Divide & Recombine for Large Complex Data (a.k.a. Big Data): Goals

Provide the data analyst with statistical methods and a computational environment that enable deep study of large complex data. This requires application of analytic methods to the data at their finest level of granularity, and not just to summary statistics of the detailed data. This must include visualization methods.

The analyst uses, at the front end, an interactive language for data analysis that is powerful, so analysis can be tailored to the data and not just canned, and that enables highly time-efficient programming with the data.

At the back end, a distributed database and parallel compute engine running on a cluster that makes computation feasible and practical, and is easily addressable from within the language. The back end must provide a mechanism for fair sharing of the cluster resource by multiple analysts.

Within the environment, access to the 1000s of methods of statistics, machine learning, and visualization.

Write software packages that enable communication between front and back so that a variety of back-ends can be readily plugged in.

Use the system continuously to analyze large complex data in settings where results are judged by subject matter findings. This serves as a heat engine for generating new research ideas and for testing them.

D&R Statistics

The D&R Framework

Two types of analytic methods applied to subsets are treated differently

1. Number-category methods
   - outputs are numeric and categorical
   - e.g., logistic regression

2. Visualization methods
   - outputs are plots
   - e.g., scatterplot matrices

The D&R Framework

Division

- a **division method** divides the data into subsets
- a division persists and is used for many analytic methods

Number-category analytic methods are applied to each of the subsets

- no communication between the computations
- **embarrassingly parallel**

Visualization analytic methods are applied to each subset in a sample of subsets

- typically too many to look at plots for all subsets
- compute between-subset variables with one value per subset to enable rigorous sampling plans
- sampling plans: representative, focused, cognostics thanks to JWT

For each analytic method

- **recombination** method: subset outputs recombined, providing the D&R result for the analytic method
- often has a component of **parallel computation**
- there are many potential recombination methods, individually for analytic methods, and for classes of analytic methods

N.B. Computationally, this is a very simple.

Conditioning-Variable Division

In very many cases, it is natural to break up the data based on the subject matter in a way that would be done whatever the size

Data are embarrassingly divisible

The major division method used in our own analyses

Example

- 25 years of 100 daily financial variables for 10^5 banks in the U.S.
- division by bank
- bank is a conditioning variable

There can be multiple conditioning variables that form the division

Conditioning-variable division has already been widely used in statistics, machine learning, and visualization

Replicate Division

Observations are seen as exchangeable, with no conditioning variables considered

Division methods based on statistical matters, not the subject matter as in conditioning-variable division

Replicate Division

Consider logistic regression

- $n = 2^{30}$ observations = 1 gigaob
- $p = 2^7 - 1 = 127$ explanatory variables, all numeric
- one response of 0's and 1's
- $v = p + 1 = 2^7 = 128$ variables altogether
- memory size of each value is 8 bytes
- memory size of dataset is 2^{40} bytes = 1 terabyte

Suppose each subset has m observations, where $r = n/m$ is an integer

r subsets, so r logistic regressions

Outputs: r p-vectors of estimates of regression coefficients and associated statistical information

Give the outputs to a recombination method

Statistical Accuracy for Replicate Division

The statistical division and recombination methods have an immense impact on the accuracy of the D&R result

Examples for the logistic regression above

- division method: random
- recombination method: weighted by the inverses of estimates of the covariance matrices of the subset regression coefficient estimates

These are obvious choices; we know we can do better

AND, we need along with the final D&R estimates, characterizations of the statistical accuracy

We can do this, too, right now for the class of analytic methods with a certain structure like that of logistic regression

Statistical Accuracy for Replicate Division

The D&R result is not the same as that of the application of the method directly to all of the data

D&R research in statistical theory seeks to maximize the accuracy of D&R results

The statistical accuracy of the D&R result is typically less that that of the direct all-data result

Our results so far show that this is a small penalty to pay for the very simple, fast D&R computation that touches subsets just once for each analytic method.

The Big Question

Isn’t this just MapReduce?

Answer

Our D&R statistical work reveals how to best “chunk” the data, and how to best put things back together.

D&R Computational Environment

D&R Computational Environment: Front End

R

Elegant design makes programming with the data very efficient

Very powerful, allowing the analyst to readily tailor the analysis to the data

Saves the analyst time, which is more important than computer time

Access to 1000s of analytic methods of statistics, machine learning, and visualization

Very large supporting community

Big user community: best estimate is about 2 million

Back End

Hadoop running on a cluster

Distributed file system (HDFS)

Distributed parallel compute engine (MapReduce)

What the Analyst Specifies in R

D[dr], A[dr], and R[dr] Computations

D[dr]

- division method to divide the data into subsets using a division method
- structure of the R objects that hold the data and are written to disk

A[dr]

- analytic methods applied to each subset (number-category) or to each in a sample (visualization)
- structure of the R objects that hold the outputs

R[dr]

- for each applied analytic method, the recombination method and the structure of the R objects that hold the D&R result

What Hadoop Does with the Analyst R Commands

D[dr]

Computes subsets, forms R objects that contain them, and writes the R objects across the nodes of the cluster into the HDFS

A[dr]

Applies the analytic method to subsets in parallel on the cores of the cluster with no communication among subset computations, and if there is no accompanying recombination, writes the output to the HDFS

R[dr]

Takes outputs of the A[dr] computations and carries out the recombination method across them with the computation, and writes the results to the HDFS

The Hadoop Scheduler Enables Sharing

A[dr] computations run by mappers, each with a core

R[dr] computations run by reducers, each with a core

D[dr] typically uses mappers and reducers

These mappers and reducers are running micro-computations

If there is a single analyst, when one micro-computation ends, Hadoop schedules another

If analyst 2 appears, Hadoop begins scheduling cores to the micro-computations of analyst 2 as those for analyst 1 end until usage is fair

This is a beautiful thing

Three D&R Software Packages Between Front and Back Ends

What we want

- communication between R and `Hadoop`
- make D&R programming easy
- protect the analyst from the details of Hadoop computation
- design to make easy connection to other back ends

`RHIPE`

`datadr`

`Trelliscope`

The whole environment

```
linux, Hadoop, protocol buffers, datadr,
Trelliscope RHIPE, R
```

It is all open source

RHIPE: The R and Hadoop Integrated Programming Environment

Merger of R and `Hadoop` that uses `Hadoop` streaming

Pronounced: hree-pay′

“In a moment” in Greek

Analyst gives R code for D[dr], A[dr], and R[dr] computations to `RHIPE` R functions, which then manage communication with Hadoop

First written by Saptarshi Guha in 2010 while a Purdue Stat PhD student

An R package available on Github (not CRAN)

Needs `protocol buffers`

It is all open source

datadr

Provides a simpler interface for

- division
- applying the analytic method
- recombination
- other data operations such as sample, filter, and join

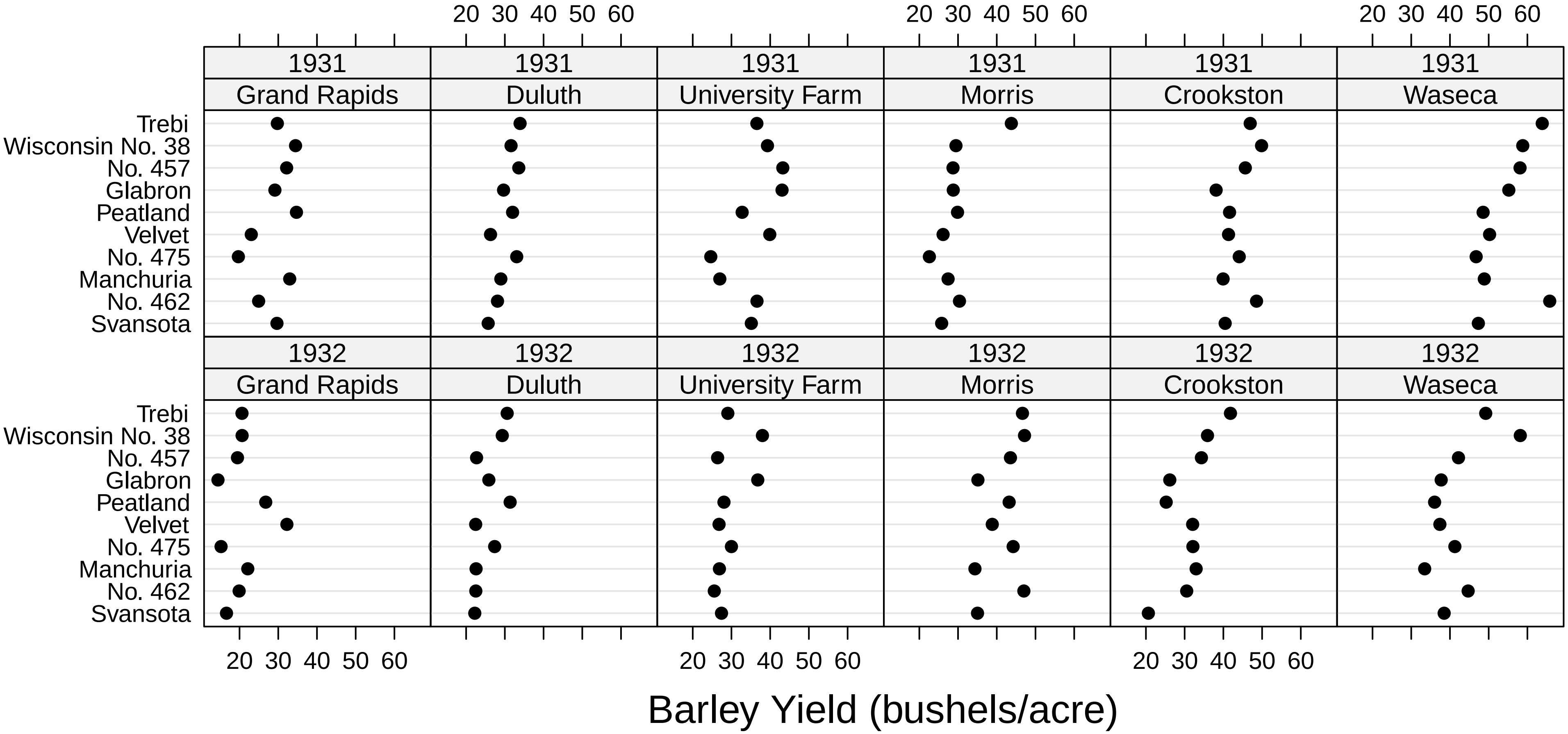
First written by Ryan Hafen at PNNL

Comes with a generic MapReduce interface

- fundamental data structure is key-value pairs
- designed to be easily extensible to any distributed key-value store and MapReduce system
- can support key-value pairs stored in memory, on local disk, or on an HDFS using `RHIPE`

Trelliscope

The trellis display visualization framework, implemented in R as the lattice graphics package, is very powerful because it is based on conditioning-variable division, good for data big and small.



Trelliscope

Extends trellis display to large complex data

Like `datadr`, is between `R` and `RHIPE`

Display panels have sampled subsets from rigorous sampling plans as described above

First written by Ryan Hafen at PNNL

It is open source

The Target Audience for the D&R Methods and Environment

D&R cannot solve all problems of large complex data

Who might benefit?

People who carry out deep analysis of data

Collectively around the world, they use the 1000s of methods of statistics, machine learning, and statistics

For the current D&R computational environment, R is our front end, so R users are a prime audience

The Target Audience for the D&R Methods and Environment

“large” or “big” by themselves are not good terms for what we face

Complexity is a critical factor, in some cases, the real challenge

It just happens that size and complexity are positively correlated

A 1 TB dataset can be dramatically more difficult than a 10 TB dataset

Size-complexity at which analysis becomes a serious problem without a large complex data solution, varies immensely across analysts

Main reason is that the computing resources available to analysts varies immensely

The Target Audience for the D&R Methods and Environment

We developed D&R for “large complex” data to scale up as the power of the hardware goes up

However, we believe it can scale down, too, and significantly help those with limited resources

In our own work at Purdue we now have 3 clusters on which our D&R team does data analysis

Pushing pretty far down, we built a 2-node cluster, each node with

- Dual 2.33GHz 4-core Intel(R) Xeon(R) E5410 processors (8 cores)
- 32 GB memory
- 2TB disk in SAS-RAID
- 1 Gbps Ethernet interconnect

We are looking out for the little guy, too