Fast Moment-Based Estimation for Hierarchical Models

Patrick O. Perry
New York University

Motivating Application: Recommender Systems

- Large population of Users and Items
- Goal: Recommend items to users
- Examples: online shopping (“you might also like…”), targeted advertising

Items

movie	title	genre
1	Toy Story (1995)	Comedy
2	Jumanji (1995)	Children
3	Grumpier Old Men (1995)	Comedy
4	Waiting to Exhale (1995)	Drama
5	Father of the Bride Part II (1995)	Comedy
.	.	.
.	.	.
.	.	.
10677	Bedtime Stories (2008)	Children
10678	Manhattan Melodrama (1934)	Drama
10679	Choke (2008)	Comedy
10680	Revolutionary Road (2008)	Drama
10681	Blackadder Back & Forth (1999)	Comedy

Users

user
1
2
3
4
5
.
.
.
.
.
69873
69874
69875
69876
69877
69878

Ratings

	user	movie	score	time
1	36072	21	3	1995-01-09 11:46:49
2	36072	47	5	1995-01-09 11:46:49
3	36072	1058	3	1995-01-09 11:46:49
4	34294	1	4	1996-01-29 00:00:00
5	34294	10	4	1996-01-29 00:00:00
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
10000050	61718	2395	3.5	2009-01-05 04:52:12
10000051	61718	6887	3.5	2009-01-05 04:52:17
10000052	61718	2869	2	2009-01-05 04:52:22
10000053	61141	4691	2.5	2009-01-05 04:55:03
10000054	61141	9153	3	2009-01-05 05:02:16

Two Main Approaches

Content-based: recommend items similar to those the user liked in the past

Collaborative: recommend items that similar users liked

A Model That Does Both

Group (User) i = 1,...,M:

- predictors X_i ($n_i \times p$)
- response y_i (n_i)

Model:

$$E(y_{ij} \mid \beta_i) = x_{ij}^T \beta_i$$

$$\beta_i \sim N(\mu, \Sigma)$$

(Condliff et al. 1999; Ansari et al. 2000)

Response

$$y_i = \begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{in_i} \end{bmatrix}$$

rating
5
3
5
3
3
5
.
.
.

Features

$$X_i = \begin{bmatrix} x_{i11} & x_{i12} & \cdots & x_{i1p} \\ x_{i21} & x_{i22} & \cdots & x_{i2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{in_i1} & x_{in_i2} & \cdots & x_{in_ip} \end{bmatrix}$$

action	children	comedy	drama	movie.popularity	user.liked.last
0	0	0	1	1.1	0
0	0	0	1	0.1	0
1	0	0	0	1.8	0
1	0	0	0	-0.8	1
0	0	0	1	0.3	0
0	0	1	0	0.4	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

User i's expected rating for item j:

$$E(y_{ij} \mid \beta_i) = x_{ij}^{\top}\beta_i$$

Item attributes

User tastes

Example:

"User i likes action movies,
movie j is an action movie;
we should recommend movie j to user i."

User i's tastes:

$$\beta_i \sim N(\mu, \Sigma)$$

Population average   Covariance structure

Example:

"User i likes action movies,
users who like action tend to dislike drama;
we should assume that user i dislikes drama."

Hierarchical Model

Content-based: $E(y_{ij} \mid \beta_i) = x_{ij}^T \beta_i$

Collaborative: $\beta_i \sim N(\mu, \Sigma)$

Problem: Fitting at Commercial Scale

(Zhang and Koren, 2007; Agarwal, 2008; Naik et al., 2008; Agarwal and Chen, 2009)

Likelihood-Based Fitting is Slow

Method	Initial Cost	Cost per Iteration	Iterations
Expectation-Maximization	Np^2	Mp^3	Hundreds
Newton-Raphson	Np^2	Mp^4	Tens
Profile Likelihood (lme4)	Np^2	Mp^3	Tens to Hundreds

(Movielens: $M \approx 10^5$, $N \approx 10^7$, $p \approx 10$)

Popular Approach #1: Split/Combine

Idea: divide data between K processors, compute separate estimates, then combine

Pro: cuts wall clock time by a factor of K (but does not reduce total amount of computation)

(Huang and Gelman, 2005; Gebregziabher et al. 2012; Scott et al. 2013, ...)

Popular Approach #2:
Stochastic Gradient Descent (SGD)

Idea: maximize h-likelihood (treating random effects like parameters), use gradient-based optimization

Pro: often faster than maximum likelihood

Con: requires tuning parameters, can sometimes be inconsistent

(Lee and Nelder 2006; Dror et al. 2011, ...)

Today's Talk: Moment-Based Estimation

Idea: split data into M chunks, compute group-specific effect estimates, then use moment matching for population parameters

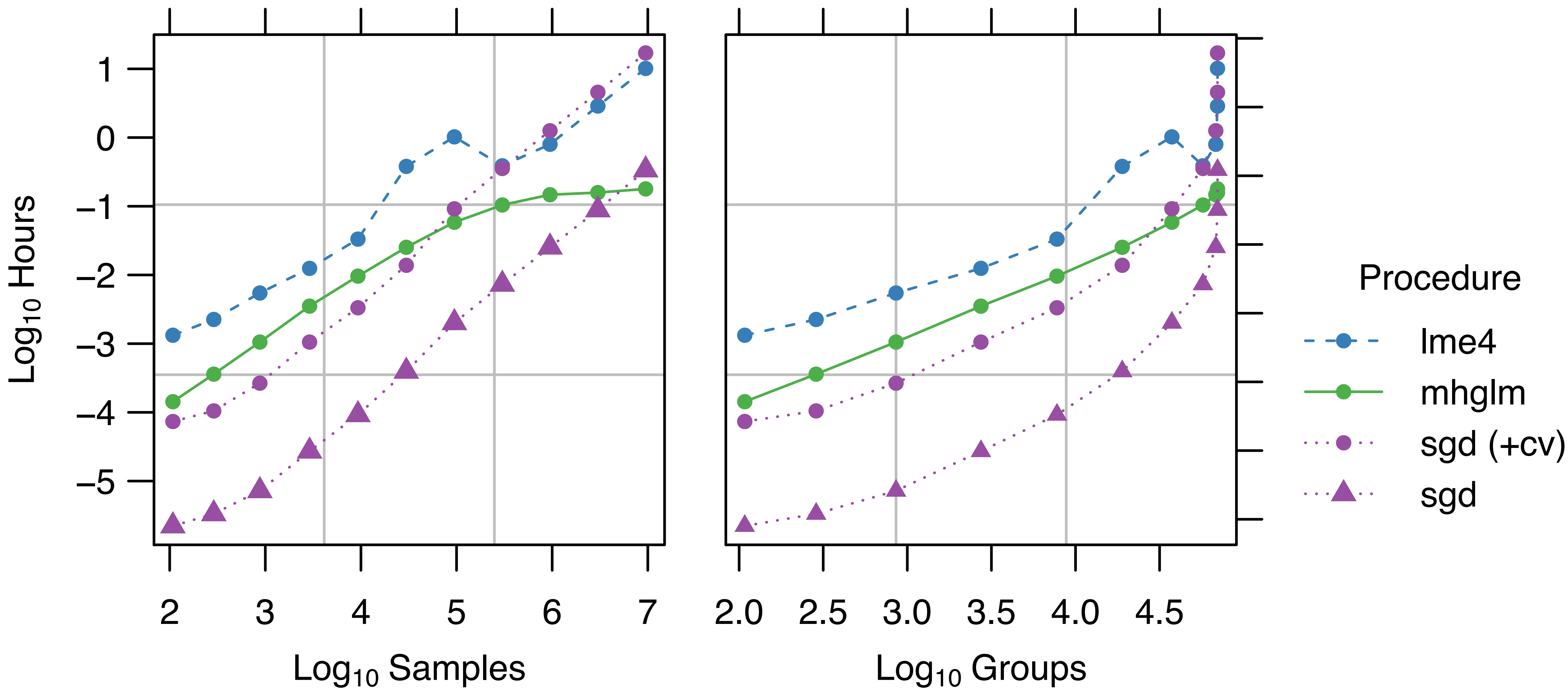
Pro: non-iterative (typically faster than ML), trivially parallelizable

Con: loss in statistical efficiency (sometimes)

Comparison

Method	Initial Cost	Cost per Iteration	Iterations
Expectation-Maximization	Np^2	Mp^3	Hundreds
Newton-Raphson	Np^2	Mp^4	Tens
Profile Likelihood (lme4)	Np^2	Mp^3	Tens to Hundreds
Stochastic Gradient Descent	0	Np	Tens
Moment-Based	Np^2	Mp^3	2

Computation Time



Remainder of the Talk

1. Moment-based as fast alternative to likelihood-based estimation
2. Consistent, asymptotically normal
3. Performs well in practice

Moment-based estimation: A new (old) estimation method, dramatically faster

History: Cochran (1937), Yates and Cochran (1938), Cochran (1954), Swamy (1970), Carter and Yang (1986), Cox and Solomon (2002)

Intuition for Moment-Based Estimation

1. Compute group-specific coefficient estimates

2. Estimate population parameters by matching coefficient moments.

Intuition for Moment-Based Estimation (Details)

1. Model:

$$y_i = X_i \beta_i + \varepsilon_i \qquad \beta_i \sim \mathrm{N}(\mu, \Sigma), \;\; \varepsilon_i \sim \mathrm{N}(0, \sigma^2 I)$$

2. Group-specific coefficient estimates:

$$b_i = (X_i^\top X_i)^{-1} X_i^\top y_i$$

3. Moments:

$$\mathrm{E}(b_i) = \mu \qquad \mathrm{Cov}(b_i) = \Sigma + \sigma^2 (X_i^\top X_i)^{-1}$$

Problem: Rank-Degenerate X

Group-specific coefficient estimates:

$$b_i = (X_i^\top X_i)^\dagger X_i^\top y_i$$

Biased estimate:

$$E(b_i \mid \beta_i) \neq \beta_i$$

Solution for Rank-Degenerate X

1. Group-specific coefficient estimates:

$$b_i = (X_i^\top X_i)^\dagger X_i^\top y_i$$ *(biased in nullspace of X_i)*

2. Choose weight matrices:

$$W_i$$ *(same nullspace as X_i)*

3. Moments:

$$\mathrm{E}(W_i b_i) = W_i \mu \qquad \mathrm{Cov}(W_i b_i) = W_i \{\Sigma + \sigma^2 (X_i^\top X_i)^\dagger\} W_i^\top$$

Moment Matching for Mean

$$\hat{\mu} = \Omega_1^{-1} \sum_{i=1}^{M} W_i b_i$$

$$\Omega_1 = \sum_{i=1}^{M} W_i$$

Moment Matching for Covariance

$$\hat{A} = \sum_{i=1}^{M} W_i (b_i - \hat{\mu})(b_i - \hat{\mu})^T W_i$$

$$\mathrm{vec}(\hat{\Sigma}) = \Omega_2^{-1} \mathrm{vec}(\hat{A}) - \mathrm{Bias}$$

$$\Omega_2 = \sum_{i=1}^{M} W_i \otimes W_i$$

Optimal Weights

$$W_i = U_i[U_i^\top\{\Sigma + \sigma^2(X_i^\top X_i)^\dagger U_i]^{-1}U_i^\top$$

Orthonormal columns,
same span as $X_i^\top$

Unknown parameters

Computational Complexity

- Compute M group-specific estimates: $O(Np^2)$
- Match weighted moments: $O(Mp^3)$

Theoretical Properties

Theorem 1: Moment-Based Estimates are Consistent

Theorem 1: Details

Statement:

$$\hat{\mu} = \mu + O_P(\|\Omega_1^{-1}\|^{1/2})$$

$$\hat{\Sigma} = \Sigma + O_P(\|\Omega_2^{-1}\|^{1/2})$$

Main Assumption:

group-specific estimates have finite fourth moments

Proof:

Linear Algebra + Markov's Inequality

Theorem 2: Two-Step Moment Based Estimates are Asymptotically Relatively Efficient

Theorem 2: Details

Statement:

$$\hat{\mu}_{\text{two-step}} = \hat{\mu}_{\text{optimal}} + o_P(\|\Omega_1^{-1}\|^{1/2})$$

Assumptions:

(same as Theorem 1)

Proof:

Taylor expansion / Matrix inverse perturbation

Theorem 3: Two-Step Moment Based Estimates are Asymptotically Normal

Theorem 3: Details

Statement:

$$\Omega_1^{1/2}(\hat{\mu} - \mu) \Longrightarrow \mathcal{N}(0, I)$$

Assumptions:

(same as Theorem 1 + large M)

Proof:

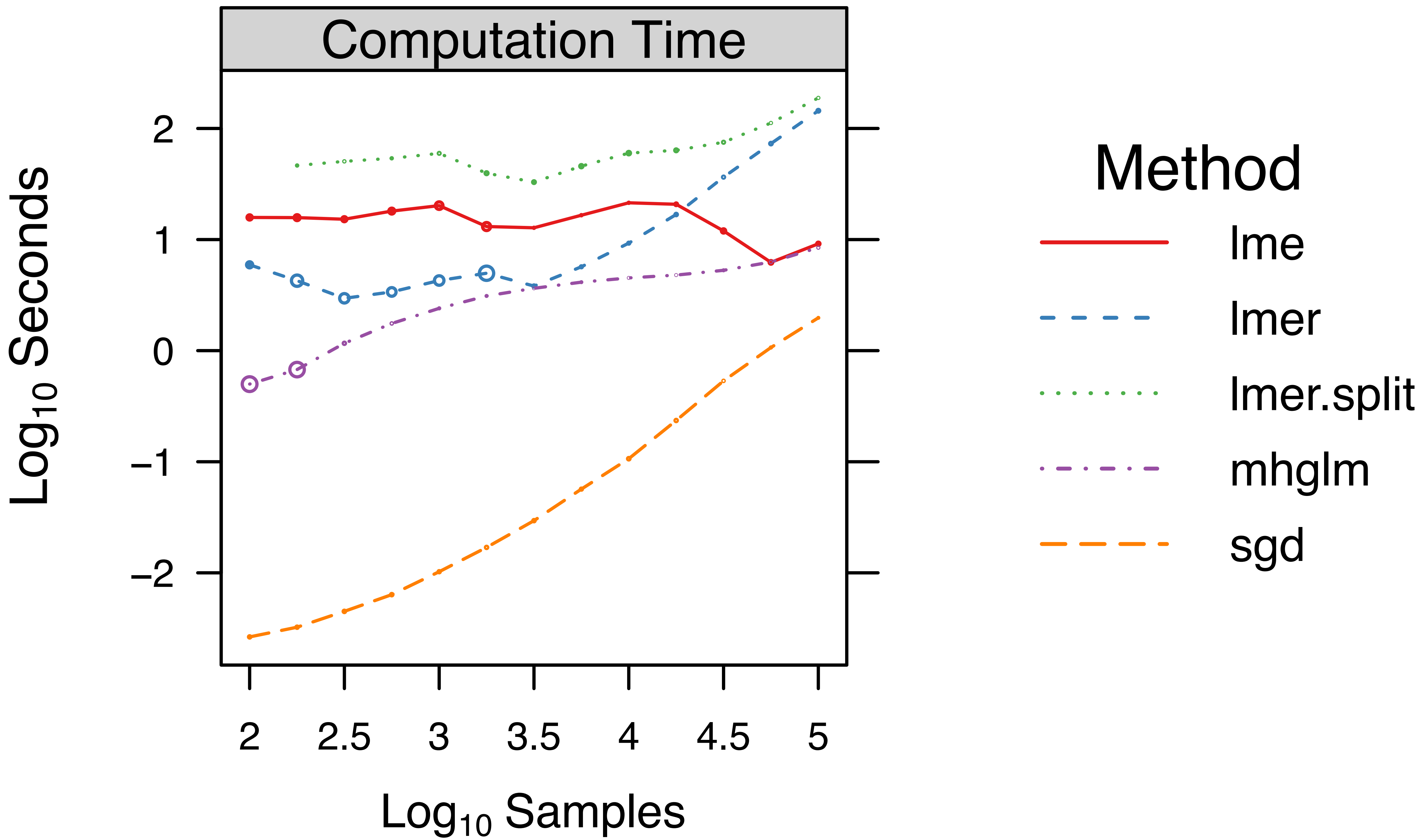
Central Limit Theorem

Theory: Recap

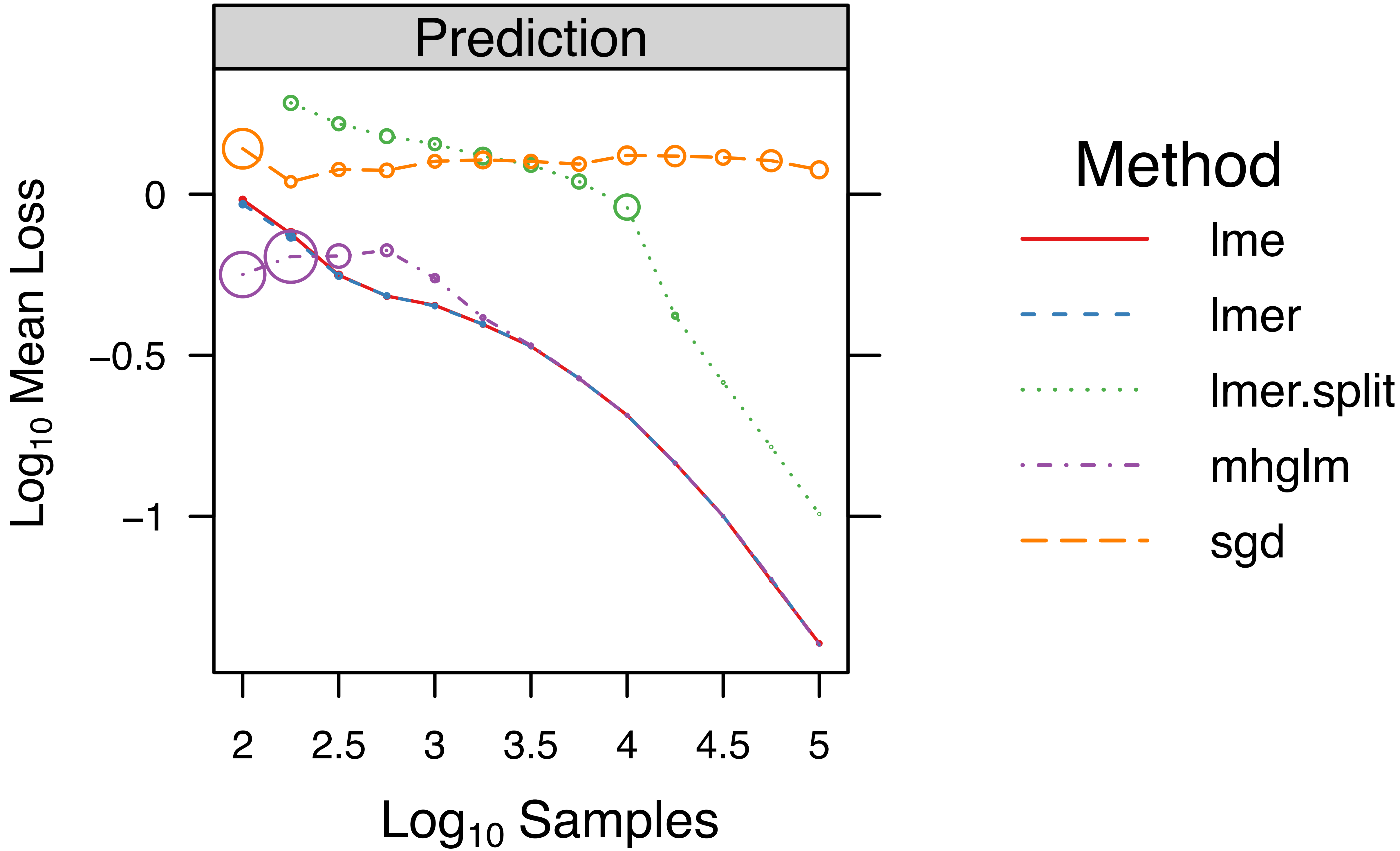
- Estimates are consistent
- Two-step estimates are asymptotically relatively efficient
- Estimates are asymptotically normal

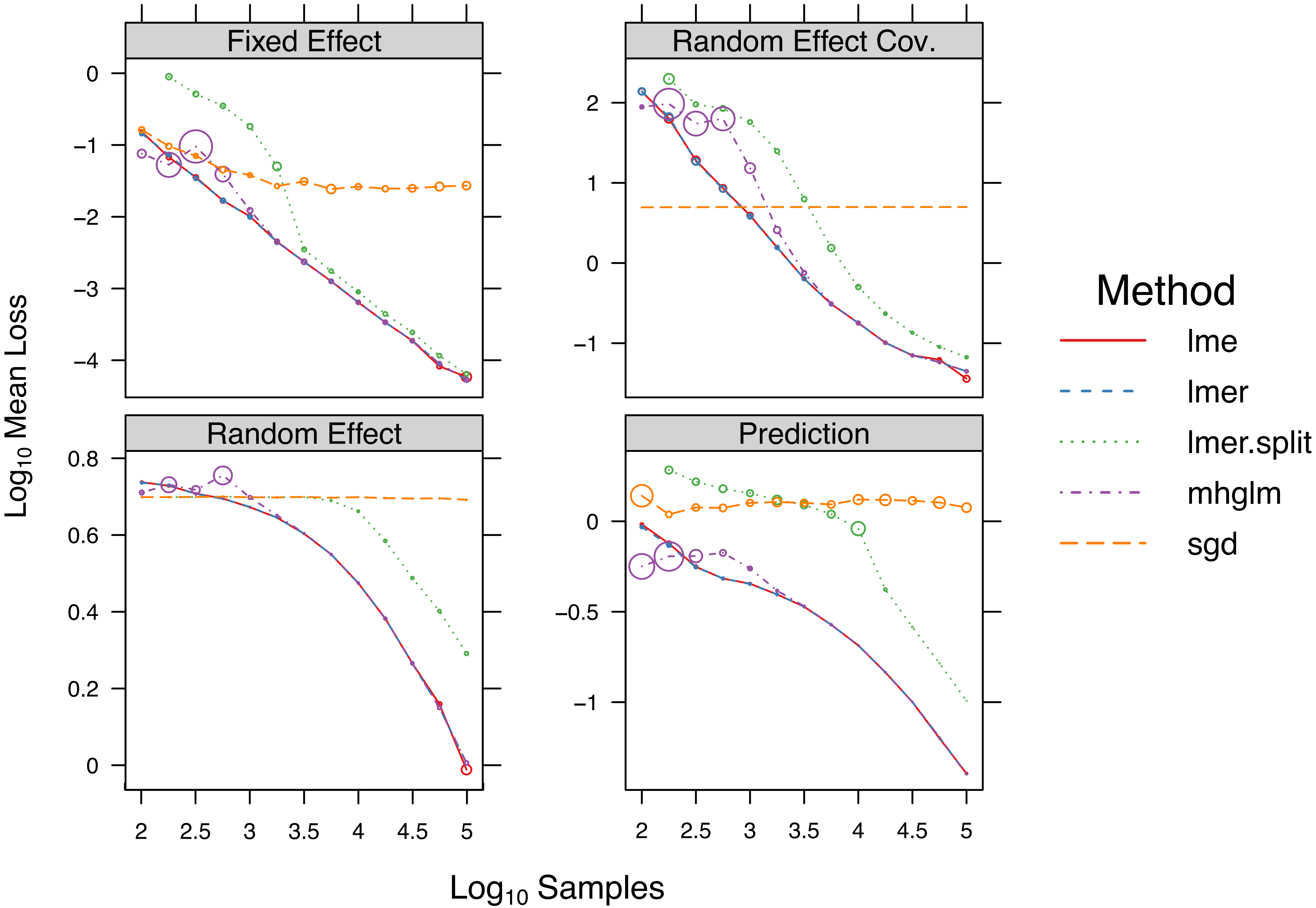
Performance in Hierarchical Linear Model Simulations

Linear Model: Time



Linear Model: Accuracy

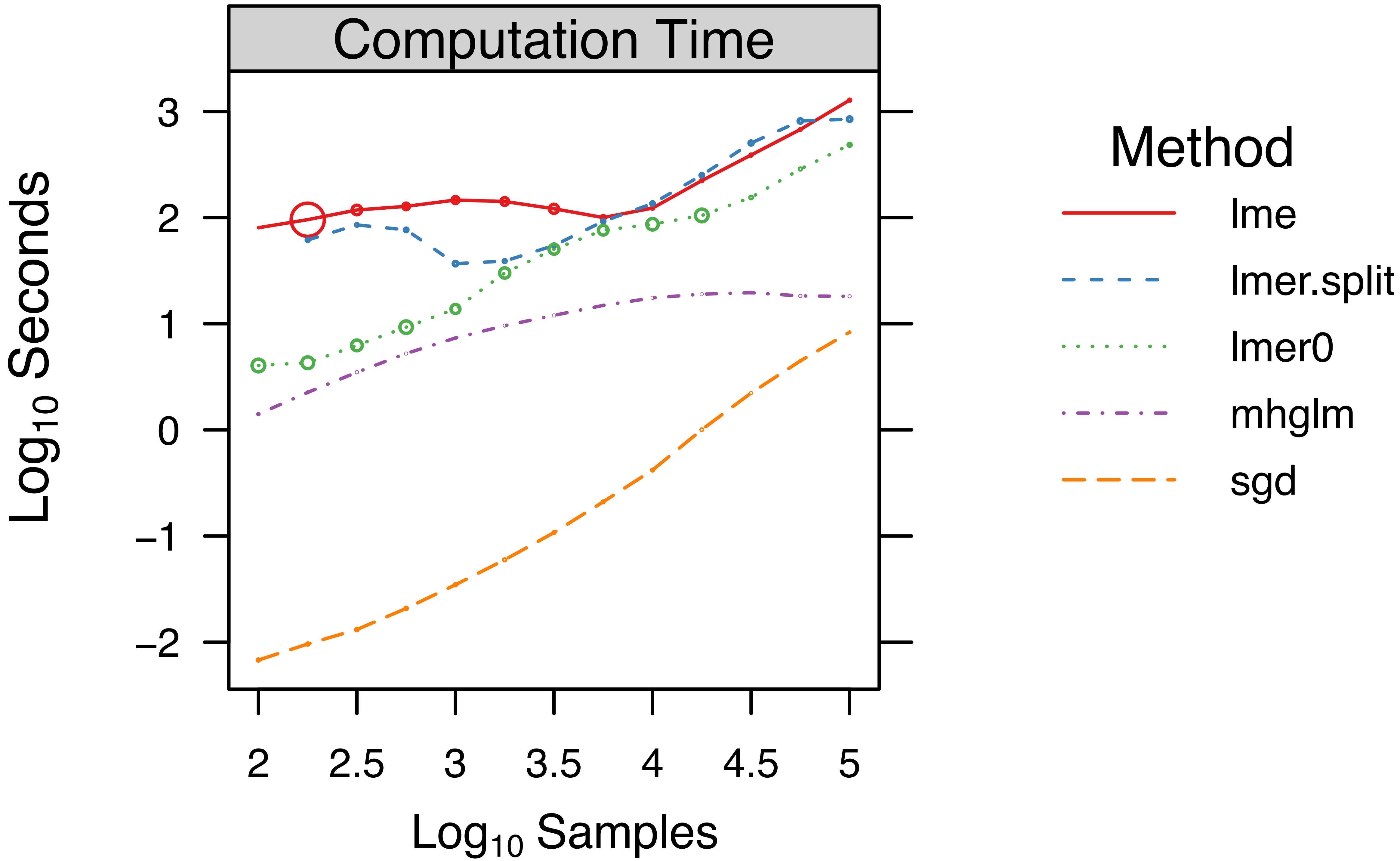




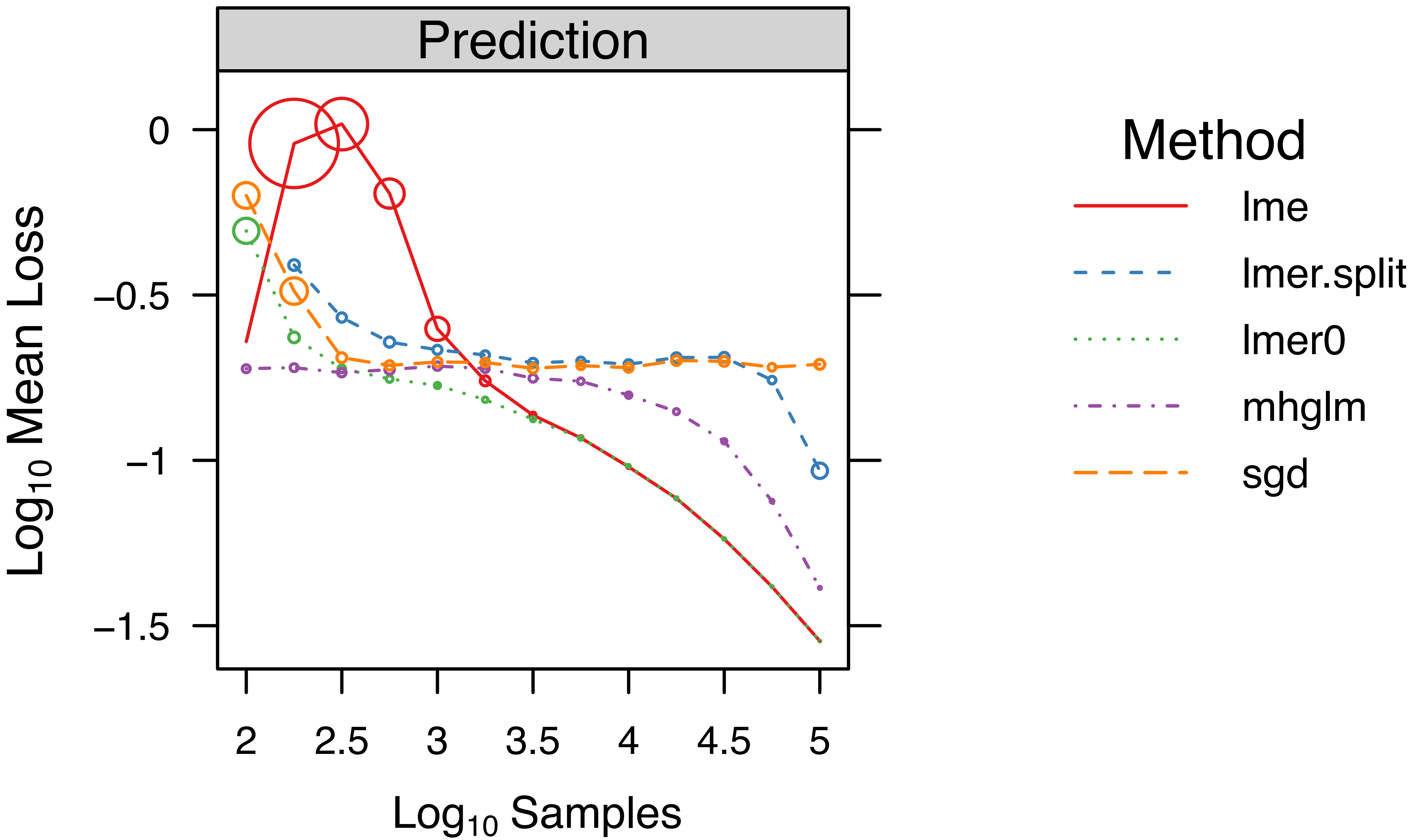
Fixed Effect
Random Effect Cov.
Random Effect
Prediction
Log_{10} Mean Loss
Log_{10} Samples
Method
lme
lmer
lmer.split
mhglm
sgd

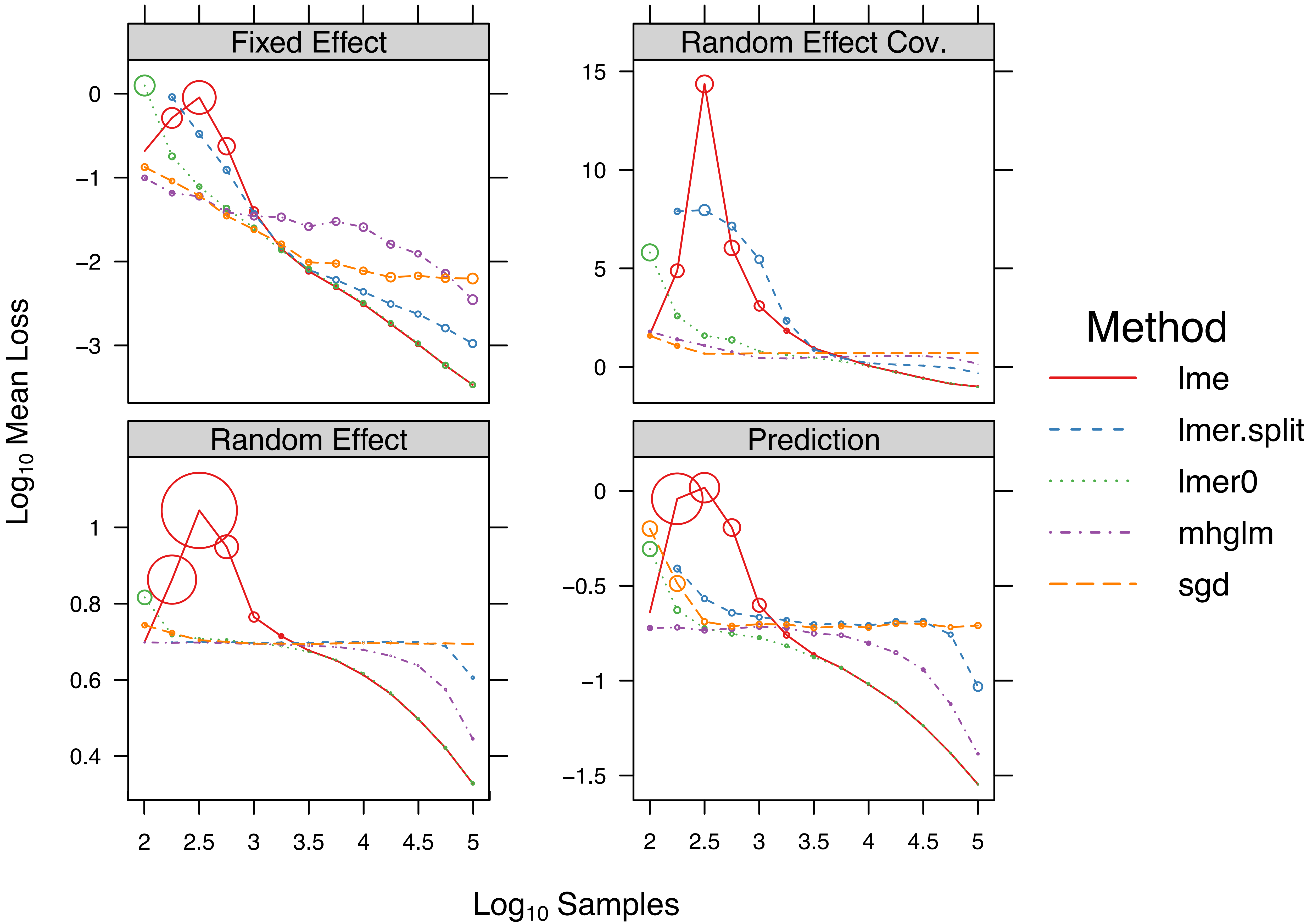
Performance in Hierarchical Logistic Model Simulations

Logistic Model: Time



Logistic Model: Accuracy





Fixed Effect
Random Effect Cov.
Random Effect
Prediction
Method
lme
lmer.split
lmer0
mhglm
sgd
Log_{10} Mean Loss
Log_{10} Samples
0
−1
−2
−3
15
10
5
0
1
0.8
0.6
0.4
0
−0.5
−1
−1.5
2
2.5
3
3.5
4
4.5
5

Performance in Practice

Application: MovieLens 10M

MovieLens 10M dataset:

- 10 million ratings
- 70 thousand users
- 10 thousand movies
- Predictors: genre, item popularity, user mood

Time to fit with *glmer*:

10 hours

Time to fit with *mhglm*:

10 minutes

Response

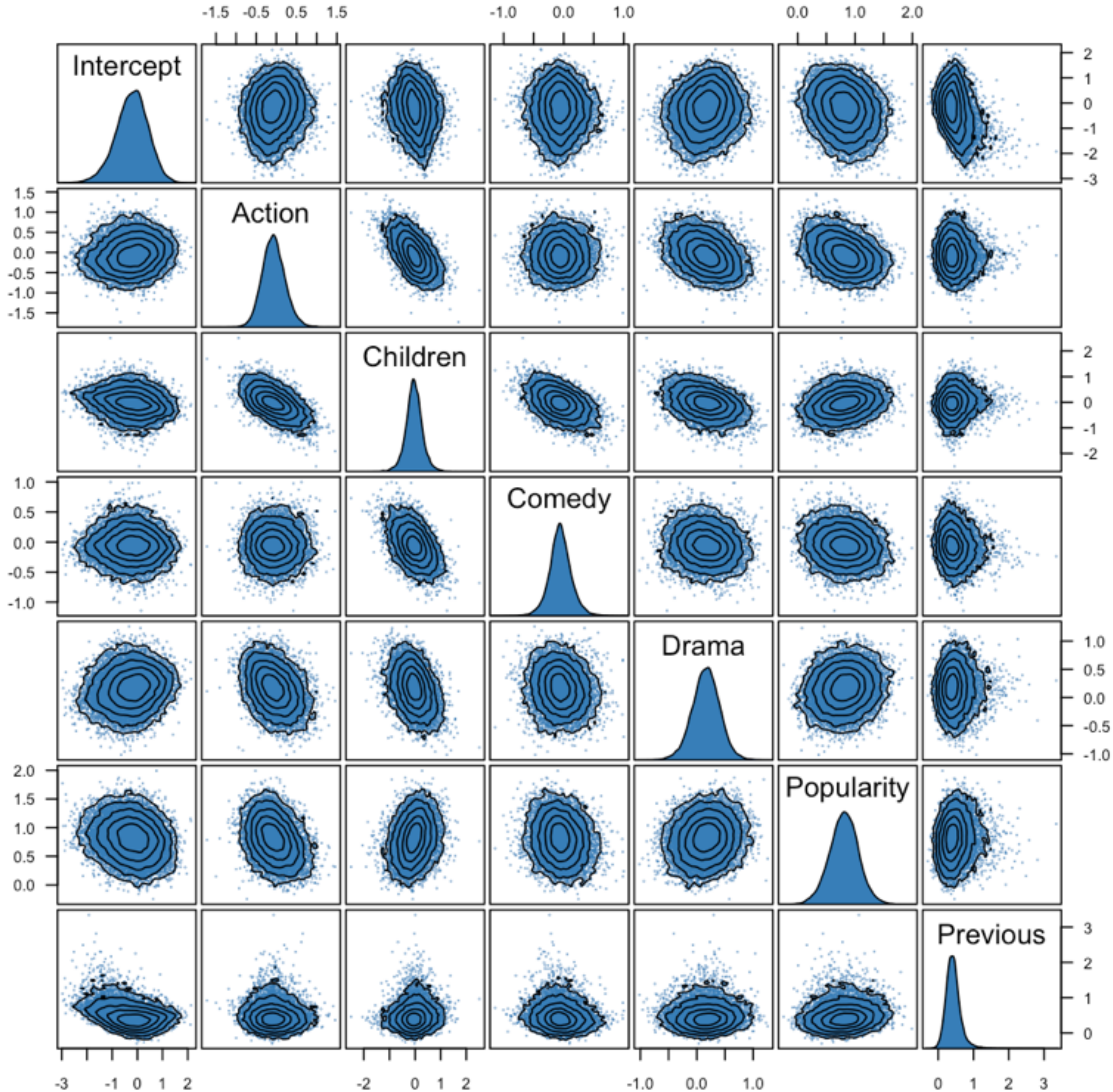
- Given that user i rates movie j:

  y_{ij} = rating is positive (4 or 5 stars)

- Ratings per user: 140 (ranges from10 to1000)

Predictors

- Genre: Action, Children, Comedy, or Drama
- Popularity: logit(current popularity of movie), computed from 30 most recent ratings
- Previous: indicator of whether user's previous review was positive

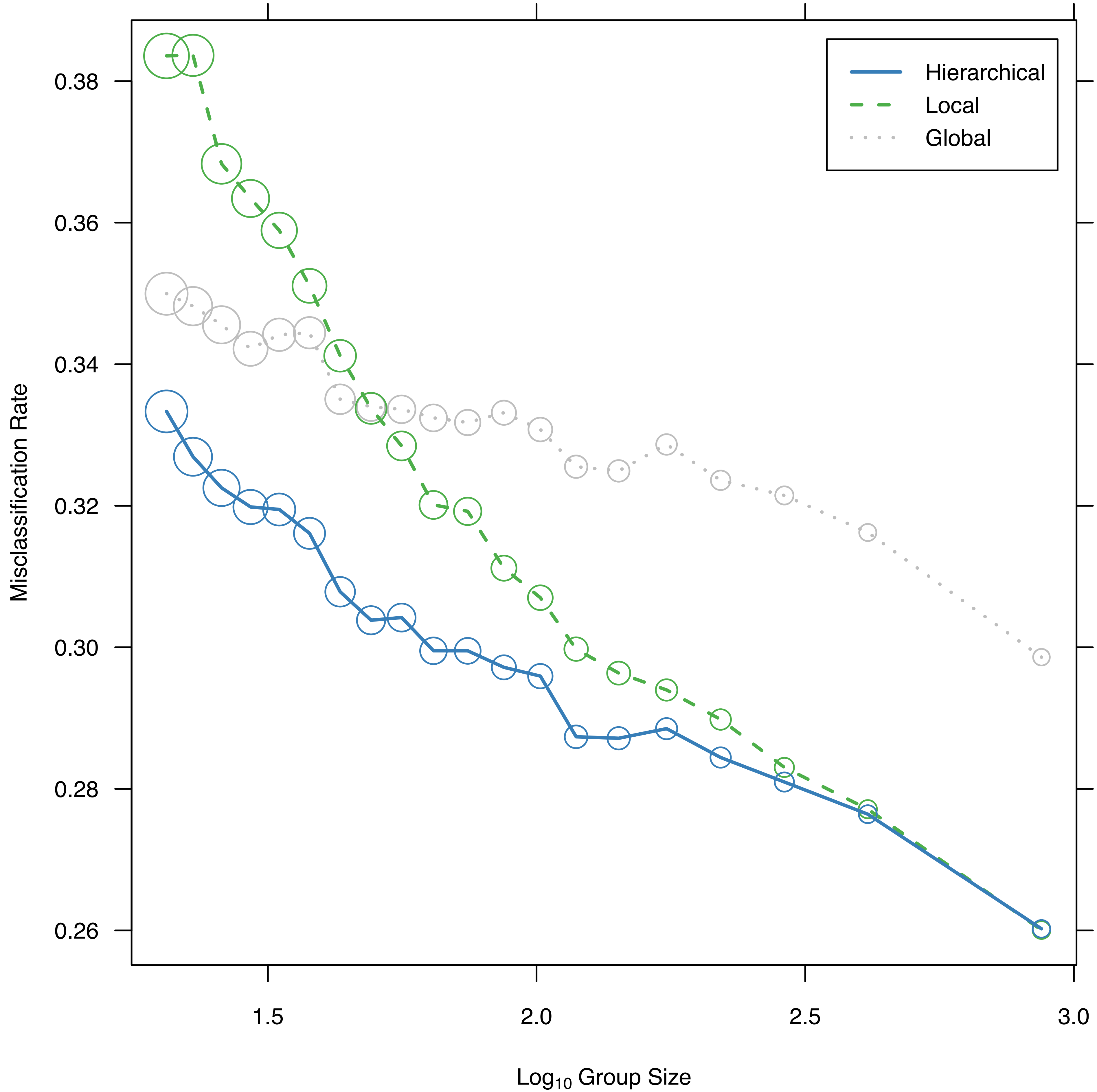


Intercept
Action
Children
Comedy
Drama
Popularity
Previous

Out-of-Sample Performance

Table 3: MovieLens test set error.

Method	Loss Function		
	Log	Squared Error	Misclassification
Hierarchical	0.55	0.18	0.28
Local	0.57	0.19	0.29
Global	0.59	0.20	0.32


Hierarchical
Local
Global
Misclassification Rate
0.38
0.36
0.34
0.32
0.30
0.28
0.26
1.5
2.0
2.5
3.0
Log_{10} Group Size

Summary: Moment-Based Estimation

- Fast
- Theoretically Sound
- Works In Practice

R package: mbest

```
mhglm(y ~ genre + popularity
        + (1 + genre | user),
      family=binomial)
```

Available on CRAN

Thank you!