A Quick Guide for the pbdDEMO Package

Drew Schmidt[1], **Wei-Chen Chen**[2], **George Ostrouchov**[1,2], **Pragneshkumar Patel**[1]

[1]Remote Data Analysis and Visualization Center,
University of Tennessee,
Knoxville, TN, USA
[2]Computer Science and Mathematics Division,
Oak Ridge National Laboratory,
Oak Ridge, TN, USA

Contents

Acknowledgement

Ostrouchov, Schmidt, and Patel were supported in part by the project "NICS Remote Data Analysis and Visualization Center" funded by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center. Chen and Ostrouchov were supported in part by the project "Visual Data Exploration and Analysis of Ultra-large Climate Data" funded by U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725.

This work used resources of National Institute for Computational Sciences at the University of Tennessee, Knoxville, which is supported by the Office of Cyberinfrastructure of the U.S. National Science Foundation under Award No. ARRA-NSF-OCI-0906324 for NICS-RDAV center. This work also used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This work used resources of the Newton HPC Program at the University of Tennessee, Knoxville.

We also thank Brian D. Ripley, Kurt Hornik, and Uwe Ligges from the R Core Team for discussing package release issues and helping us solve portability problems on different platforms.

Warning: This document is written to explain the main functions of **pbdDEMO** (Schmidt *et al.* 2012b), version 0.1-0. Every effort will be made to ensure future versions are consistent with these instructions, but features in later versions may not be explained in this document.

Information about the functionality of this package, and any changes in future versions can be found on website: "Programming with Big Data in R" at http://r-pbd.org/.

1. Introduction

This vignette is to explain some pbdR (Ostrouchov *et al.* 2012) examples which are higher level applications and may be commonly found in basic Statistics. The purposes is to show how to reuse the pre-exist functions, and quickly solve problems in an efficient way. The functions built for examples may not be exactly same idea of original R (R Core Team 2012) functions, but can be adjusted in similar wa. You are very welcome to use these as templates and rewrite your own functions or packages.

1.1. Installation and Quick Start

One can download **pbdDEMO** from CRAN at http://cran.r-project.org, and the intallation can be done with the following commands

Shell Command

```
tar zxvf pbdDEMO_0.1-0.tar.gz
R CMD INSTALL pbdDEMO
```

2. Timing

Throughout ...

Part I

Ad Hoc Methods

3. Statistics Examples

This section introduces four simple examples and explains a little about distributed data computing. These implemented functions are selected from the Cookbook of HPSC website **??** at http://thirteen-01.stat.iastate.edu/snoweye/hpsc/?item=cookbook. Please see more details there.

Warning: We presume that readers have idea about SPMD programming. If not, please read pbdMPI's vignette Chen *et al.* (2012b) first. If possible, readers are encouraged to run the demo of pbdMPI package and go through the code step by step.

3.1. Sample Mean and Sample Variance

Suppose $\boldsymbol{x} = \{x_1, x_2, \ldots, x_N\}$ are observed samples, and N is very large. We can distribute $\boldsymbol{x}$ in 2 processors, and each processor roughly takes half of data. One simple way to compute sample mean $\bar{x}$ and sample variance s_x is based on the formulas:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$= \sum_{n=1}^{N} \frac{x_n}{N}$$

and

$$s_x = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2$$

$$= \frac{1}{N-1}\sum_{n=1}^{N} x_n^2 - \frac{2\bar{x}}{N-1}\sum_{n=1}^{N} x_n + \frac{1}{N-1}\sum_{n=1}^{N} \bar{x}^2$$

$$= \sum_{n=1}^{N}\left(\frac{x_n^2}{N-1}\right) - \frac{N\bar{x}^2}{N-1}$$

The demo command is

Shell Command

```
### At the shell prompt, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e "demo(sample_stat,'pbdDEMO',ask=F,echo=F)"
```

The demo `sample_stat` generates fake data on 2 processors, then utilize `mpi.stat` function as

R Code

```
mpi.stat <- function(x.spmd){
  ### For mean(x).
  N <- allreduce(length(x.spmd), op = "sum")
  bar.x.spmd <- sum(x.spmd / N)
  bar.x <- allreduce(bar.x.spmd, op = "sum")

  ### For var(x).
```

```
  s.x.spmd <- sum(x.spmd^2 / (N - 1))
  s.x <- allreduce(s.x.spmd, op = "sum") - bar.x^2 * (N / (N -
     1))

  list(mean = bar.x, s = s.x)
} # End of mpi.stat().
```

where `allreduce` in **pbdMPI** (Chen *et al.* 2012a) can be utilized in this examples to aggregate local information across all processors.

Note that we tend to use suffix `.spmd` to indicate a distributed local object which is a portion of big object. We also tend to use common variables without subfix `.spmd` since SPMD programming. In SPMD case, it may be a redundant idea to invent a `spmd` S4 class or methods for this purpose. The S3 class and methods are sufficient and fast in most SPMD case. In contrast, it is a better idea to invent a `ddmatrix` class as in **pbdBASE** (Schmidt *et al.* 2012a) and **pbdDMAT** (Schmidt *et al.* 2012c), then let efficient libraries handle computing and avoid tedious coding. A silly example can be found in the Section 3.4.

3.2. Binning

Binning is a classical statistics and can quickly summarize the data structure by setting some breaks between max and min of data. The demo command is

Shell Command

```
### At the shell prompt, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e "demo(binning,'pbdDEMO',ask=F,echo=F)"
```

The demo `binning` generates fake data on 2 processors, then utilize `mpi.bin` function as

R Code

```
mpi.bin <- function(x.spmd, breaks = pi / 3 * (-3:3)){
  bin.spmd <- table(cut(x.spmd, breaks = breaks))
  bin <- as.array(allreduce(bin.spmd, op = "sum"))
  dimnames(bin) <- dimnames(bin.spmd)
  class(bin) <- class(bin.spmd)
  bin
} # End of mpi.bin().
```

An easy implementation is to utilize `table` function to obtain local counts, then call `allreduce` to obtain global counts.

3.3. Quantile

Quantile is the other useful tool from fundamental statistics which provides data distribution for given quantile. The demo code is

Shell Command

```
### At the shell prompt, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e "demo(quantile,'pbdDEMO',ask=F,echo=F)"
```

This is only an implicit implementation to approximate a quantile and is not equivalent to the original `quantile` function in R. But in some sense, it should work well in large scale. The demo `quantile` generates fake data on 2 processors, then utilizes `mpi.quantile` function as

R Code

```
mpi.quantile <- function(x.spmd, prob = 0.5){
  if(sum(prob < 0 | prob > 1) > 0){
    stop("prob should be in (0, 1)")
  }

  N <- allreduce(length(x.spmd), op = "sum")
  x.max <- allreduce(max(x.spmd), op = "max")
  x.min <- allreduce(min(x.spmd), op = "min")

  f.quantile <- function(x, prob = 0.5){
    allreduce(sum(x.spmd <= x), op = "sum") / N - prob
  }

  uniroot(f.quantile, c(x.min, x.max), prob = prob[1])\$root
} # End of mpi.quantile().
```

where a numerical function is solved by `uniroot` to find out the appropriate value such that cumulated probability is less than or equal to the specified quantile.

This simple example shows that the SPMD is greatly applicable on large scale data analysis and likelihood computing. Note that the `uniroot` call is working in parallel and on distributed data, i.e. other optimization functions can work the same way, since SPMD simly assumes every processors do the same work simulatinuously.

3.4. Ordinary Least Square

This is a fundamental tool to find a solution for

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{y}$ is $N \times 1$ observed vector, $\boldsymbol{X}$ is $N \times p$ designed matrix which is full rank and $N >> p$, $\boldsymbol{\beta}$ is the insterested parameters and unknown to be estimated, and $\boldsymbol{\epsilon}$ is errors and to be minimized. A classical solution is to

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^t\boldsymbol{X})^{-1}\boldsymbol{X}^t\boldsymbol{y}$$

The demo code is

Shell Command

```
### At the shell prompt, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e "demo(ols,'pbdDEMO',ask=F,echo=F)"
```

The implementation is straight forward as

R Code

```
mpi.ols <- function(y.spmd, X.spmd){
  if(length(y.spmd) != nrow(X.spmd)){
    stop("length(y.spmd) != nrow(X.spmd)")
  }

  t.X.spmd <- t(X.spmd)
  A <- allreduce(t.X.spmd %*% X.spmd, op = "sum")
  B <- allreduce(t.X.spmd %*% y.spmd, op = "sum")

  solve(matrix(A, ncol = ncol(X.spmd))) %*% B
} # End of mpi.ols().
```

Note that this is a silly implementation for demonstrations and explain fundamental idea of OLS. This is only efficient for small N and small p. For larger scale, we suggest to simply convert `y.spmd` and `X.spmd` into block-cyclic format as in the Part II and to utilize **pbdBASE** and **pbdDMAT** for all matrix computation.

Part II

Distributed Matrix Methods

The **pbdBASE** and **pbdDMAT** packages offer a distributed matrix class, `ddmatrix`, as well as a collection of high-level methods for performing common matrix operations. For example, if you want to compute the mean of an R matrix `x`, you would call

```
mean(x)
```

That's exactly the same command you would issue if `x` is no longer an ordinary R matrix, but a distributed matrix. These methods range from

Unfortunately, using these higher methods comes with a different cost: getting the data into the distributed matrix class. This can be especially frustrating because we assume that the any object of class `ddmatrix` is *block cyclically distributed*. This concept is discussed at length in the **pbdBASE** vignette Schmidt *et al.* (2012d), and we do not intend to discuss the concept

of a block cyclic data distribution at length herein. However, we will demonstrate several examples of getting data into and out of the distributed block cyclic matrix format.

In short, once the hurdle of getting the data into the "right format" is out of the way, these methods offer very simple syntax (designed to mimic R as closely as possible) with the ability to scale computations on very large distributed machines.

4. Generating Random Data

4.1. SPMD to DMAT

pbdDEMO also provides reader examples

Shell Command

```
### At the shell prompt, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e "demo(spmd2dmat,'pbdDEMO',ask=F,echo=F)"
```

Shell Command

```
### At the shell prompt, run the demo with 4 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 4 Rscript -e "demo(dmat_random,'pbdDEMO',ask=F,echo=F)"
```

5. Reading Data

5.1. CSV Files

It is simple enough to read in a csv file serially and then distribute the data out to the other processors.

However, this is inefficient, especially if the user has access to a parallel file system. In this case, several processes should be used to read parts of the file, and then distribute that data out to the larger process grid.

5.2. SQL Databases

References

Chen WC, Ostrouchov G, Schmidt D, Patel P, Yu H (2012a). "pbdMPI: Programming with Big Data – Interface to MPI." R Package, URL http://cran.r-project.org/package=pbdMPI.

Chen WC, Ostrouchov G, Schmidt D, Patel P, Yu H (2012b). "A Quick Guide for the pbdMPI package." R Vignette, URL http://cran.r-project.org/package=pbdMPI.

Ostrouchov G, Chen WC, Schmidt D, Patel P (2012). "Programming with Big Data in R." URL http://r-pbd.org/.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.r-project.org/.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012a). "pbdBASE: Programming with Big Data – Core pbd Classes and Methods." R Package, URL http://cran.r-project.org/package=pbdBASE.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012b). "pbdDEMO: Programming with Big Data – Demonstrations of pbd Packages." R Package, URL http://cran.r-project.org/package=pbdDEMO.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012c). "pbdDMAT: Programming with Big Data – Distributed Matrix Algebra Computation." R Package, URL http://cran.r-project.org/package=pbdDMAT.

Schmidt D, Chen WC, Ostrouchov G, Patel P (2012d). "A Quick Guide for the pbdBASE package." R Vignette, URL http://cran.r-project.org/package=pbdBASE.