From 1 Core to Thousands: R to pbdR

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

1. University of Tennessee, USA
2. Oak Ridge National Laboratory, USA

August 8, 2013

Affiliations and Support

The pbdR Core Team
http://r-pbd.org

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

Ostrouchov, Patel, and Schmidt 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.

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

About This Presentation

Downloads

This presentation and supplemental materials are available at:

http://r-pbd.org/tutorial

Sample R scripts and pbs job scripts available on Nautilus from:

/lustre/medusa/mschmid3/tutorial/scripts.tar.gz

Contents

Contents

What is R?

- *lingua franca* for data analytics and statistical computing.
- Part programming language, part data analysis package.
- Dialect of S (Bell Labs).
- Syntax designed for data. scoping semantics, and 2 official OOP systems.

Who uses R?

Google, Pfizer, Merck, Bank of America, Shell[a],
Oracle[b], Facebook, bing, Mozilla, okcupid[c] ,
ebay[d],
kickstarter[e], the New York Times[f]

---

[a] https://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=0

[b] http://www.oracle.com/us/corporate/features/features-oracle-r-enterprise-498732.html

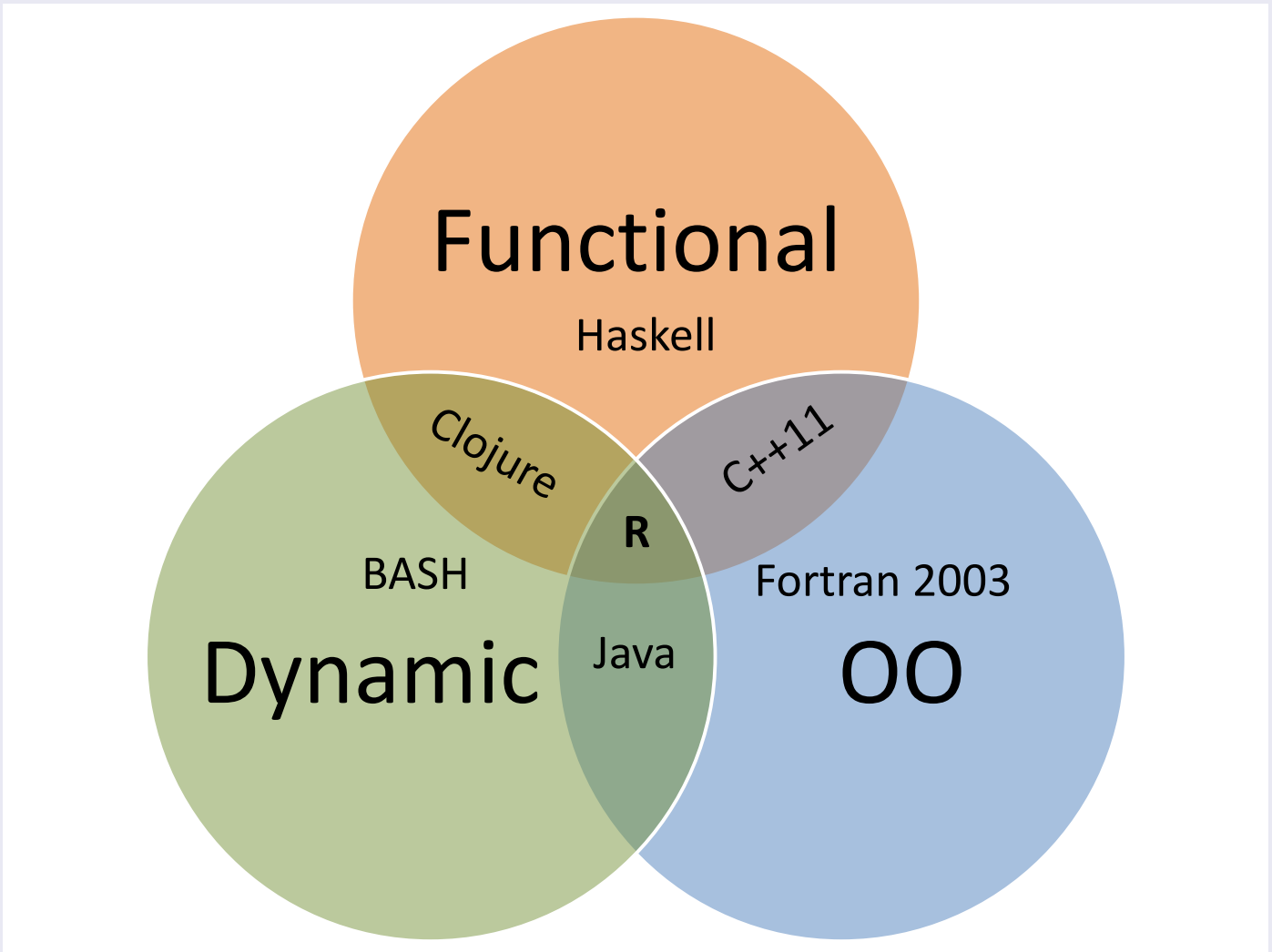
[c] http://www.revolutionanalytics.com/what-is-open-source-r/companies-using-r.php

[d] http://blog.revolutionanalytics.com/2012/09/using-r-in-production-industry-experts-share-their-experiences.html

[e] http://blog.revolutionanalytics.com/2012/09/kickstarter-facilitates-50m-in-indie-game-funding.html

[f] http://blog.revolutionanalytics.com/2012/05/nyt-charts-the-facebook-ipo-with-r.html

Language Paradigms



Data Types

- Storage: logical, int, double, double complex, character
- Structures: vector, matrix, array, list, dataframe
- Caveats: (Logical) `TRUE`, `FALSE`, `NA`

For the remainder of the tutorial, we will restrict ourselves to real number matrix computations.

High Level Syntax

```
1 x <- matrix(rnorm(30), nrow=10)
2 x <- x[-1, 2:5]
3 x <- log(abs(x) + 1)
4 xtx <- t(x) %*% x
5 ans <- svd(solve(xtx))
```

More than just a Matlab clone...

- Data science (machine learning, statistics, data mining, ...) is mostly matrix algebra.

  So what about Matlab/Python/Julia/...?
- Depends on your "religion"
- As a *data analysis* package, R is king.

High Level Syntax *for Data*

```
1 pca <- prcomp(x, retx=TRUE, scale=TRUE)
2 prop_var <- cumsum(pca$sdev)/sum(pca$sdev)
3 i <- min(which(prop_var > 0.9)) - 1
4
5 y <- pca$x[, 1:i]
```

Basics (1 of 2)

- The default method is to print:

```
R> sum
function (..., na.rm = FALSE)   .Primitive("sum")
```

- Use <- for assignment:

```
R> x <- 1
R> x+1
[1] 2
```

- Naming rules: mostly like C.
- R is case sensitive.
- We use . the way most languages use _, e.g., `La.svd()` instead of `La_svd()`.
- We use $ (sometimes @) the way most languages use .

Basics (2 of 2)

- Use ? or ?? to search help

```
1 R> ?set.seed
2 R> ?comm.set.seed
3 No documentation for 'comm.set.seed' in
    specified packages and libraries:
4 you could try '??comm.set.seed'
5 R> ??comm.set.seed
```

Addons and Extras

R has the Comprehensive R Archive Network (CRAN), which is a package repository like CTAN and CPAN.

From R

```
1 install.packages("pbdMPI") # install
2 library(pbdMPI)            # load
```

From Shell

```
1 R CMD INSTALL pbdMPI_0.1-6.tar.gz
```

Lists (1 of 1)

```
R> l <- list(a=1, b="a")
R> l
$a
[1] 1

$b
[1] "a"

R> l$a
[1] 1

R> list(x=list(a=1, b="a"), y=TRUE)
$x
$x$a
[1] 1

$x$b
[1] "a"


$y
[1] TRUE
```

Vectors and Matrices (1 of 2)

```
1  R> c(1, 2, 3, 4, 5, 6)
2  [1] 1 2 3 4 5 6
3
4  R> matrix(1:6, nrow=2, ncol=3)
5       [,1] [,2] [,3]
6  [1,]    1    3    5
7  [2,]    2    4    6
8
9  R> x <- matrix(1:6, nrow=2, ncol=3)
10
11 R> x[, -1]
12      [,1] [,2]
13 [1,]    3    5
14 [2,]    4    6
15
16 R> x[1, 1:2]
17 [1] 1 3
```

Vectors and Matrices (2 of 2)

```
R> dim(x)
[1] 2 3

R> dim(x) <- NULL
R> x
[1] 1 2 3 4 5 6

R> dim(x) <- c(3,2)
R> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Vector and Matrix Arithmetic (1 of 2)

```
R> 1:4 + 4:1
[1] 5 5 5 5

R> x <- matrix(0, nrow=2, ncol=3)

R> x + 1
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1

R> x + 1:3
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    1    3
```

Vector and Matrix Arithmetic (2 of 2)

```
R> x <- matrix(1:6, nrow=2)

R> x*x
     [,1] [,2] [,3]
[1,]    1    9   25
[2,]    4   16   36

R> x %*% x
Error in x %*% x : non-conformable arguments

R> t(x) %*% x
     [,1] [,2] [,3]
[1,]    5   11   17
[2,]   11   25   39
[3,]   17   39   61

R> crossprod(x)
     [,1] [,2] [,3]
[1,]    5   11   17
[2,]   11   25   39
[3,]   17   39   61
```

Linear Algebra (1 of 2): Matrix Inverse

$$x_{n\times n} \text{ invertible} \iff \exists y_{n\times n}\,(xy = yx = Id_{n\times n})$$

```
1  R> x <- matrix(rnorm(5*5), nrow=5)
2  R> y <- solve(x)
3
4  R> round(x %*% y)
5        [,1] [,2] [,3] [,4] [,5]
6  [1,]    1    0    0    0    0
7  [2,]    0    1    0    0    0
8  [3,]    0    0    1    0    0
9  [4,]    0    0    0    1    0
10 [5,]    0    0    0    0    1
```

Linear Algebra (2 of 2): Singular Value Decomposition

$$x = U\Sigma V^T$$

```
R> x <- matrix(rnorm(2*3), nrow=3)
R> svd(x)
$d
[1] 2.4050716 0.3105008

$u
           [,1]       [,2]
[1,] 0.8582569 -0.1701879
[2,] 0.2885390  0.9402076
[3,] 0.4244295 -0.2950353

$v
             [,1]        [,2]
[1,] -0.05024326 -0.99873701
[2,] -0.99873701  0.05024326
```

More than just a Matlab clone...

- Data science (machine learning, statistics, data mining, ...) is mostly matrix algebra.

  So what about Matlab/Python/Julia/...?
- The one you prefer depends more on your "religion" rather than differences in capabilities.
- As a *data analysis* package, R is king.

Simple Statistics (1 of 2): Summary Statistics

```
1  R> x <- matrix(rnorm(30, mean=10, sd=3), nrow=10)
2
3  R> mean(x)
4  [1] 9.825177
5
6  R> median(x)
7  [1] 9.919243
8
9  R> sd(as.vector(x))
10 [1] 3.239388
11
12 R> colMeans(x)
13 [1]  9.661822 10.654686  9.159025
14
15 R> apply(x, MARGIN=2, FUN=sd)
16 [1] 2.101059 3.377347 4.087131
```

Simple Statistics (2 of 2): Sample Covariance

$$cov(x_{n\times p}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)(x_i - \mu_x)^T$$

```
1  x <- matrix(rnorm(30), nrow=10)
2
3  # least recommended
4  cm <- colMeans(x)
5  crossprod(sweep(x, MARGIN=2, STATS=cm))
6
7  # less recommended
8  crossprod(scale(x, center=TRUE, scale=FALSE))
9
10 # recommended
11 cov(x)
```

Advanced Statistics (1 of 2): Principal Components

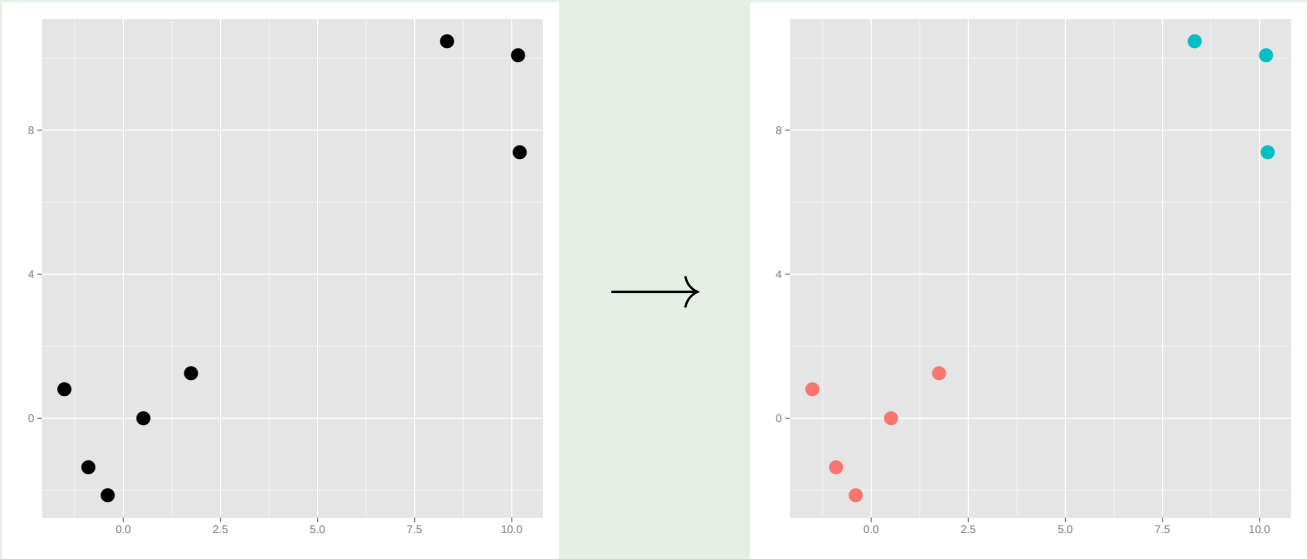
PCA = centering + scaling + rotation (via SVD)

```
R> x <- matrix(rnorm(30), nrow=10)

R> prcomp(x, retx=TRUE, scale=TRUE)
Standard deviations:
[1] 1.1203373 1.0617440 0.7858397

Rotation:
              PC1         PC2         PC3
[1,]  0.71697825 -0.3275365  0.6153552
[2,] -0.03382385  0.8653562  0.5000147
[3,]  0.69627447  0.3793133 -0.6093630
```

Advanced Statistics (2 of 2): k-Means Clustering




```
R> x <- rbind(matrix(rnorm(5*2, mean=0), ncol=2),
              matrix(rnorm(3*2, mean=10), ncol=2))
```

Advanced Statistics (2 of 2): k-Means Clustering

```
R> kmeans(x, centers=2)
K-means clustering with 2 clusters of sizes 5, 3

Cluster means:
        [,1]       [,2]
1 -0.1080612 -0.2827576
2  9.5695365  9.3191892

Clustering vector:
[1] 1 1 1 1 1 2 2 2

Within cluster sum of squares by cluster:
[1] 14.675072  7.912641
 (between_SS / total_SS =  93.9 %)

Available components:

[1] "cluster"      "centers"      "totss"
    "withinss"     "tot.withinss"
[6] "betweenss"    "size"
```

Contents

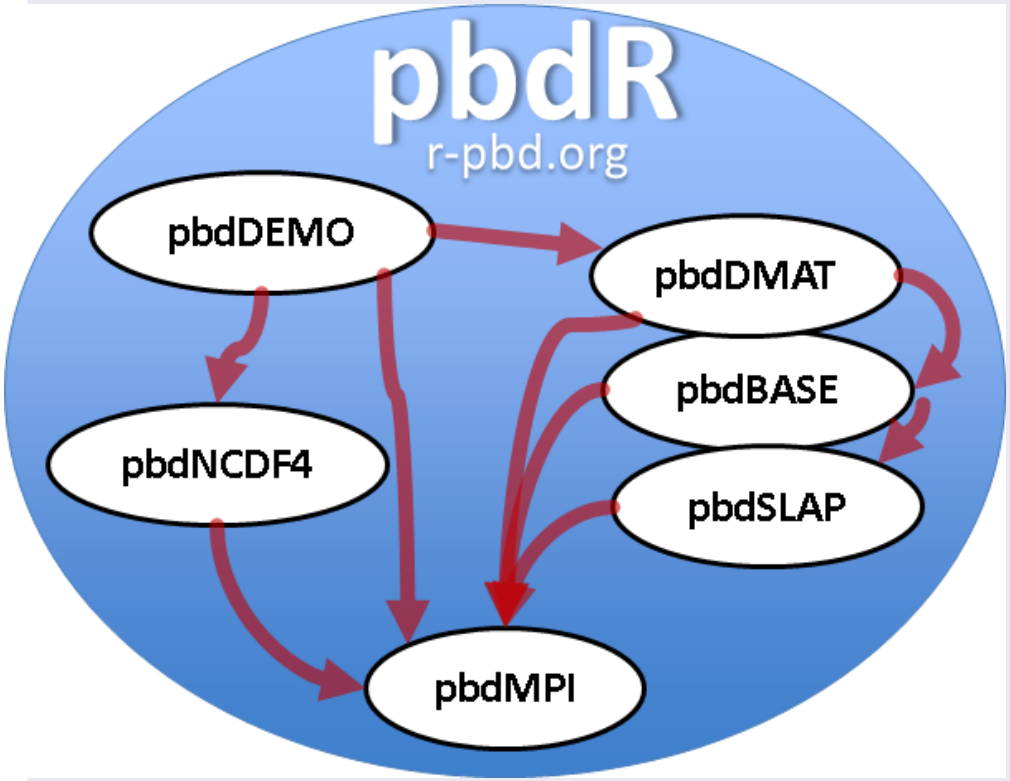
Programming with Big Data in R (pbdR)

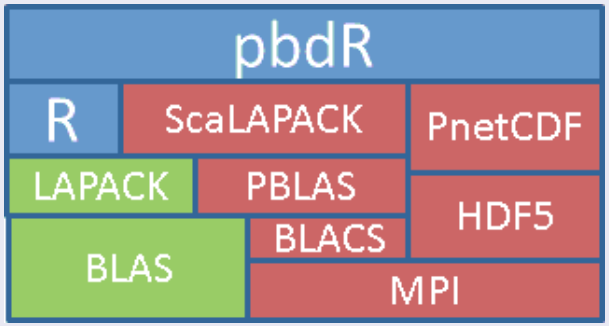
Striving for *Productivity, Portability, Performance*

- *Free*[a] R packages.
- Bridging high-performance C with high-productivity of R
- Scalable, big data analytics.
- Distributed data details implicitly managed.
- Methods have syntax *identical* to R.
- Powered by state of the art numerical libraries (MPI, ScaLAPACK, . . .)

[a] MPL, BSD, and GPL licensed

pbdR Packages





pbdR on HPC Resources

pbdR is currently installed and maintained on:

- Nautilus, UTK
- Kraken, UTK
- Newton, UTK
- Lens, ORNL
- Titan, ORNL
- tara, UMBC

If you are interested in maintaining pbdR, contact us at RBigData@gmail.com

Example Syntax

```
1 x <- x[-1, 2:5]
2 x <- log(abs(x) + 1)
3 xtx <- t(x) %*% x
4 ans <- svd(solve(xtx))
```

Look familiar?

The above runs on 1 core with R or 10,000 cores with pbdR

pbdR Paradigms

Programs that use pbdR utilize:

- Batch execution
- Single Program/Multiple Data (SPMD) style

And generally utilize:

- Data Parallelism

Batch Execution

- Non-interactive
- Use

```
Rscript my_script.r
```

or

```
R CMD BATCH my_script.r
```

- In parallel:

```
mpirun -np 2 Rscript my_par_script.r
```

Single Program/Multiple Data (SPMD)

- Difficult to describe, easy to do...
- Only one program is written, executed in batch on all processors.
- Different processors are autonomous; there is no manager.
- The dominant programming model for large machines.

Contents

Non-Optimal Choices Throughout

1. Only libre software used (no MKL, ACML, etc.).
2. 1 core = 1 MPI process.
3. No tuning for data distribution.

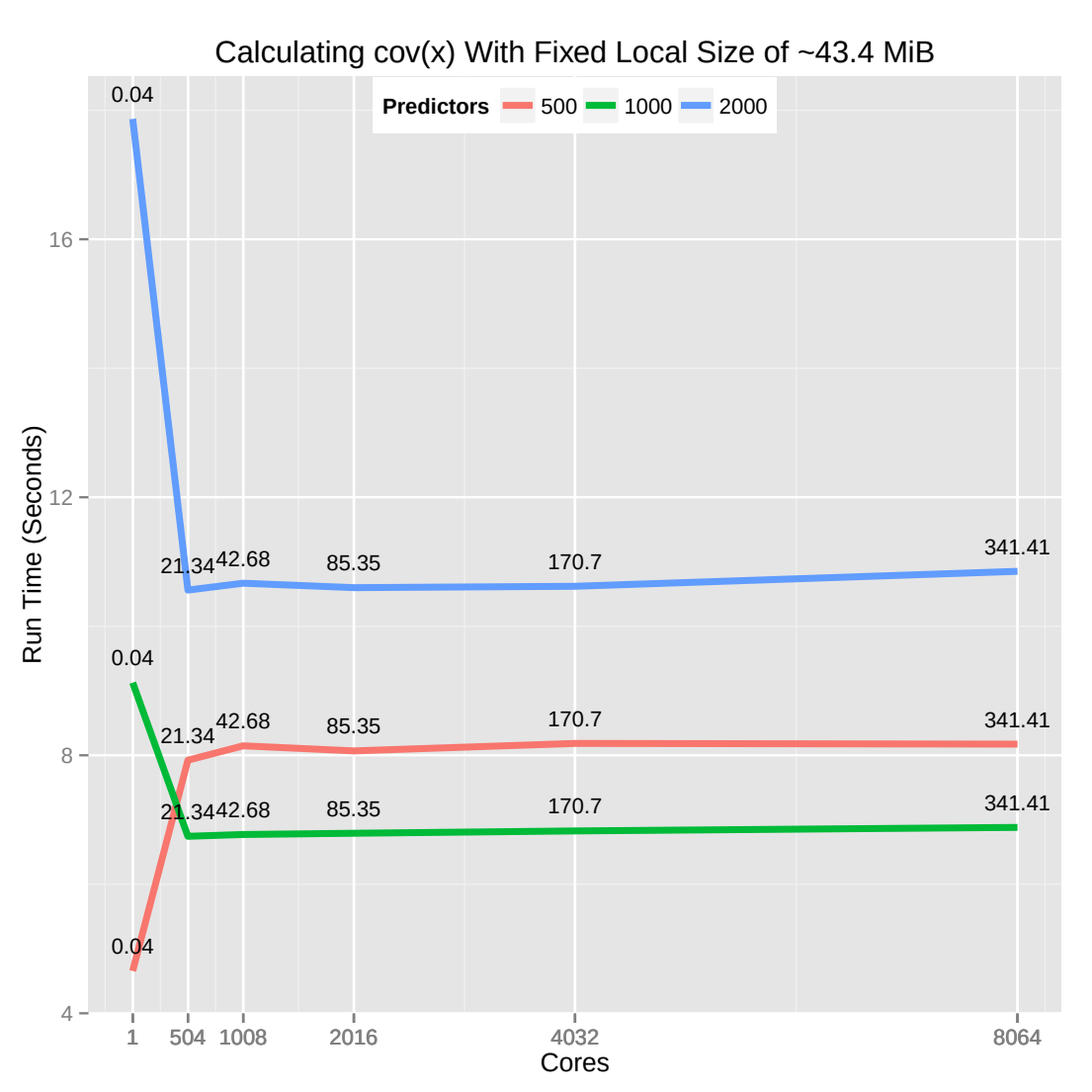
Benchmark Data

1. Random normal $N(100, 10000)$.
2. Local problem size of $\approx 43.4 MiB$.
3. Three sets: 500, 1000, and 2000 columns.
4. Several runs at different core sizes within each set.

Covariance Code

```
1 x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=mean, sd=sd)
2
3 cov.x <- cov(x)
```

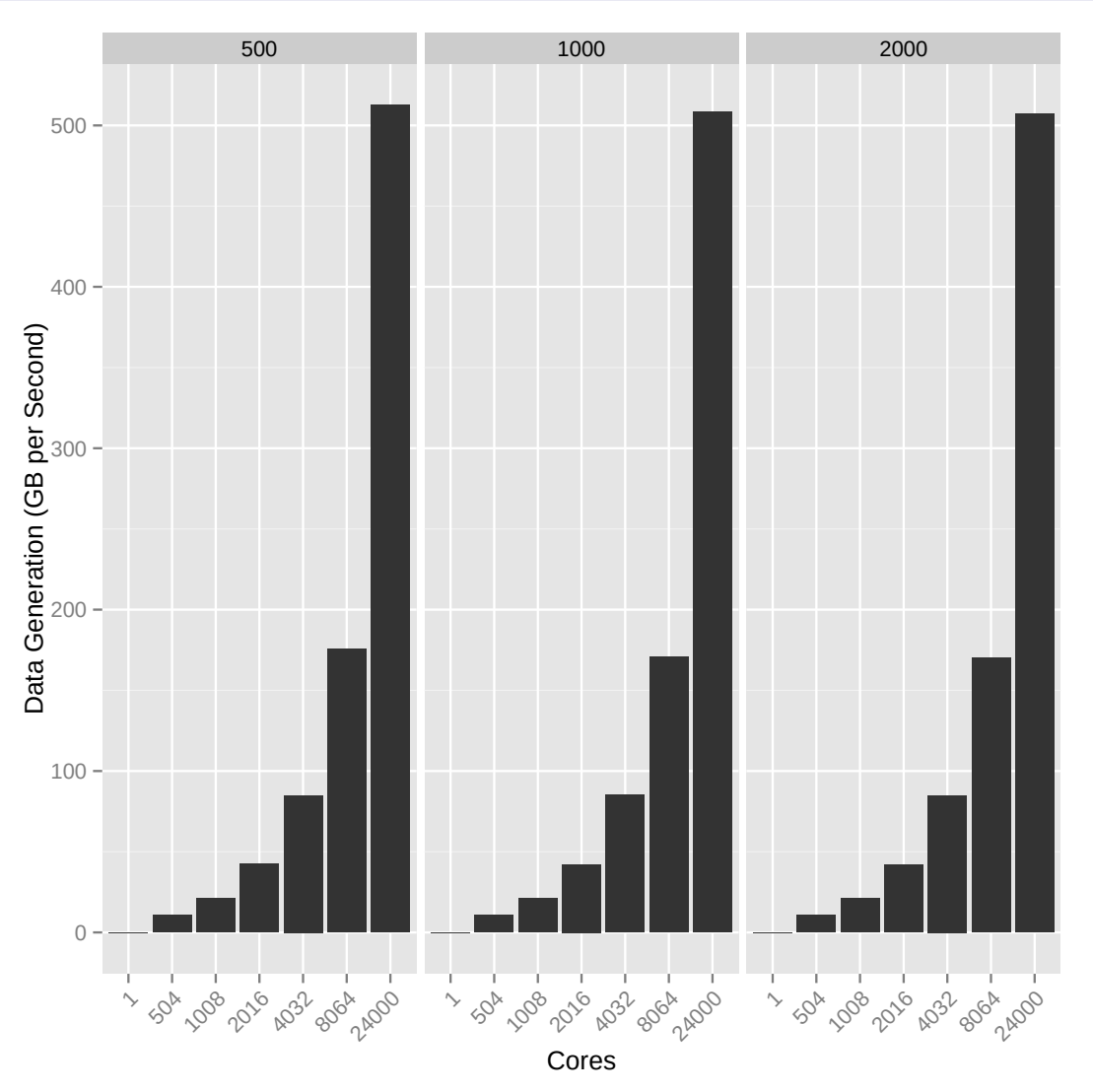
cov()


Calculating cov(x) With Fixed Local Size of ~43.4 MiB
Predictors 500 1000 2000
Run Time (Seconds)
16
12
8
4
0.04
21.34
42.68
85.35
170.7
341.41
Cores
1
504
1008
2016
4032
8064

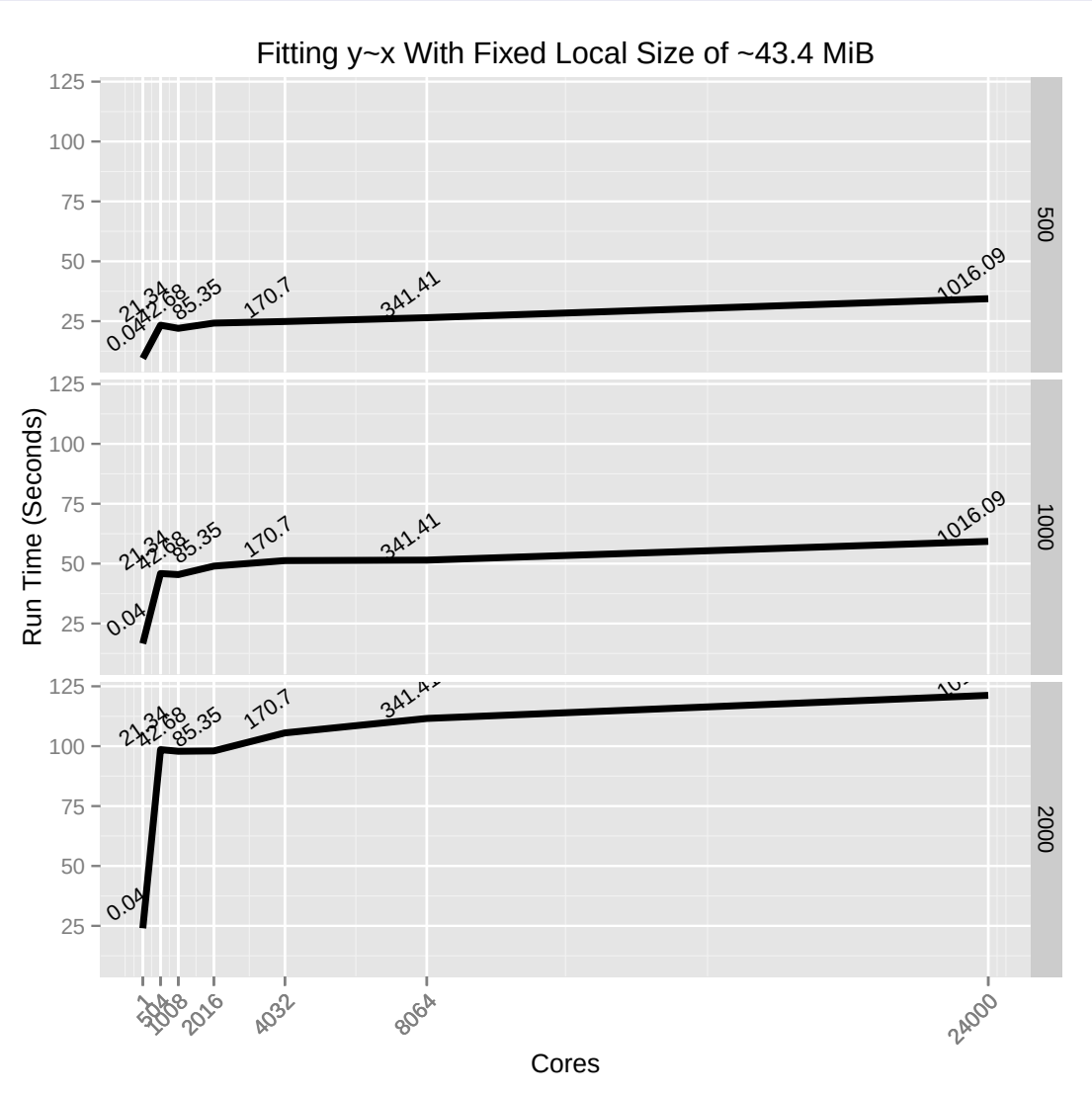
Linear Model Code

```
1 x <- ddmatrix("rnorm", nrow=n, ncol=p, mean=mean, sd=sd)
2 beta_true <- ddmatrix("runif", nrow=p, ncol=1)
3
4 y <- x %*% beta_true
5
6 beta_est <- lm.fit(x=x, y=y)$coefficients
```

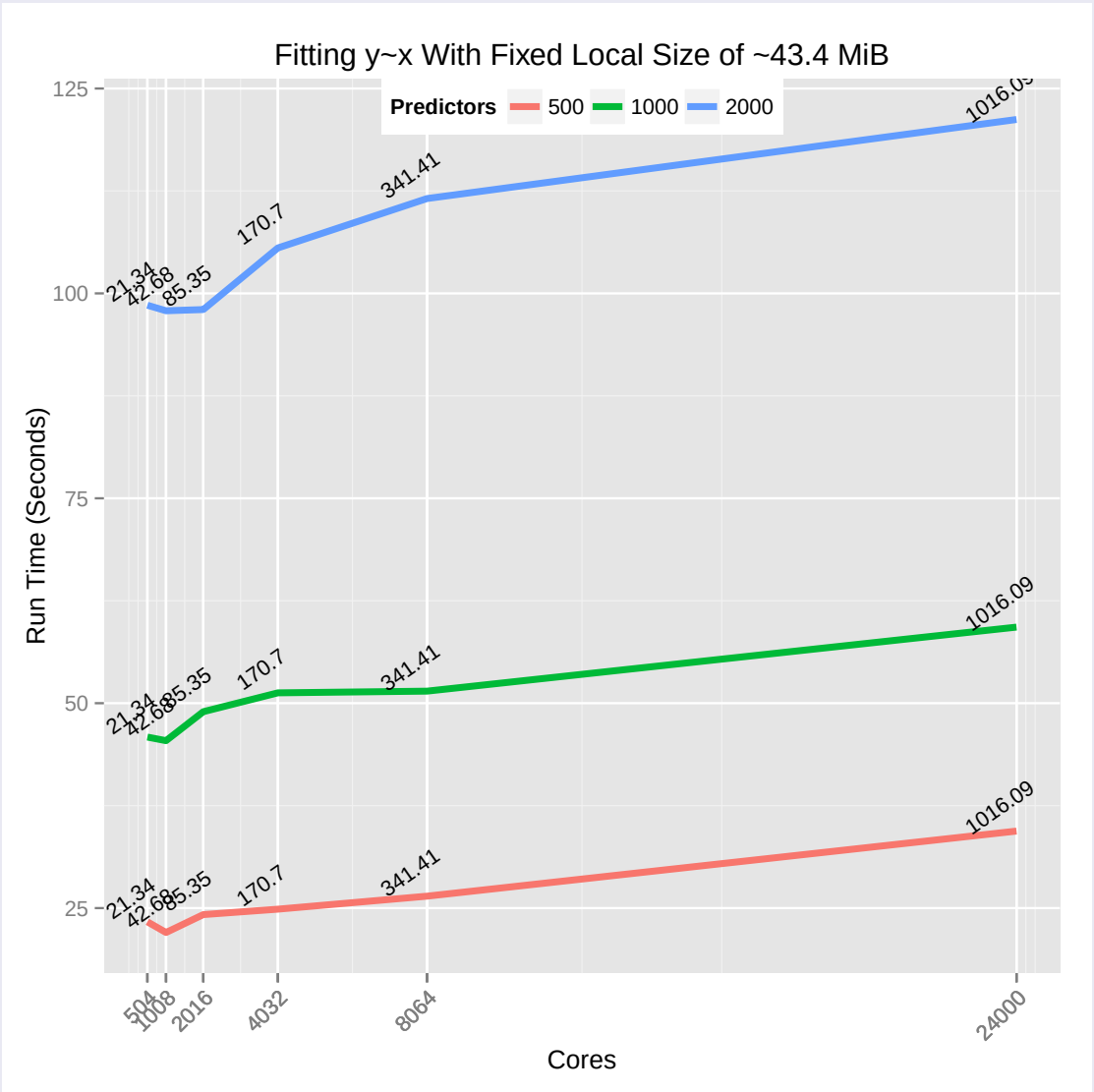
Data Generation



lm.fit()


Fitting y~x With Fixed Local Size of ~43.4 MiB
Run Time (Seconds)
Cores
500
1000
2000
0.04
21.34
42.68
85.35
170.7
341.41
1016.09
1
504
1008
2016
4032
8064
24000

lm.fit()

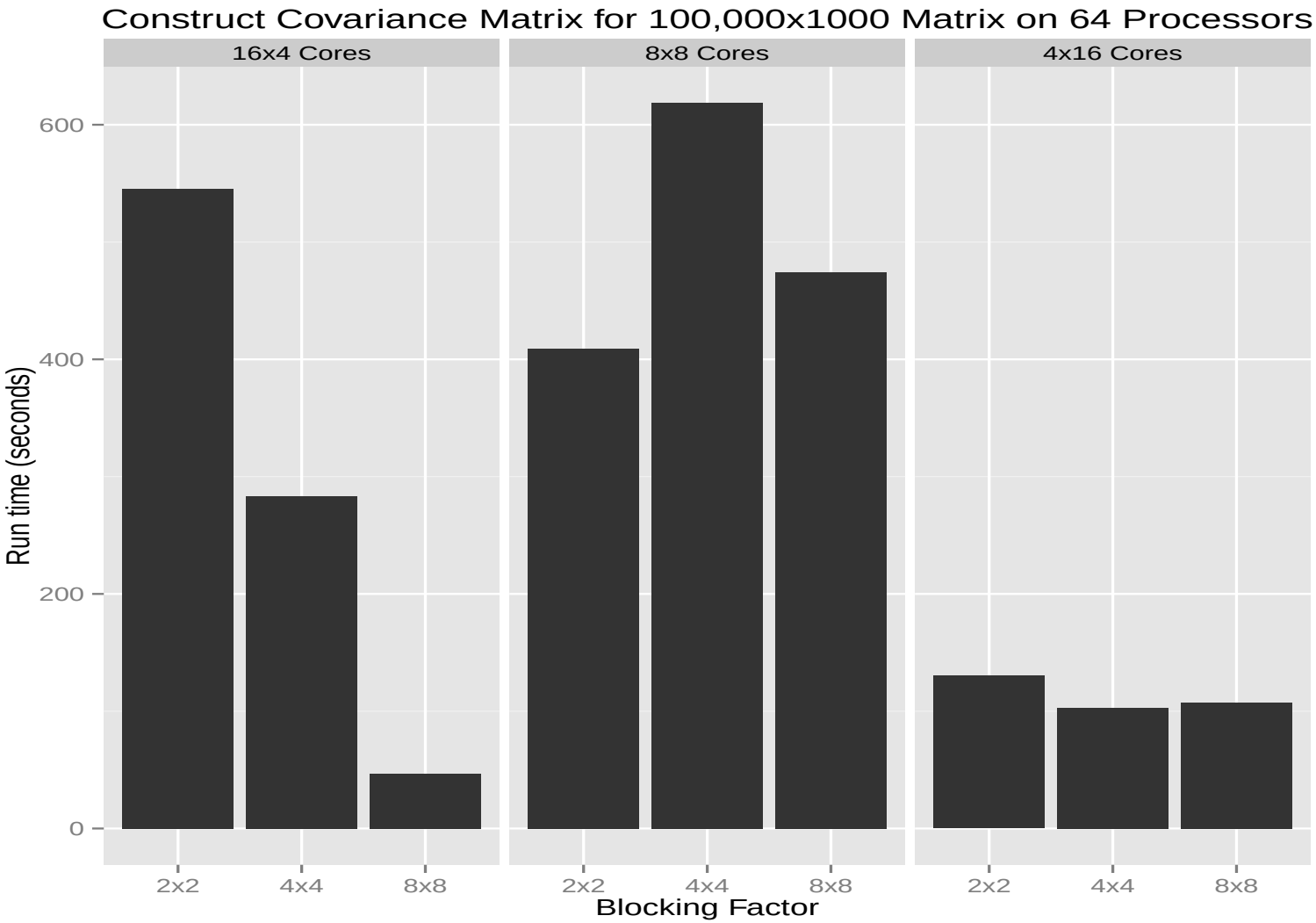

Fitting y~x With Fixed Local Size of ~43.4 MiB
Predictors 500 1000 2000
Run Time (Seconds)
125
100
75
50
25
21.34
42.68
85.35
170.7
341.41
1016.09
504
1008
2016
4032
8064
24000
Cores

Contents

Challenges

- Perceptions.
- Library loading.
- Profiling.

Covariance Revisited: Distributed Data Parameter Calibration



Tutorials

- SC13, November 17-22, Denver, Colorado, USA

Invited Talks

- IASC, Aug 22-23, Seoul
- World Statistics Congress, August 25-30, Hong Kong

Thanks for coming!

Questions?

Be sure to stick around for the tutorial