Case Study
Bayes and Big Data:
The Consensus Monte Carlo Algorithm (Scott et al. 2013)

Artur Begyan, Hans Hanley, Parley R Yang

UNIVERSITY OF OXFORD
DEPARTMENT OF STATISTICS

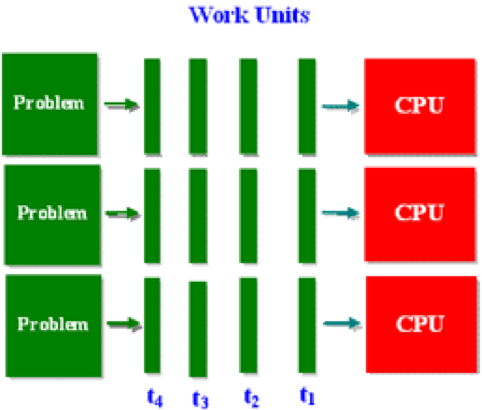
3 Dec 2019

Oxford, UK

Overview

- Background
  - Parallel Computing
  - Monte Carlo Sampling
  - Monte-Carlo Algorithm with Multiple Machines
- Consensus Monte-Carlo Methods
- Consensus Monte-Carlo Examples
  - Bernoulli-Beta Experiments
  - Application to Data with Hierarchical Models
- Conclusion

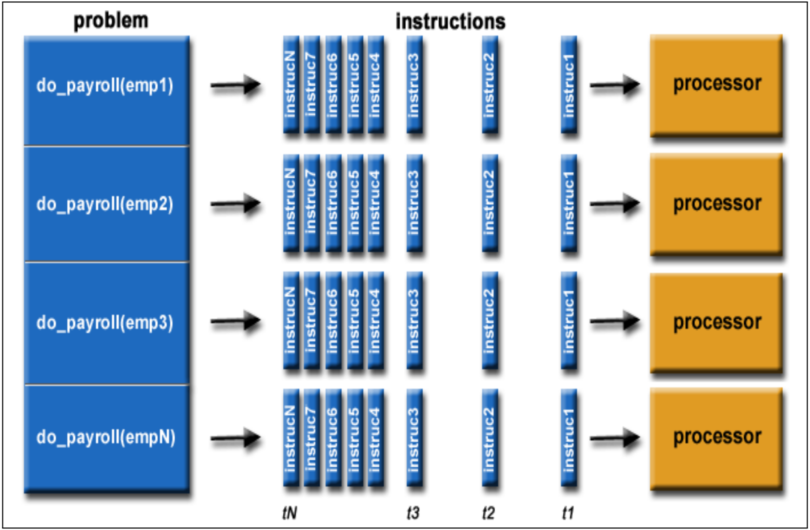
What is Parallel Computing?



What is Parallel Computing?

- Parallel computing is a form of computation in which many calculations are carried simultaneously.
- How is Parallel Computing Accomplished
  - A problem is broken into discrete parts
  - The discrete parts are handled by:
    - Multiple CPU Cores
    - Multiple CPUs/GPUs
    - Multiple Machines

Parallel Computing



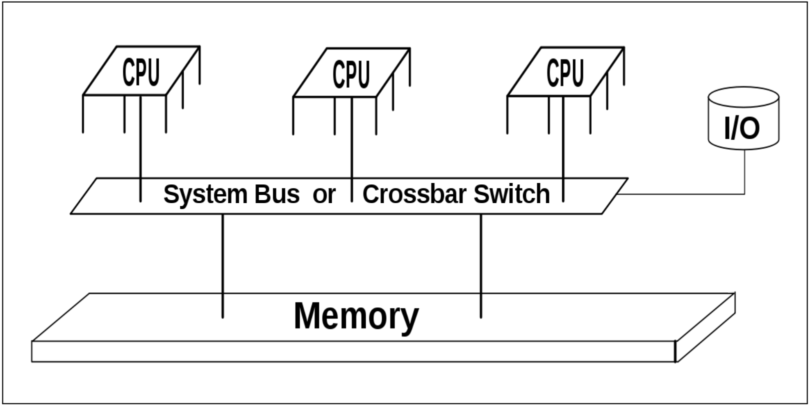
Benefits of Parallel Computing

- Benefits in all types of parallel computing
  - Save time/memory
  - Provide concurrency (use of unused resources)
- Benefits in multi-machine computing
  - Solve larger problems (can handle 'Big' data; for large data then, multi-machine parallel computing is a must!)
  - Use of non-local resources, (i.e. machines in cloud or rented servers)

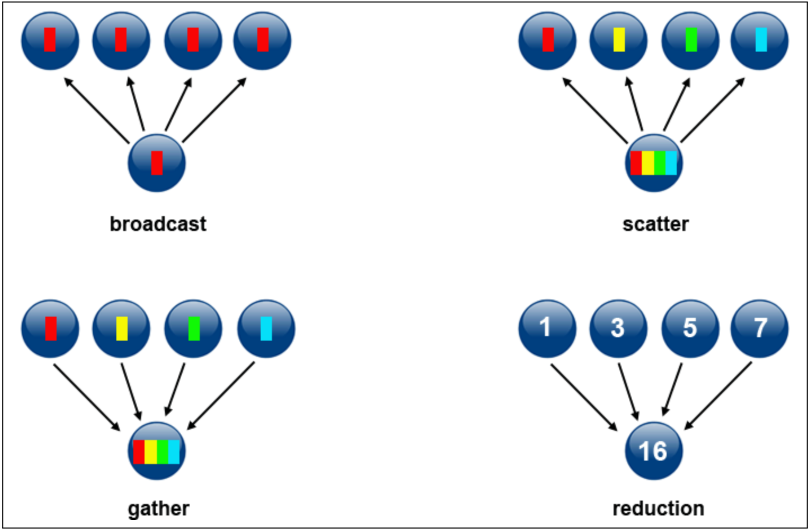
Issues with Parallel Computing

- Issues with multi-core or multi CPU/GPU parallel computing
  - Cannot alleviate bottlenecks of memory and disk access
  - Difficult programming- (GPU computing is notoriously difficult to debug; race conditions)
- Issues with multi-machine parallel computing
  - Efficiency is lower significantly
  - Communication is inherently slower. Passing messages among machines is expensive.

Communication in Parallel Computing



Communication in Parallel Computing



Monte-Carlo Sampling

- Often we want to compute expectation of a posterior distribution.
- For expectations $\Phi = g(\theta)$, we use the following:

$$\int_{g(\Theta)} \phi p(\Phi|Y)d\Phi = \int_{\Theta} g(\phi)p(\Phi|Y)d\Theta$$

- Sometimes we do not know how to compute the integral!
- We use Markov Chain Monte Carlo sampling!

Simulations from Distributions

- We take a sample of S values from the posterior distribution of θ for large S:

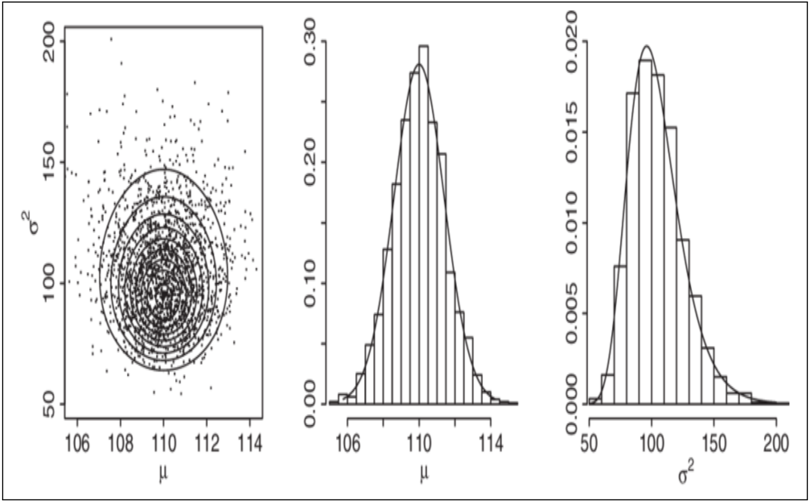
$$\theta^{(1)}, ..., \theta^{(S)} \quad \sim \quad p(\theta|Y)$$

- By the Law of Large Numbers

$$\frac{1}{S}\Sigma\theta^{(i)} \rightarrow E[\theta|Y]$$

$$\frac{1}{S}\Sigma g(\theta^{(i)}) \rightarrow E[g(\theta)|Y]$$

Monte-Carlo


200
150
100
50
σ^2
106 108 110 112 114
μ
0.30
0.20
0.10
0.00
106 110 114
μ
0.020
0.015
0.010
0.005
0.000
50 100 150 200
σ^2

Monte-Carlo Full Algorithm

Data: Y, p$(\theta), S$
Use Baye's Theorem for the Posterior p$(\theta|Y)$
for $i < S$ **do**
  Take Sample(p$(\theta|$Y))
end

Result: Estimated expectation of posterior E$[\theta|Y]$
Algorithm 1: How to estimate Θ from posterior distribution

Parallel Computing In Practice!

What if we want to use Parallel Computing to compute a posterior distribution?

Multiple Machine Monte-Carlo

- Attacks 'Big Data' problems by dividing the data across multiple machines.
- The Monte-Carlo algorithm is then performed on each machine
- Posterior draws from each machine are then combined to beliefs about the model.

Example of MCMC on multi-machines

Example on a single layer hierarchical logistic regression model.

$$
\begin{aligned}
y_{ij} &\sim Binomial(n_{ij}, p_{ij}) \\
\text{logit}(\text{p}_{ij}) &= (x)_{ij}^T \beta_i \\
\beta &\sim \mathbb{N}(\mu, \Sigma) \\
\mu|\Sigma &\sim \mathbb{N}(0, \Sigma/\kappa) \\
\Sigma^{-1} &\sim W(I, \nu)
\end{aligned}
$$

where $W(I, \nu)$ is the Wishart distribution with sum of squares I and scale parameter ν.

Example of MCMC on multi-machines

- Partition the data by domain
- Assign one worker to be the master node responsible for the full prior
- Draw each β_i given the current values of μ and Σ on each machine
- Draw μ and Σ based on the current β_i on the master machine.
- Repeat until convergence

Example of MCMC on multi-machines

- Partition the data by domain
- Assign one worker to be the master node responsible for the full prior
- Draw each β_i given the current values of μ and Σ on each machine
- Draw μ and Σ based on the current β_i on the master machine.
- Repeat until convergence

Conclusions of Example

- The 50 machine run completed in 5 hours
- The 500 machine run completed in 2.75 hours (1 iteration per second)
- There is a 5x inefficiency at play
- From the graph, we see that job failures were not enough to cause the discrepancy
- Communication was the problem!
  - For the 50 machine run: (μ, Σ) step takes 100ms
  - For the 500 machine run: (μ, Σ) step takes 250ms
  - Logging in the code shows that the inefficiency comes form communication.
- It takes time for the machines to communicate!

Step Times for the MCMC algorithm

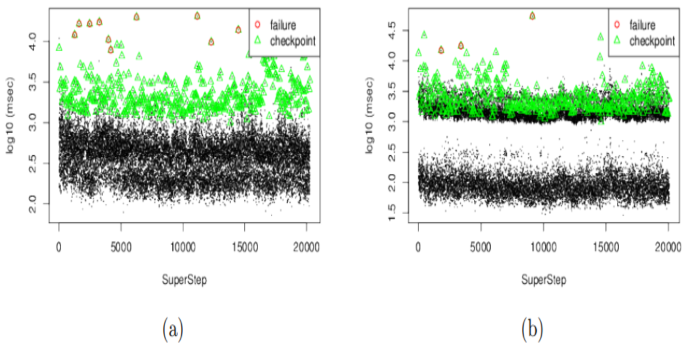


Figure 1: *Step times for the naive MCMC algorithm in Section 2 with (a) 500 and (b) 50 machines.*

Consensus Monte Carlo

- Break the data into groups (called "shards")
- Give each shard to a worker machine which does a full Monte Carlo simulation from a posterior distribution given its own data
- Combine simulations from each worker to produce a set of global draws representing the consensus belief among the workers

1. Divide $\mathbf{y}$ into shards $\mathbf{y}_1, \ldots, \mathbf{y}_S$.
2. Run S separate Monte Carlo algorithms to sample $\theta_{sg} \sim p(\theta|\mathbf{y}_s)$ for $g = 1, \ldots, G$, with each shard using the fractionated prior $p(\theta)^{1/S}$.
3. Combine the draws across shards using weighted averages: $\theta_g = \left(\sum_s W_s\right)^{-1} \left(\sum_s W_s \theta_{sg}\right)$.

Consensus Monte Carlo

- Let $\mathbf{y}$ represent the full data, let $\mathbf{y}_s$ denote shard s, and let θ denote the model parameters
- For models with the appropriate independence structure, the system can be written
$$p(\theta|\mathbf{y}) \propto \prod_{s=1}^{S} p(\mathbf{y}_s|\theta)p(\theta)^{\frac{1}{S}}$$
- The prior distribution $p(\theta) = \prod_{s=1}^{S} p(\theta)^{\frac{1}{S}}$ is split into S components to preserve the total amount of prior information in the system

Combining draws by weighted averages

- Suppose worker s generates draws $\theta_{s1}, \ldots, \theta_{sG}$ from $p(\theta|\mathbf{y}_s) \propto p(\mathbf{y}_s|\theta)p(\theta)^{\frac{1}{S}}$.
- Suppose each worker is assigned a weight represented by a matrix W_s. The consensus posterior for draw g is

$$\theta_g = (\sum_s W_s)^{-1} \sum_s W_s \theta_{sg}$$

- When each $p(\theta|\mathbf{y}_s)$ is Gaussian, the joint posterior $p(\theta|\mathbf{y})$ is also Gaussian, hence the above equation can be made to yield exact draws from $p(\theta|\mathbf{y})$

Choosing Weights

- The weight $W_s = \Sigma_s^{-1}$ is optimal (for Gaussian models), where $\Sigma_s = Var(\theta|\mathbf{y}_s)$
- An obvious Monte Carlo estimate of Σ_s is sample variance of $\theta_{s1}, ..., \theta_{sG}$
- Sub-optimal but computationally efficient weighting
    - Ignore the covariances in Σ_s
    - Apply equal weighs
- In practice, information-based weighting will usually be necessary

Nested Hierarchical models

- If the data has nested structure, where $y_{\text{ij}} \sim f(y|\phi_{\text{j}})$ and $\phi_{\text{j}} \sim p(\phi|\theta)$ then Consensus Monte Carlo (CMC) can be applied in a straightforward way
- For CMC to work, data needs to be partitioned so that no group is split across multiple shards
- Run CMS, store the θ draws and discard ϕ_{j} draws. Combining the draws of θ_{sg} produces a set of draws $\theta_1, ..., \theta_{\text{G}}$ approximating $p(\theta|\mathbf{y})$.
- Conditional on the simulated draws of θ, sampling $\phi_{\text{j}} \sim p(\phi_{\text{j}}|\mathbf{y}) = p(\phi_{\text{j}}|\mathbf{y}_{\text{j}}, \theta)$ is an embarrassingly parallel problem.

Binomial data with a beta prior: basic case

- Data from 1000 Bernoulli with one head.
- 100 machines with equal weights each with prior $\theta \sim Beta(0.01, 0.01)$
- In 1 machine case, we start with $\theta \sim Beta(1, 1)$ thus expect to see posterior $Beta(2, 1000)$

Binomial data with a beta prior: basic case

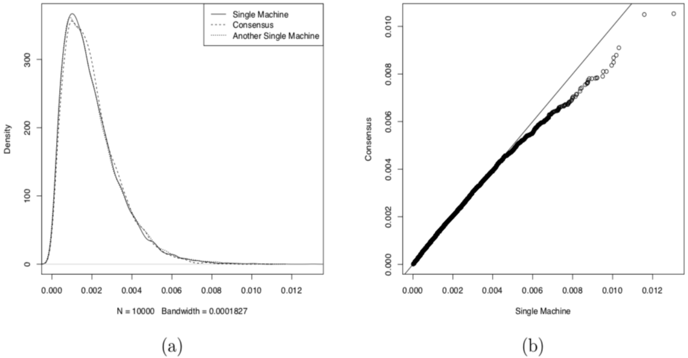
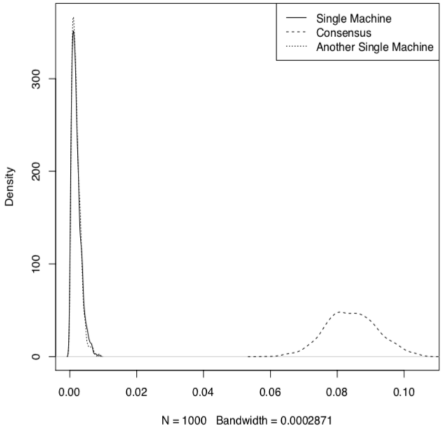
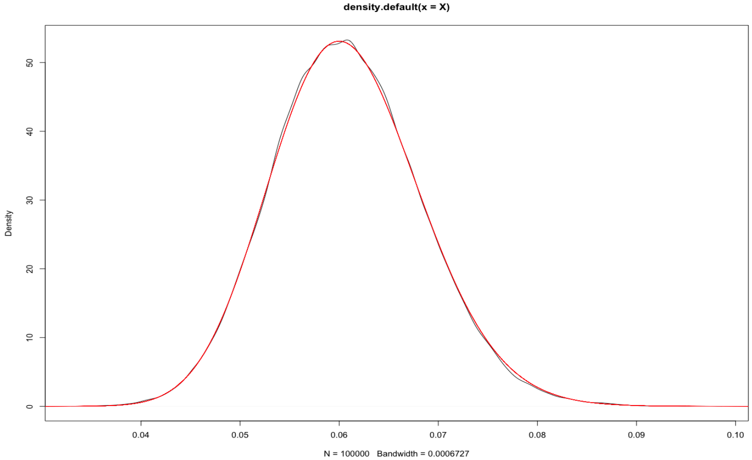


Figure 4: *(a) Posterior draws from binomial data. (b) A qq plot showing that the tails of the consensus Monte Carlo distribution in panel (a) are slightly too light.*

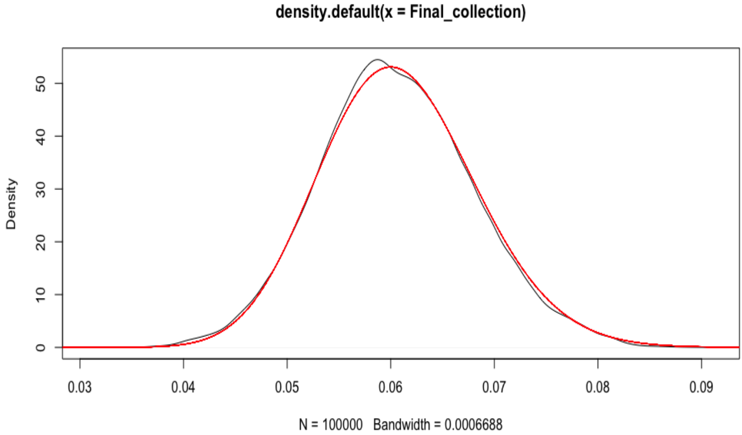
Binomial data with a uniform prior



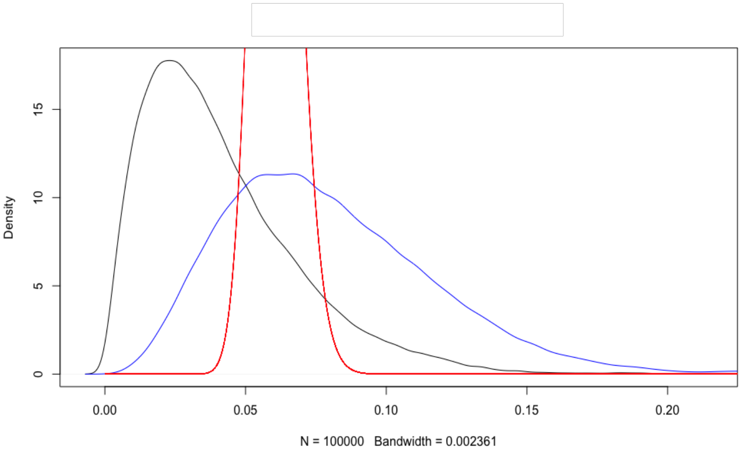
Binomial data with a beta prior: deeper insight — 1 Machine (standard MCMC result)



Binomial data with a beta prior: deeper insight — 20 Machines summary



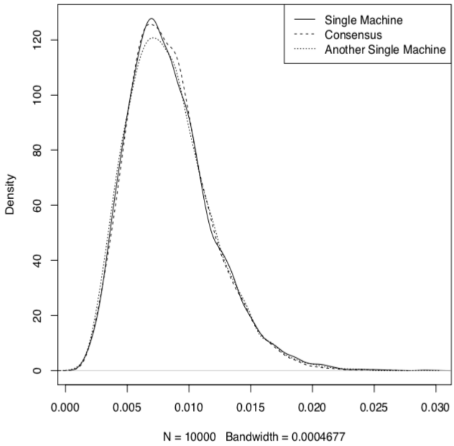
Binomial data with a beta prior: deeper insight — Some individual machines



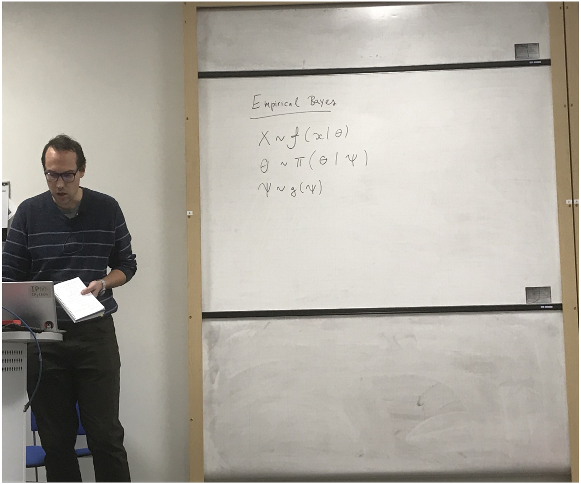
Binomial data with a beta prior: different weights

- Data from 1000 Bernoulli with $p = 0.01$
- Weights assigned differently: 20,20,70,100,500; each of the 5 machines start with prior Beta(0.2,0.2)
- The 1 machine case starts with $Beta(1, 1)$

Binomial data with a beta prior: different weights



Hierarchical models: preliminaries



Hierarchical models: setting

- Main job: fitting a hierarchical model.
- $$y_{ij} \sim Poisson(N_{ij}\lambda_{ij})$$
  where y_{ij} is the number of times advertisement i from advertiser j was clicked, N_{ij} is the number of times it was shwon.
- $$\log(\lambda_{ij}) = \beta_j^T \mathbf{x}_{ij}$$
  where $\mathbf{x}_{ij}$ is a small set of explanatory variables,β follows the distribution
  $$\beta_j \sim N(\mu, \Sigma)$$
  while μ follows normal distribution and Σ^{-1} follows Wishart distribution.

Hierarchical models: setting

- Data has 24 million observations, split into 867 shards.
- 10k MCMC interations
- One machine with 5+ cores work on 5 shards of data in parallel (CMC) vs. one machine with 1 core work on the same amount of data in sequential order (MC).

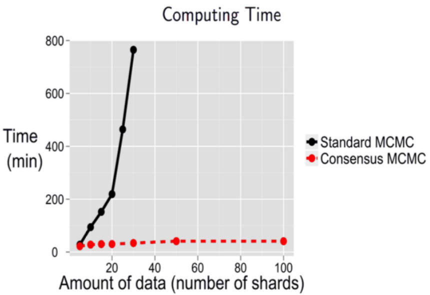
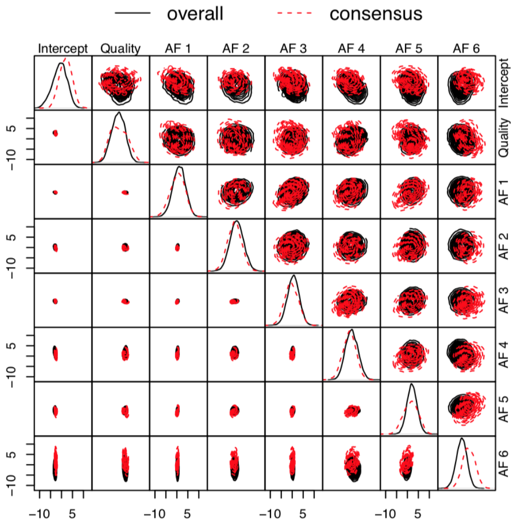
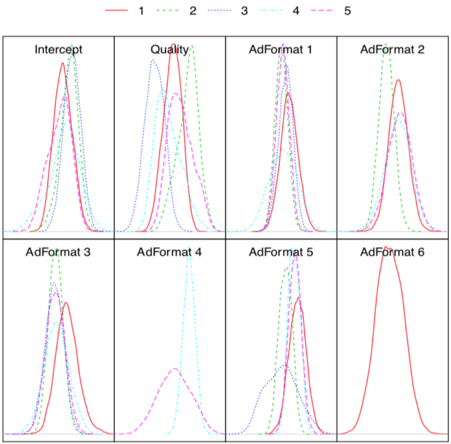


Figure 10: *Time required to complete 10,000 MCMC draws with different numbers of shards under the single machine and consensus Monte Carlo algorithm.*

Hierarchical models: overall results



Hierarchical models: individual results



Conclusion

- For 'Big Data' due the size often multiple machines are required in order to preform computations
- Communication issues can often inhibit the efficiency of 'Big Data' algorithms
- CMC is a way of minimising communication costs in parallel computing while calculating statistics on a posterior
- We have shown here, two examples (one binomial, the other hierarchical) that illustrate the benefits of this approach