developed statistical methodologies and/or computational methodologies. Despite the impression that statisticians are left behind in media discussions or governmental summits on big data, some statisticians have made important contributions and are pushing the frontier. Sound statistical procedures that are scalable computationally to massive datasets have been proposed (Jordan, 2013). Examples are subsampling-based approaches (Kleiner et al., 2014; Liang et al., 2013; Ma et al., 2013), divide and conquer approaches (Lin & Xi, 2011; Chen & Xie, 2014), and sequential updating approaches (Schifano et al., 2014). From a computational perspective, much effort has been put into the most active, open source statistical environment, R (R Core Team, 2014a). Statistician R developers are relentless in their drive to extend the reach of R into big data (Rickert, 2013). Recent UseR! conferences had many presentations that directly addressed big data, including a 2014 keynote lecture by John Chambers, the inventor of the S language (Chambers, 2014). Most cutting edge methods are first and easily implemented in R. Given the open source nature of R and the active recent development, our focus on software for big data will be on R and R packages. Revolution R Enterprise (RRE) is a commercialized version of R, but it offers free academic

use, so it is also included in our case study and benchmarked. Other commercial software such as SAS, SPSS, and MATLAB will be briefly touched upon for completeness.

The rest of the article is organized as follows. Recent methodological developments in statistics on big data are summarized in Section 2. Resources from open source software R for analyzing big data with classical models are reviewed in Section 3. Commercial software products are presented in Section 4. A case study on a logistic model for the chance of airline delay is presented in Section 5. A discussion concludes in Section 6.

2 Statistical Methods

The recent methodologies for big data can be loosely grouped into three categories: resamplin based, divide and conquer, and sequential updating. To put the different methods in a context, consider a dataset with n independent and identically distributed observations, where n is too big for standard statistical routines such as logistic regression.

2.1 Subsampling-Based Methods

2.1.1 Bags of Little Bootstrap

Kleiner *et al.* (2014) proposed the bags of little bootstrap (BLB) approach that provides both point estimates and quality measures such as variance or confidence intervals. It is a combination of subsampling (Politis *et al.*, 1999), the m-out-of-n bootstrap (Bickel *et al.*, 1997), and the bootstrap (Efron, 1979) to achieve computational efficiency. BLB consists of the following steps. First, draw s subsamples of size m from the original data of size n. For each of the s subsets, draw r bootstrap samples of size n instead of m, and obtain the point estimates and their quality measures (e.g., confidence interval) from the r bootstrap sample. Then, the s bootstrap point estimates and quality measures are combined (e.g., by average) to yield the overall point estimates and quality measures. In summary, BLB has two nested procedures: the inner procedure applies the bootstrap to a subsample, and the outer procedure combines these multiple bootstrap estimates. The subsample size m was suggested to be n^γ with $\gamma \in [0.5, 1]$ (Kleiner *et al.*, 2014), a much smaller number than n. Although the inner bootstrap procedure conceptually generates multiple resampled data of

size n, what is really needed in the storage and computation is a sample of size m with a weight vector. In contrast to subsampling and the m-out-of-n bootstrap, there is no need for an analytic correction (e.g., $\sqrt{m/n}$) to rescale the confidence intervals from the final result. The BLB procedure facilitates distributed computing by letting each subsample of size m be processed by a separate processor. Kleiner *et al.* (2014) proved the consistency of BLB and provided high order correctness. Their simulation study showed good accuracy, convergence rate and the remarkable computational efficiency.

2.1.2 Mean Log-likelihood

Liang *et al.* (2013) proposed a resampling-based stochastic approximation approach with an application to big geostatistical data. The method uses Monte Carlo averages calculated from subsamples to approximate the quantities needed for the full data. Motivated from minimizing the Kullback–Leibler (KL) divergence, they approximate the KL divergence by averages calculated from subsamples. This leads to a maximum mean log-likelihood estimation method. The solution to the mean score equation is obtained from a stochastic approximation procedure, where at each iteration, the current estimate is updated based on a subsample of size m drawn from the full data. As m is much smaller than n, the method is scalable to big data. Liang *et al.* (2013) established the consistency and asymptotic normality of the resulting estimator under mild conditions. In a simulation study, the convergence rate of the method was almost independent of n, the sample size of the full data. Liang & Kim (2013) extended the mean log-likelihood into a bootstrap Metropolis–Hastings algorithm in Markov chain Monte Carlo (MCMC). The likelihood ratio of the proposal and current estimate in the Metropolis–Hastings ratio is replaced with that approximated from the mean log-likelihood based on k bootstrap samples of size m. The algorithm can be implemented exploiting the embarrassingly parallel structure and avoids repeated scans of the full dataset in iterations.

2.1.3 Leveraging

Ma & Sun (2014) proposed to use leveraging to facilitate scientific discoveries from big data using limited computing resources. In a leveraging method, one samples a small proportion

of the data with certain weights (subsample) from the full sample, and then performs intended computations for the full sample using the small subsample as a surrogate. The key to success of the leveraging methods is to construct the weights, the nonuniform sampling probabilities, so that influential data points are sampled with high probabilities (Ma *et al.*, 2013). Leveraging methods are different from the traditional subsampling or m-out-of-n bootstrap in that 1) they are used to achieve feasible computation even if the simple analytic results are available; 2) they enable visualization of the data when visualization of the full sample is impossible; and 3) they usually use unequal sampling probabilities for subsampling data. This approach is quite unique in allowing pervasive access to extract information from big data without resorting to high performance computing.

2.2 Divide and Conquer

A divide and conquer algorithm has three steps: 1) partitions a big dataset into K blocks; 2) processes each block separately (possibly in parallel); and 3) aggregates the solutions from each block to form a final solution to the full data. For a linear regression model, the least squares estimator for the regression coefficient β for the full data can be expressed as a weighted average of the least squares estimator for each block with weight being the inverse of the estimated variance matrix. The success of this method for linear regression depends on the linearity of the estimating equations in β and that the estimating equation for the full data is a simple summation of that for all the blocks. For general nonlinear estimating equations, Lin & Xi (2011) proposed a linear approximation of the estimating equations with the Taylor expansion at the solution in each block, and, hence, reduce the nonlinear estimating equation to the linear case so that the solutions to all the blocks are combined by a weighted average. The weight of each block is the slope matrix of the estimating function at the solution in that block, which is the Fisher information or inverse of the variance matrix if the equations are score equations. Lin & Xi (2011) showed that, under certain technical conditions including $K = O(n^\gamma)$ for some $\gamma \in (0, 1)$, the aggregated estimator has the same limit as the estimator from the full data.

Chen & Xie (2014) consider a divide and conquer approach for generalized linear models (GLM) where both the number of observations n and the number of covariates p are large

by incorporating variable selection via penalized regression into subset processing step. More specifically, for p bounded or increasing to infinity slowly, (p_n not faster than $o(e^{n_k})$, while model size may increase at a rate of $o(n_k)$), they propose to first randomly split the data of size n into K blocks (size $n_k = O(n/K)$). In step 2, penalized regression is applied to each block separately with a sparsity-inducing penalty function satisfying certain regularity conditions. This approach can lead to differential variable selection among the blocks, as different blocks of data may result in penalized estimates with different non-zero regression coefficients. Thus, in step 3, the results from the K blocks are combined by majority vote to create a combined estimator. The implicit assumption is that real effects should be found persistently and therefore should be present even under perturbation by subsampling (e.g. Meinshausen & Buhlmann, 2010). The derivation of the combined estimator in step 3 stems from ideas for combining confidence distributions in meta-analysis (Singh *et al.*, 2005; Xie *et al.*, 2011), where one can think of the K blocks as K independent and separate analyses to be combined in a meta-analysis. The authors show under certain regularity conditions that their combined estimator in step 3 is model selection consistent, asymptotically equivalent to the penalized estimator that would result from using all of the data simultaneously, and achieves the oracle property when it is attainable for the penalized estimator from each block (see e.g., Fan & Lv, 2011). They additionally establish an upper bound for the expected number of incorrectly selected variables and a lower bound for the expected number of correctly selected variables.

2.3 Sequential Updating for Stream Data

In some applications, the data come in streams or large chunks, and a sequentially updated analysis is desirable without storage requirements. Schifano *et al.* (2014) expand upon the work of Lin & Xi (2011) in several important ways. First, they introduce divide-and-conquer-type variance estimates of regression parameters in the linear model and estimating equation settings. These estimates of variability allow for users to make inferences about the true regression parameters based upon the previously developed divide-and-conquer point estimates of the regression parameters. Second, they develop iterative estimating algorithms and statistical inferences for linear models and estimating equations that up-

date as new data arrive. Thus, while the divide-and-conquer setting is quite amenable to parallel processing for each subset, the online-updating approach for data streams is inherently sequential in nature. Their algorithms were designed to be computationally efficient and minimally storage-intensive, as they assume no access/storage of the historical data. Third, the authors address the issue of possible rank deficiencies when dealing with blocks of data, and the uniqueness properties of the combined and cumulative estimators when using a generalized inverse. The authors also provide methods for assessing goodness of fit in the linear model setting, as standard residual-based diagnostics cannot be performed with the cumulative data without access to historical data. Instead, they propose outlier tests relying on predictive residuals, which are based on the predictive values computed from the cumulative estimate of the regression coefficients attained at the previous accumulation point. Additionally, they introduce a new online-updated estimator of the regression coefficients and corresponding estimator of the standard error in the estimating equation setting that takes advantage of information from the previous data. They show theoretically that this new estimator, the cumulative updated estimating equation (CUEE) estimator, is asymptotically consistent, and show empirically that the CUEE estimator is less biased in their finite sample simulations than the cumulatively estimated version of the estimator of Lin & Xi (2011).

3 Open Source R and R Packages

Handling big data is one of the topics of high performance computing. As the most popular open source statistical software, R and its adds-on packages provide a wide range of high performance computing; see Comprehensive R Archive Network (CRAN) task view on "High-Performance and Parallel Computing with R" (Eddelbuettel, 2014). The focus of this section is on how to break the computer memory barrier and the computing power barrier in the context of big data.

3.1 Breaking the Memory Barrier

The size of big data is relative to the available computing resources. The theoretical limit of random access memory (RAM) is determined by the width of memory addresses: 4 gigabyte (GB) (2^{32} bytes) for a 32-bit computer and 16.8 million terabyte (2^{64} bytes) for a 64-bit computer. In practice, however, the latter is limited by the physical space of a computer case, the operating system, and specific software. Individual objects in R have limits in size too; an R user can hardly work with any object of size close to that limit. Emerson & Kane (2012) suggested that a data set would be considered *large* if it exceeds 20% of RAM on a given machine and *massive* if it exceeds 50%, in which case, even the simplest calculation would consume all the remaining RAM.

Memory boundary can be broken with an external memory algorithms (EMA) (e.g., Vitter, 2001), which conceptually works by storing the data on a disk storage (which has a much greater limit than RAM), and processing one chunk of it at a time in RAM (e.g., Lumley, 2013). The results from each chunk will be saved or updated and the process continues until the entire dataset is exhausted; then, if needed as in an iterative algorithm, the process is reset from the beginning of the data. To implement an EMA for each statistical function, one need to address 1) data management and 2) numerical calculation.

3.1.1 Data Management

Earlier solutions to oversize data resorted to relational databases. This method depends on an external database management system (DBMS) such as MySQL, PostgreSQL, SQLite, H2, ODBC, Oracle, and others. Interfaces to R are provided through many R packages such as **sqldf** (Grothendieck, 2014), **RDBI** (Hollensbe, 2009), **RSQLite** (Wickham *et al.*, 2014), and others. The database approach requires a DBMS to be installed and maintained, and knowledge of structured query language (SQL); an exception for simpler applications is package **filehash** (Peng, 2006), which comes with a simple key-value database implementation itself. The numerical functionality of SQL is quite limited, and calculations for most statistical analyses require copying subsets of the data into objects in R facilitated by the interfaces. Extracting chunks from an external DBMS is computationally much less efficient than the more recent approaches discussed below (Kane *et al.*, 2013).

Two R packages, **bigmemory** (Kane *et al.*, 2013) and **ff** (Adler *et al.*, 2014) provide data structures for massive data while retaining a look and feel of R objects. Package **bigmemory** defines a data structure `big.matrix` for numeric matrices which uses memory-mapped files to allow matrices to exceed the RAM size on computers with 64-bit operating systems. The underling technology is memory mapping on modern operating systems that associates a segment of virtual memory in a one-to-one correspondence with contents of a file. These files are accessed at a much faster speed than in the database approaches because operations are handled at the operating-system level. The `big.matrix` structure has several advantages such as support of shared memory for efficiency in parallel computing, reference behavior that avoids unnecessary temporary copies of massive objects, and column-major format that is compatible with legacy linear algebra packages (e.g., BLAS, LAPACK) (Kane *et al.*, 2013). The package provides utility to read in a csv file to form a `big.matrix` object, but it only allows one type of data, numeric; it is a numeric matrix after all.

Package **ff** provides data structures that are stored in binary flat files but behave (almost) as if they were in RAM by transparently mapping only a section (pagesize) of meta data in main memory. Unlike **bigmemory**, it supports R's standard atomic data types (e.g., double or logical) as well as nonstandard, storage efficient atomic types (e.g., the 2-bit unsigned `quad` type allows efficient storage of genomic data as a factor with levels A, T, G, and, C). It also provides class `ffdf` which is like `data.frame` in R, and import/export filters for csv files. A binary flat file can be shared by multiple `ff` objects in the same or multiple R processes for parallel access. Utility functions allow interactive process of selections of big data.

3.1.2 Numerical Calculation

The data management systems in packages **bigmemory** or **ff** do not mean that one can apply existing R functions yet. Even a simple statistical analysis such as linear model or survival analysis will need to be implemented for the new data structures. Chunks of big data will be processed in RAM one at a time, and often, the process needs to be iterated over the whole data. A special case is the linear model fitting, where one pass of the data is sufficient and no resetting from the beginning is needed. Consider a regression model

$E[Y] = X\beta$ with $n \times 1$ response Y, $n \times p$ model matrix X and $p \times 1$ coefficient β. The base R implementation `lm.fit` takes $O(np + p^2)$ memory, which can be reduced dramatically by processing in chunks. The first option is to compute $X'X$ and $X'y$ in increment, and get the least squares estimate of β, $\hat{\beta} = (X'X)^{-1}X'Y$. This method is adopted in package **speedglm** (Enea, 2014). A slower but more accurate option is to compute the incremental QR decomposition (Miller, 1992) of $X = QR$ to get R and $Q'Y$, and then solve β from $R\beta = Q'Y$. This option is implemented in package **biglm** (Lumley, 2013). Function `biglm` uses only p^2 memory of p variables and the fitted object can be updated with more data using `update`. The package also provides an incremental computation of sandwich variance estimator by accumulating a $(p+1)^2 \times (p+1)^2$ matrix of products of X and Y without a second pass of the data.

In general, a numerical calculation needs an iterative algorithm in computation and, hence, multiple passes of the data are necessary. For example, a GLM fitting is often obtained through the iterated reweighted least squares (IRLS) algorithm. The `bigglm` function in package **biglm** implements the generic IRLS algorithm that can be applied to any specific data management system such as DBMS, **bigmemory**, or **ff**, provided that a function `data(reset = FALSE)` supplies the next chunk of data or zero-row data if there is no more, and `data(reset = TRUE)` resets to the beginning of the data for the next iteration. Specific implementation of the `data` function for object of class `big.matrix` and `ffdf` are provided in package **biganalytics** (Emerson & Kane, 2013a) and **ffbase** (Jonge *et al.*, 2014), respectively.

For any statistical analysis on big data making use of the data management system, one would need to implement the necessary numerical calculations like what package **biglm** does for GLM. The family of **bigmemory** provides a collection of functions for `big.matrix` objects: **biganalytics** for basic analytic and statistical functions, **bigtabulate** for tabulation operations (Emerson & Kane, 2013b), and **bigalgebra** for matrix operation with the BLAS and LAPACK libraries (Kane *et al.*, 2014). Some additional functions for `big.matrix` objects are available from other contributed packages, such as **bigpca** for principal component analysis and single-value decomposition (Cooper, 2014), and **bigrf** for random forest (Lim *et al.*, 2014). For `ff` objects, package **ffbase** provides basic statistical functions

(Jonge *et al.*, 2014). Additional functions for `ff` objects are provided in other packages, with examples including **biglars** for least angle regression and LASSO (Seligman *et al.*, 2011) and **PopGenome** for population genetic and genomic analysis (Pfeifer *et al.*, 2014).

If some statistical analysis, such as generalized estimating equations or Cox proportional hazards model, has not been implemented for big data, then one will need to modify the existing algorithm to implement it. As pointed out by Kane *et al.* (2013, p.5), this would open Pandora's box of recoding which is not a long-term solution for scalable statistical analyses; this calls for redesign of the next-generation statistical programming environment which could provide seamless scalability through file-backed memory-mapping for big data, help avoid the need for specialized tools for big data management, and allow statisticians and developers to focus on new methods and algorithms.

3.2   Breaking the Computing Power Barrier

3.2.1   Speeding Up

As a high level interpreted language, for which most of instructions are executed directly, R is infamously slow with loops. Some loops can be avoided by taking advantage of the vectorized functions in R or by clever vectorizing with some effort. When vectorization is not straightforward or loops are unavoidable, as in the case of MCMC, acceleration is much desired, especially for big data. The least expensive tool in a programmer's effort to speed up R code is to compile them to byte code with the **compiler** package, which was developed by Luke Tierney and is now part of base R. The byte code compiler translates the high-level R into a very simple language that can be interpreted by a very fast byte code interpreter, or virtual machine. Starting with R 2.14.0 in 2011, the base and recommended packages were pre-compiled into byte-code by default. Users' functions, expressions, scripts, and packages can be compiled for an immediate boost in speed by a factor of 2 to 5.

Computing bottlenecks can be implemented in a compiled language such as C/C++ or FORTRAN and interfaced to R through R's foreign language interfaces (R Core Team, 2014b, ch.5). Typical bottlenecks are loops, recursions, and complex data structures. Recent developments have made the interfacing with C++ much easier than it used to be (Eddelbuettel, 2013). Package **inline** (Sklyar *et al.*, 2013) provides functions that wrap

C/C++ (or FORTRAN) code as strings in R and takes care of compiling, linking, and loading by placing the resulting dynamically-loadable object code in the per-session temporary directory used by R. For more general usage, package **Rcpp** (Eddelbuettel *et al.*, 2011) provides C++ classes for many basic R data types, which allow straightforward passing of data in both directions. Package **RcpEigen** (Bates *et al.*, 2014) provides access to the high-performance linear algebra library Eigen for a wide variety of matrix methods, various decompositions and support of sparse matrices. Package **RcppArmadillo** (Eddelbuettel & Sanderson, 2014) connects R with Armadillo, a powerful templated linear algebra library which provides a good balance between speed and ease of use. Package **RInside** (Eddelbuettel & Francois, 2014) gives easy access of R objects from C++ by wrapping the existing R embedding API in C++ classes. The **Rcpp** project has revolutionized the integration of R with C++; it is now used by hundreds of R packages.

Diagnostic tools can help identify the bottlenecks in R code. Package **microbenchmark** (Mersmann, 2014) provides very precise timings for small pieces of source code, making it possible to compare operations that only take a tiny amount of time. For a collection of code, run-time of each individual operation can be measured with realistic inputs; the process is known as profiling. Function Rprof in R does the profiling, but the outputs are not intuitive to understand for many users. Packages **proftools** (Tierney & Jarjour, 2013) and **aprof** (Visser, 2014) provide tools to analyze profiling outputs. Packages **profr** (Wickham, 2014b), **lineprof** (Wickham, 2014c), and **GUIProfiler** (de Villar & Rubio, 2014) provide visualization of profiling results.

3.2.2   Scaling Up

The R package system has long embraced integration of parallel computing of various technologies to address the big data challenges. For embarrassingly parallelizable jobs such as bootstrap or simulation, where there is no dependency or communication between parallel tasks, many options are available with computer clusters or multicores. Schmidberger *et al.* (2009) reviewed the then state-of-the-art parallel computing with R, highlighting two packages for cluster use: **Rmpi** (Yu, 2002) which provides an R interface to the Message Passing Interface (MPI) in parallel computing; **snow** (Rossini *et al.*, 2007)

which provides an abstract layer with the communication details hidden from the end users. Since then, some packages have been developed and some discontinued. Packages **snowFT** (Sevcikova & Rossini, 2012a) and **snowfall** (Knaus, 2013) extend **snow** with fault tolerance and wrappers for easier development of parallel R programs. Package **multicore** (Urbanek, 2014) provides parallel processing of R code on machines with multiple cores or CPUs. Its work and some of **snow** have been incorporated into the base R package **parallel**, which was first included in R 2.14.0 in 2011. Package **foreach** (Revolution Analytics & Weston, 2014) allows general iteration over elements in a collection without any explicit loop counter. Using `foreach` loop without side effects facilitates executing the loop in parallel with different parallel mechanisms, including those provided by **parallel**, **Rmpi**, and **snow**. For massive data that exceed the computer memory, a combination of **foreach** and **bigmemory**, with shared-memory data structure referenced by multiple processes, provides a framework with ease of development and efficiency of execution (both in speed and memory) as illustrated by Kane *et al.* (2013). Package **Rdsm** provides facilities for distributed shared memory parallelism at the R level, and combined with **bigmemory**, it enables parallel processing on massive, out-of-core matrices.

The "Programming with Big Data in R" project (pbdR) enables high-level distributed data parallelism in R with easy utilization of large clusters with thousands of cores (Ostroucho 2012). Big data are interpreted quite literally to mean that a dataset requires parallel processing either because it does not fit in the memory of a single machine or because its processing time needs to be made tolerable. The project focuses on distributed memory systems where data are distributed across processors and communications between processors are based on MPI. It consists of a collection of R packages in a hierarchy. Package **pbdMPI** provides S4 classes to directly interface with MPI to support the Single Program Multiple Data (SPMD) parallelism. Package **pbdSLAP** serves as a mechanism to utilize a subset of functions of scalable dense linear algebra in ScaLAPACK (Blackford *et al.*, 1997), a subset of LAPACK routines redesigned with the SPMD style. Package **pbdBASE** contains a set of wrappers of low level functions in ScaLAPACK, upon which package **pbdMAT** builds to provide distributed dense matrix computing while preserving the friendly and familiar R syntax for these computations. Demonstrations on how to use these and other packages

from the pbdR are available in package **pbdDEMO**.

A recent, widely adopted open source framework for massive data storage and distributed computing is Hadoop (The Apache Software Foundation, 2014b). Its heart is an implementation of the MapReduce programming model first developed at Google (Dean & Gl 2008), which divides the data to distributed systems and computes for each group (the map step), and then recombines the results (the reduce step). It provides fault tolerant and scalable storage of massive datasets across machines in a cluster (White, 2011). The model suits perfectly the embarrassingly parallelizable jobs and the distributed file system helps break the memory boundary. McCallum & Weston (2011, ch.5–8) demonstrated three ways to combine Hadoop and R. The first is to submit R scripts directly to a Hadoop cluster, which gives the user the most control and the most power, but comes at the cost of a Hadoop learning curve. The second is a pure R solution via package **Rhipe**, which hides the communications to Hadoop from R users. The package (not on CRAN) is from the RHIPE project, which stands for R and Hadoop Integrated Programming Environment (Guha *et al.*, 2012). With **Rhipe**, data analysts only need to write R code for the map step and the reduce step (Guha *et al.*, 2012), and get the power of Hadoop without leaving R. The third approach targets specifically the Elastic MapReduce (EMR) at Amazon by CRAN package **segue** (Long, 2012), which makes it easy to use EMR as a parallel backend for `lapply`-style operations. A more recent approach is the open source RHadoop project, which is being actively developed by Revolution Analytics (Revolution Analytics, 2014). This project is a collection of R packages that allow users to manage and analyze data with Hadoop: **rhbase** provides functions for database management for the HBase distributed database, **rhdfs** provides functions for Hadoop distributed file system, **rmr** provides functions to Hadoop MapReduce functionality, **plymr** provides higher level data processing for structured data, and the most recent addition **ravro** provides reading and writing functions for files in `avro` format, an efficient data serialization system developed at Apache (The Apache Software Foundation, 2014a).

As multicores have become the standard setup for computers today, it is desirable to automatically make use of the cores in implicit parallelism without any explicit requests from the user. The experimental packages **pnmath** and **pnmath0** by Luke Tierney replace

a number of internal vector operations in R with alternatives that can take advantage of multicores (Tierney, 2009). For a serial algorithm such as MCMC, it is desirable to parallelize the computation bottleneck if possible, but this generally involves learning new computing tools and the debugging can be challenging. For instance, Yan *et al.* (2007) used the parallel linear algebra package (PLAPACK) (van de Geijn, 1997) for the matrix operations (especially the Cholesky decomposition) in a MCMC algorithm for Bayesian spatiotemporal geostatistical models, but the scalability was only moderate.

When random numbers are involved as in the case of simulation, extra care is needed to make sure the parallelized jobs run independent (and preferably reproducible) random-number streams. Package **rsprng** (Li, 2010) provides an interface to the Scalable Parallel Random Number Generators (SPRNG) (Mascagni & Srinivasan, 2000). Package **rlecuyer** (Sevcikova & Rossini, 2012b) provides an interface to the random number generator with multiple independent streams developed by L'Ecuyer *et al.* (2002), the ideas of which are also implemented in the base package **parallel**: make independent streams by separating a single stream with a sufficiently large number of steps apart. Package **doRNG** (Gaujoux, 2014) provides functions to perform reproducible parallel `foreach` loops, independent of the parallel environment and associated `foreach` backend.

From a hardware perspective, many computers have mini clusters of graphics processing units (GPUs) that can help with bottlenecks. GPUs are dedicated numerical processors that were originally designed for rendering three dimensional computer graphics. A GPU has hundreds of processor cores on a single chip and can be programmed to apply the same numerical operations on large data array. Suchard *et al.* (2010) investigated the use of GPUs in massively parallel massive mixture modeling, and showed better performance of GPUs than multicore CPUs, especially for larger samples. To reap the advantage, however, one needs to learn the related tools such as Compute Unified Device Architecture (CUDA), Open Computing Language (OpenCL), and so on, which may be prohibitive. An R package **gputools** (Buckner *et al.*, 2013) provides interface to NVidia CUDA toolkit and others.

4 Commercial Software

RRE is the core product of Revolution Analytics (formerly Revolution Computing), a company that provides R tools, support, and training. RRE focuses on big data, large scale multiprocessor (or high performance) computing, and multicore functionality. Massive datasets are handled via EMA and parallel EMA (PEMA) when multiprocessors or multicores are available. The commercial package **RevoScaleR** (Revolution Analytics, 2013) breaks the memory boundary by a special XDF data format that allows efficient storage and retrieval of data. Functions in the package (e.g., rxGlm for GLM fitting) know to work on a massive dataset one chunk at a time. The computing power boundary is also addressed — functions in the package can exploit multicores or computer clusters. Packages from the aforementioned open source project RHadoop developed by the company provide support for Hadoop. Other components in RRE allow high speed connection for various types of data sources and threading and inter-process communication for parallel and distributed computing. The same code works on small and big data, and on workstations, servers, clusters, Hadoop, or in the cloud. The single workstation version of RRE is free for academic use currently, and is used in the case study in Section 5.

SAS, one of the most widely used commercial software for statistical analysis, provides big data support through SAS High Performance Analytics. Massive datasets are approached by grid computing, in-database processing, in-memory analytics and connection to Hadoop. The SAS High Performance Analytics Products include statistics, econometrics, optimization, forecasting, data mining, and text mining, which, respectively, correspond to SAS products STAS, ETS, OR, high-performance forecasting, enterprise miner, and text miner (Cohen & Rodriguez, 2013).

IBM SPSS, the Statistical Product and Services Solution, provides big data analytics through SPSS Modeler, SPSS Analytic Server, SPSS Collaboration and Deployment Services, and SPSS Analytic Catalyst (IBM, 2014). SPSS Analytic Server is the foundation and it focuses on high performance analytics for data stored in Hadoop-based distributed systems. SPSS modeler is the high-performance data mining workbench, utilizing SPSS Analytic Server to leverage big data in Hadoop environments. Analysts can define analysis in a familiar and accessible workbench to conduct analysis modeling and scoring over

high volumes of varied data. SPSS Collaboration and Deployment Services helps manage analytical assets, automate processes and efficiently share results widely and securely. SPSS Analytic Catalyst is the automation of analysis that makes analytics and data more accessible to users.

MATLAB provides a number of tools to tackle the challenges of big data analytics (Inc., 2014). Memory mapped variables map a file or a proportion of a file to a variable in RAM; disk variables direct access to variables from files on disk; datastore allows access to data that do not fit into RAM. Their combination addresses the memory boundary. The computation power boundary is broken by intrinsic multicore math, GPU computing, parallel computing, cloud computing, and Hadoop support.

5 A Case Study

The airline on-time performance data from the 2009 ASA Data Expo (`http://stat-computi` is used as a case study to demonstrate a logistic model fitting with a massive dataset that exceeds the RAM of a single computer. The data is publicly available and has been used for demonstration with big data by Kane *et al.* (2013) and others. It consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. About 12 million flights were recorded with 29 variables. A compressed version of the pre-processed data set from the bigmemory project (`http://data.jstatsoft.org/v55/i14/Airline.tar.bz2`) is approximately 1.7GB, and it takes 12GB when uncompressed.

The response of the logistic regression is late arrival which was set to 1 if a flight was late by more than 15 minutes and 0 otherwise. Two binary covariates were created from the departure time: night (1 if departure occurred between 8pm and 5am) and weekend (1 if departure occurred on weekends and 0 otherwise). Two continuous covariates were included: departure hour (DepHour, range 0 to 24) and distance from origin to destination (in 1000 miles). In the raw data, the departure time was an integer of the HHmm format. It was converted to minutes first to prepare for DepHour. Three methods are considered in the case study: 1) combination of `bigglm` with package **bigmemory**; 2) combination of `bigglm` with package **ff**; and 3) the academic, single workstation version of RRE. The

Table 1: Timing results (in seconds) for reading in the whole 12GB data, transforming to create new variables, and fitting the logistic regression with three methods: **bigmemory**, **ff**, and RRE.

	Reading	Transforming	Fitting
bigmemory	968.6	105.5	1501.7
ff	1111.3	528.4	1988.0
RRE	851.7	107.5	189.4

default settings of **ff** were used. Before fitting the logistic regression, the 12GB raw data needs to be read in from the csv format, and new variables needs to be generated. This leads to a total of $120, 748, 239$ observations with no missing data. The R scripts for the three methods are in the supplementary materials for interested readers.

The R scripts were executed in batch mode on a 8-core machine running CenOS (a free Linux distribution functionally compatible with Red Hat Enterprise Linux which is officially supported by RRE), with Intel Core i7 2.93GHz CPU, and 16GB memory. Table 1 summarizes the timing results of reading in the whole 12GB data, transforming to create new variables, and fitting the logistic regression with the three methods. The chunk sizes were set to be 500,000 observations for all three methods. For RRE, this was set when reading in the data to the XDF format; for the other two methods, this was set at the fitting stage using function `bigglm`. Under the current settings, RRE has a clear advantage in fitting with only 8% of the time used by the other two approaches. This is a result of the joint force of its using all 8 cores implicitly and efficient storage and retrieval of the data; the XDF version of the data is about $1/10$ of the size of the external files saved by **bigmemory** or **ff**. Using **bigmemory** and using **ff** in `bigglm` had very similar performance in fitting the logistic regression, but the former took less time in reading, and significantly less time (only about $1/5$) in transforming variables of the latter. The **bigmemory** method was quite close to the RRE method in the reading and the transforming tasks. The **ff** method took longer in reading and transforming than the **bigmemory** method, possibly because it used much less memory.

The results of the logistic regression are identical from all methods, and are summarized

Table 2: Logistic regression results for late arrival.

	Estimate	Std. Error ($\times 10^4$)
(Intercept)	-2.985	9.470
DepHour	0.104	0.601
Distance	0.235	4.032
Night	-0.448	8.173
Weekend	-0.177	5.412

Table 3: Time results (in seconds) for parallel computing quantiles of departure delay for each day of the week with 1 to 8 cores using **foreach**.

	1	2	3	4	5	6	7	8
bigmemory	22.1	11.2	7.8	6.9	6.2	6.3	6.4	6.8
ff	21.4	11.0	7.1	6.7	5.8	5.9	6.1	6.8

in Table 2. Flights with later departure hour or longer distance are more likely to be delayed. Night flights or weekend flights are less likely to be delayed. Given the huge sample size, all coefficients were highly significant. It is possible, however, that p-values can still be useful. A binary covariate with very low rate of event may still have an estimated coefficient with a not-so-low p-value (Schifano *et al.*, 2014), an effect only estimable with big data.

As an illustration of **foreach** for embarrassingly parallel computing, the example in Kane *et al.* (2013) is expanded to include both **bigmemory** and **ff**. The task is to find three quantiles (0.5, 0.9, and 0.99) of departure delays for each day of the week; that is, 7 independent jobs can run on 7 cores separately. To make the task bigger, each job was set to run twice. The resulting 14 jobs were parallelized with `foreach` on the same Linux machine using 1 to 8 cores for the sake of illustration. The R script is included in the supplementary materials. The timing results are summarized in Table 3. There is little difference between the two implementations. When there is no communication overhead, with 14 jobs one would expect the run time to reduce to 1/2, 5/14, 4/14, 3/14, 3/14, 2/14, and 2/14, respectively, with 2, 3, 4, 5, 6, 7 and 8 cores. The impact of communication cost is obvious in Table 3. The time reduction is only closer to the expectation in the ideal case

when the number of cores is smaller.

6   Discussion

The scope of this review is limited to standard statistical analysis with big data that exceed the memory and computing capacity of a single computer. Albeit under-appreciated by the general public or even mainstream academic community, computational statisticians have made respectable progress in extending standard statistical analysis to big data, with the most notable achievements in the open source R community. Packages **bigmemory** and **ff** make it possible in principle to implement any statistical analysis with their data structure. Nonetheless, for anything that has not been already implemented (e.g., survival analysis, GEE, mixed effects model, etc.), one would need to implement an EMA version of the computation task, which may not be straightforward and may involve some steep learning curves. Hadoop allows easy extension of algorithms that do not require multiple passes of the data, but such analyses are mostly descriptive. An example is visualization, an important tool in exploratory analysis. With big data, the bottleneck is the number of pixels in the screen. The bin-summarize-smooth framework for visualization of large data of Wickham (2014a) with package **bigvis** (Wickham, 2013) may be adapted to work with Hadoop.

Big data present challenges much further beyond the territory of classic statistics, requiring joint workforce with domain knowledge, computing skills, and statistical thinking (Yu, 2014). Statisticians have much to contribute to both the intellectual vitality and the practical utility of big data, but will have to expand their comfort zone to engage high-impact, real world problems which are often less structured or with ambiguity (Jordan & Lin, 2014). Examples are to provide structure for poorly defined problems, or to develop methods/models for new types of data such as image or network. As suggested by Yu (2014), to play a critical role in the arena of big data or own data science, statisticians need to work on real problems and relevant methodology and theory will follow naturally.

Acknowledgment

Supplementary Materials

Four R scripts (and their outputs), along with a descriptive README file are provided for the case study. The first three are the logistic regression with, respectively, combination of **bigmemory** with `bigglm` (`bigmemory.R`), combination of **ff** with `bigglm` (`ff.R`), and RRE (`RevR.R`); their output files have `.Rout` extensions. The first two run with R, while the third one needs RRE. The fourth script is for the parallel computing with **foreach** combined with **bigmemory** and **ff**, respectively.

References

ADLER, D., GLSER, C., NENADIC, O., OEHLSCHLGEL, J. & ZUCCHINI, W. (2014). *ff: Memory-efficient Storage of Large Data on Disk and Fast Access Functions*. URL `http://CRAN.R-project.org/package=ff`. R package version 2.2-13.

BATES, D., FRANCOIS, R. & EDDELBUETTEL, D. (2014). *RcppEigen: Rcpp Integration for the Eigen Templated Linear Algebra Library*. URL `http://CRAN.R-project.org/package=Rcppeigen`. R package version 0.3.2.2.0.

BICKEL, P. J., GÖTZE, F. & VAN ZWET, W. R. (1997). Resampling fewer than n observations: Gains, losses, and remedies for losses. *Statistica Sinica* **7**, 1–31.

BLACKFORD, L. S., CHOI, J., CLEARY, A., D'AZEVEDO, E., DEMMEL, J., DHILLON, I., DONGARRA, J., HAMMARLING, S., HENRY, G., PETITET, A., STANLEY, K.,

WALKER, D. & WHALEY, R. C. (1997). *ScaLAPACK Users' Guide*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

BUCKNER, J., SELIGMAN, M. & WILSON, J. (2013). *gputools: A few GPU Enabled Functions*. URL `http://CRAN.R-project.org/package=gputools`. R package version 0.28.

CHAMBERS, J. (2014). Interfaces, efficiency and big data. 2014 UseR! International R User Conference.

CHEN, X. & XIE, M.-G. (2014). A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica* Forthcoming.

COHEN, R. & RODRIGUEZ, R. (2013). High performance statistical modeling. Technical Report 401–2013, SAS Global Forum.

COOPER, N. (2014). *bigpca: PCA, Transpose and Multicore Functionality for big.matrix Objects*. URL `http://CRAN.R-project.org/package=bigpca`. R package version 1.0.

DAVIDIAN, M. (2013). Aren't we data science. *Amstat News* **433**, 3–5.

DE VILLAR, F. & RUBIO, A. (2014). *GUIProfiler: Profiler Graphical User Interface*. URL `http://CRAN.R-project.org/package=GUIProfiler`. R package version 0.1.2.

DEAN, J. & GHEMAWAT, S. (2008). MapReduce: Simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113.

DIEBOLD, F. X. (2012). A personal perspective on the origin(s) and development of "big data": The phenomenon, the term, and the discipline, second version. Pier working paper archive, Penn Institute for Economic Research, Department of Economics, University of Pennsylvania.

EDDELBUETTEL, D. (2013). *Seamless R and C++ Integration with Rcpp*. Springer.

EDDELBUETTEL, D. (2014). CRAN task view: High-performance and parallel computing with R. Version 2014-09-23.

EDDELBUETTEL, D. & FRANCOIS, R. (2014). *RInside: C++ Classes to Embed R in C++ Applications*. URL `http://CRAN.R-project.org/package=RInside`. R package version 0.2.11.

EDDELBUETTEL, D., FRANÇOIS, R., ALLAIRE, J., CHAMBERS, J., BATES, D. & USHEY, K. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software* **40**(8), 1–18.

EDDELBUETTEL, D. & SANDERSON, C. (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis* **71**, 1054–1063.

EFRON, B. (1979). Bootstrap methods: Another look at the Jackknife. *The annals of Statistics* **7**(1), 1–26.

EMERSON, J. W. & KANE, M. J. (2012). Don't drown in the data. *Significance* **9**(4), 38–39.

EMERSON, J. W. & KANE, M. J. (2013a). *biganalytics: A Library of Utilities for big.matrix Objects of Package bigmemory*. URL `http://CRAN.R-project.org/package=biganalytics`. R package version 1.1.1.

EMERSON, J. W. & KANE, M. J. (2013b). *bigtabulate: table-, tapply-, and Split-like Functionality for Matrix and big.matrix Objects*. URL `http://CRAN.R-project.org/package=bigtabulate`. R package version 1.1.2.

ENEA, M. (2014). *speedglm: Fitting Linear and Generalized Linear Models to Large Data Sets*. URL `http://CRAN.R-project.org/package=speedglm`. R package version 0.2-1.0.

FAN, J., HAN, F. & LIU, H. (2013). Challenges of big data analysis. *arXiv preprint arXiv:1308.1479* .

FAN, J. & LV, J. (2011). Non-concace penalized likelihood with NP-dimensionality. *IEEE Transactions on Information Theory* **57**, 5467–5484.

Fan, W. & Bifet, A. (2013). Mining big data: Current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter* **14**(2), 1–5.

Gaujoux, R. (2014). *doRNG: Generic Reproducible Parallel Backend for foreach Loops*. URL `http://CRAN.R-project.org/package=doRNG`. R package version 1.6.

Grothendieck, G. (2014). *sqldf: Perform SQL Selects on R Data Frames*. URL `http://CRAN.R-project.org/package=sqldf`. R package version 0.4-7.1.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B. & Cleveland, W. S. (2012). Large complex data: Divide and Recombine (D&R) with RHIPE. *Stat* **1**(1).

Hollensbe, E. (2009). *RDBI: Low-level Database Access Re-imagined*. URL `https://github.com/RDBI/rdbi`.

IBM (2014). Apply SPSS analytics technology to big data.

Inc., T. M. (2014). Big data with MATLAB.

Jonge, E. d., Wijffels, J. & van der Laan, J. (2014). *ffbase: Basic Statistical Functions for Package ff*. URL `http://CRAN.R-project.org/package=ffbase`. R package version 0.11.3.

Jordan, J. M. & Lin, D. K. J. (2014). Statistics for big data: Are statisticians ready for big data? *International Chinese Statistical Association Bulletin* **26**, 59–66.

Jordan, M. I. (2013). On statistics, computation and scalability. *Bernoulli* **19**(4), 1378–1390.

Kane, M. J., Emerson, J. & Weston, S. (2013). Scalable strategies for computing with massive data. *Journal of Statistical Software* **55**(14), 1–19.

Kane, M. J., Lewis, B. & Emerson, J. W. (2014). *bigalgebra: BLAS Routines for Native R Matrices and big.matrix Objects*. URL `http://CRAN.R-project.org/package=bigalgebra`. R package version 0.8.4.

KLEINER, A., TALWALKAR, A., SARKAR, P. & JORDAN, M. I. (2014). A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **76**(4), 795–816.

KNAUS, J. (2013). *snowfall: Easier Cluster Computing (Based on snow).* URL `http://CRAN.R-project.org/package=snowfall`. R package version 1.84-6.

LANEY, D. (2001). 3D data management: Controlling data volume, velocity, and variety. Research note, META Group.

L'ECUYER, P., SIMARD, R., CHEN, E. J. & KELTON, W. D. (2002). An object-oriented random-number package with many long streams and substreams. *Operations research* **50**(6), 1073–1075.

LI, N. (2010). *R Interface to SPRNG (Scalable Parallel Random Number Generators).* URL `http://CRAN.R-project.org/package=rsprng`. R package version 1.0.

LIANG, F., CHENG, Y., SONG, Q., PARK, J. & YANG, P. (2013). A resampling-based stochastic approximation method for analysis of large geostatistical data. *Journal of the American Statistical Association* **108**(501), 325–339.

LIANG, F. & KIM, J. (2013). A bootstrap Metropolis–Hastings algorithm for Bayesian analysis of big data. Tech. rep., Department of Statistics, Texas A & M University.

LIM, A., BREIMAN, L. & CUTLER, A. (2014). *bigrf: Big Random Forests: Classification and Regression Forests for Large Data Sets.* URL `http://CRAN.R-project.org/package=bigrf`. R package version 0.1-11.

LIN, N. & XI, R. (2011). Aggregated estimating equation estimation. *Statistics and Its Interface* **4**, 73–83.

LONG, J. (2012). *An R Language Segue into Parallel Processing on Amazon's Web Services.* URL `https://code.google.com/p/segue/`. R package version 0.05.

LUMLEY, T. (2013). *biglm: Bounded Memory Linear and Generalized Linear Models.* URL `http://CRAN.R-project.org/package=biglm`. R package version 0.9-1.

MA, P., MAHONEY, M. W. & YU, B. (2013). A statistical perspective on algorithmic leveraging. *arXiv preprint arXiv:1306.5362* .

MA, P. & SUN, X. (2014). Leveraging for big data regression. *WIREs Computational Statistics* .

MANYIKA, J., CHUI, M., BROWN, B., BUGHIN, J., DOBBS, R., ROXBURGH, C. & BYERS, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. Tech. rep., McKinsey Global Institute.

MASCAGNI, M. & SRINIVASAN, A. (2000). Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software (TOMS)* **26**(3), 436–461.

MASHEY, J. (1998). Big data and the next wave of InfraStress. Usenix.org.

MCCALLUM, Q. E. & WESTON, S. (2011). *Parallel R: Data Analysis in the Distributed World.* O'Reilly Media.

MEINSHAUSEN, N. & BUHLMANN, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**, 417–473.

MERSMANN, O. (2014). *microbenchmark: Accurate Timing Functions.* URL `http://CRAN.R-project.org/package=microbenchmark`. R package version 1.4-2.

MILLER, A. J. (1992). Algorithm AS 274: Least squares routines to supplement those of gentleman. *Applied Statistics* , 458–478.

OSTROUCHOV, G., CHEN, W.-C., SCHMIDT, D. & PATEL, P. (2012). Programming with big data in R. URL `http://r-pbd.org/`.

PENG, R. D. (2006). Interacting with data using the filehash package. *R News* **6**(4), 19–24.

PFEIFER, B., WITTELSBUERGER, U., RAMOS-ONSINS, S. E. & LERCHER, M. J. (2014). PopGenome: An efficient swiss army knife for population genomic analyses in R. *Molecular Biology and Evolution* **31**, 1929–1936.

POLITIS, D. N., ROMANO, J. P. & WOLF, M. (1999). Subsampling .

R CORE TEAM (2014a). *R: A Language and Environment for Statistical Computing.* Vienna, Austria.

R CORE TEAM (2014b). *Writing R Extensions.* Vienna, Austria.

REVOLUTION ANALYTICS (2013). *RevoScaleR 7.0 User's Guide.* Mountain View, CA.

REVOLUTION ANALYTICS (2014). RHadoop. GitHub Project Wiki.

REVOLUTION ANALYTICS & WESTON, S. (2014). *foreach: foreach Looping Construct for R.* URL `http://CRAN.R-project.org/package=foreach`. R package version 1.4.2.

RICKERT, J. (2013). Statisticians: An endangered species?

RODRIGUEZ, R. (2012). Big data and better data. *Amstat News* **420**, 3–4.

ROSSINI, A. J., TIERNEY, L. & LI, N. (2007). Simple parallel statistical computing in R. *Journal of Computational and Graphical Statistics* **16**(2), 399–420.

RUDIN, C., DUNSON, D., IRIZARRY, R., JI, H., LABER, E., LEEK, J., MCCORMICK, T., ROSE, S., SCHAFER, C., VAN DER LAAN, M., WASSERMAN, L. & XUE, L. (2014). Discovery with data: Leveraging statistics with computer science to transform science and society. White paper, American Statistical Association.

SCHENKER, N., DAVIDIAN, M. & RODRIGUEZ, R. (2013). The ASA and big data. *Amstat News* **432**, 3–4.

SCHIFANO, E. D., WU, J., WANG, C., YAN, J. & CHEN, M.-H. (2014). Online updating of statistical inference in the big data setting. Tech. Rep. 14-22, Department of Statistics, University of Connecticut, Storrs, Connecticut.

SCHMIDBERGER, M., MORGAN, M., EDDELBUETTEL, D., YU, H., TIERNEY, L. & MANSMANN, U. (2009). State of the art in parallel computing with R. *Journal of Statistical Software* **31**(1), 1–27.

SELIGMAN, M., FRALEY, C. & HESTERBERG, T. (2011). *biglars: Scalable Least-angle Regression and Lasso*. URL http://CRAN.R-project.org/package=biglars. R package version 1.0.2.

SEVCIKOVA, H. & ROSSINI, A. J. (2012a). *snowFT: Fault Tolerant Simple Network of Workstations*. URL http://CRAN.R-project.org/package=snowFT. R package version 1.3-0.

SEVCIKOVA, H. & ROSSINI, T. (2012b). *rlecuyer: R Interface to RNG with Multiple Streams*. URL http://CRAN.R-project.org/package=rlecuyer. R package version 0.3-3.

SHAW, J. (2014). Why big data is a big deal. Harvard Magazine.

SINGH, K., XIE, M. & STRAWDERMAN, W. (2005). Combining information from independent sources through confidence distributions. *Annals of Statistics* , 159–183.

SKLYAR, O., MURDOCH, D., SMITH, M., EDDELBUETTEL, D. & FRANCOIS, R. (2013). *inline: Inline C, C++, Fortran Function Calls from R*. URL http://CRAN.R-project.org/package=inline. R package version 0.3.13.

SNIJDERS, C., MATZAT, U. & REIPS, U.-D. (2012). Big data: Big gaps of knowledge in the field of internet science. *International Journal of Internet Science* **7**(1), 1–5.

SUCHARD, M. A., WANG, Q., CHAN, C., FRELINGER, J., CRON, A. & WEST, M. (2010). Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *Journal of Computational and Graphical Statistics* **19**(2).

THE APACHE SOFTWARE FOUNDATION (2014a). Apache avro. URL https://avro.apache.org/.

THE APACHE SOFTWARE FOUNDATION (2014b). Apache hadoop. URL http://hadoop.apache.org/.

TIERNEY, L. (2009). Code analysis and parallelizing vector operations in R. *Computational Statistics* **24**(2), 217–223.

TIERNEY, L. & JARJOUR, R. (2013). *proftools: Profile Output Processing Tools for R.* URL http://CRAN.R-project.org/package=proftools. R package version 0.1-0.

URBANEK, S. (2014). *multicore: A Stub Package to Ease Transition to 'parallel'.* URL http://CRAN.R-project.org/package=multicore. R package version 0.2.

VAN DE GEIJN, R. A. (1997). *Using PLAPACK.* Cambridge, MA: The MIT Press.

VISSER, M. D. (2014). *aprof: Amdahl's Profiler, Directed Optimization Made Easy.* URL http://CRAN.R-project.org/package=aprof. R package version 0.2.4.

VITTER, J. S. (2001). External memory algorithms and data structures: Dealing with massive data. *ACM Computing surveys (CsUR)* **33**(2), 209–271.

WHITE, T. (2011). *Hadoop: The Definitive Guide.* O'Reilly Media, Inc., 2nd ed.

WICKHAM, H. (2013). *Tools for Visualisation of Big Data Sets.* URL https://github.com/hadley/bigvis. R package version 0.1.

WICKHAM, H. (2014a). Bin-summarise-smooth: A framework for visualising large data.

WICKHAM, H. (2014b). *profr: An Alternative Display for Profiling Information.* URL http://CRAN.R-project.org/package=profr. R package version 0.3.1.

WICKHAM, H. (2014c). *Visualise Line Profiling Results in R.* URL https://github.com/hadley/lineprof. R package version 0.1.

WICKHAM, H., JAMES, D. A. & FALCON, S. (2014). *RSQLite: SQLite Interface for R.* URL http://CRAN.R-project.org/package=RSQLite. R package version 1.0.0.

XIE, M., SINGH, K. & STRAWDERMAN, W. (2011). Confidence distributions and a unifying framework for meta-analysis. *Journal of the American Statistical Association* **106**, 320–333.

Yan, J., Cowles, M. K., Wang, S. & Armstrong, M. P. (2007). Parallelizing
MCMC for Bayesian spatiotemporal geostatistical models. *Statistics and Computing*
17(4), 323–335.

Yu, B. (2014). Let us own data science. IMS Bulletin Online, V. 43 (7).

Yu, H. (2002). Rmpi: Parallel statistical computing in R. *R News* **2**(2), 10–14.