Some computational and modeling issues for hierarchical models

Andrew Gelman

Dept of Statistics and Dept of Political Science, Columbia University, New York
(Visiting Sciences Po, Paris, for 2009–2010)

17 Oct 2009

6 challenges in statistical modeling

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model
- Regularization or partial pooling

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model
- Regularization or partial pooling
- Fitting the model

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model
- Regularization or partial pooling
- Fitting the model
- Checking the fit to data

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model
- Regularization or partial pooling
- Fitting the model
- Checking the fit to data
- Confidence building

6 challenges in statistical modeling

- Setting up a realistic (i.e., complicated) model
- Regularization or partial pooling
- Fitting the model
- Checking the fit to data
- Confidence building
- Understanding the fitted model

The models I'm fitting

- Hierarchical generalized linear models

The models I'm fitting

- Hierarchical generalized linear models
- $y_i = \alpha + \beta x_i + \epsilon_i$

The models I'm fitting

- Hierarchical generalized linear models
- $y_i = \alpha + \beta x_i + \epsilon_i$
- $y_i = \alpha_{j[i]} + \beta_{j[i]} x_i + \epsilon_i$ (separate regression in each group)

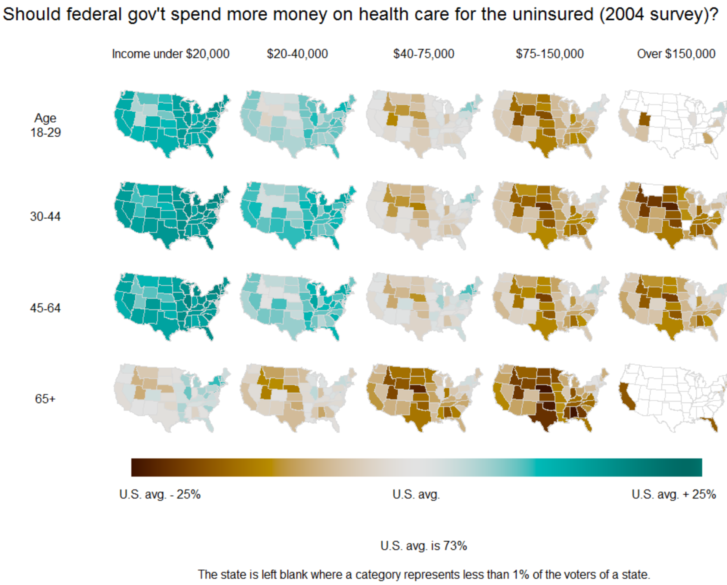
The models I'm fitting

- Hierarchical generalized linear models
- $y_i = \alpha + \beta x_i + \epsilon_i$
- $y_i = \alpha_{j[i]} + \beta_{j[i]} x_i + \epsilon_i$ (separate regression in each group)
- $\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \right)$, for $j = 1, \ldots, J$

The models I'm fitting

- Hierarchical generalized linear models
- $y_i = \alpha + \beta x_i + \epsilon_i$
- $y_i = \alpha_{j[i]} + \beta_{j[i]} x_i + \epsilon_i$ (separate regression in each group)
- $\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \right)$, for $j = 1, \ldots, J$
- Also can have group-level predictors and nonnested grouping factors

Application: public opinion in population subgroups



Software options

Software options

- R, Matlab, etc. (data processing and graphics, general computing)

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)
- What's missing?

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)
- What's missing?

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)
- What's missing?

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)
- What's missing?

Software options

- R, Matlab, etc. (data processing and graphics, general computing)
- C, Fortran, Python, etc. (fast computation)
- Bugs etc.
- Specialized multilevel modeling software (e.g., lmer in R)
- What's missing?
  - Something in between "automatic" and "program it yourself"

Some issues with Bugs, OpenBugs, Jags, etc.

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging
  - Solution: support for predictive model checking

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging
  - Solution: support for predictive model checking
- Slow convergnce for moderate to large datasets

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging
  - Solution: support for predictive model checking
- Slow convergnce for moderate to large datasets
  - Solution: make use of hierarchical structure

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging
  - Solution: support for predictive model checking
- Slow convergnce for moderate to large datasets
  - Solution: make use of hierarchical structure
- Can run slowly and even crash

Some issues with Bugs, OpenBugs, Jags, etc.

- Difficult to keep track of multiple versions of a model
  - Solution: allow unused parameters and data
- Code gets long and ugly
  - Solution: allow subroutines and macros
- How long to run the program?
  - Solution: automatic convergence check
- Lack of confidence in results. ("My model converged. Now what?")
  - Solution: automatic fake-data debugging
  - Solution: support for predictive model checking
- Slow convergnce for moderate to large datasets
  - Solution: make use of hierarchical structure
- Can run slowly and even crash
  - Solution: allow the sophisticated user/developer to "get under the hood" and fix problems

Need to have subroutines!

Need to have subroutines!

- For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[state[i]] + b[state[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
```

Need to have subroutines!

- For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[state[i]] + b[state[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
```

- We would prefer:

```
y ~ norm (a[state] + b[state]*x, sigma.y)
```

Need to have subroutines!

- For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[state[i]] + b[state[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
```

- We would prefer:

```
y ~ norm (a[state] + b[state]*x, sigma.y)
```

- And it gets worse when dimension > 2

Multiplicative redundant parameterization

Multiplicative redundant parameterization

- New code with extra parameters ξ_a, ξ_b:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- xi.a*a[state[i]] + xi.b*b[state[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
xi.a ~ dnorm (0, .01)
xi.b ~ dnorm (0, .01)
for (j in 1:J){
  a.adj[j] <- xi.a*a[j]
  b.adj[j] <- xi.b*b[j]
}
```

Multiplicative redundant parameterization

- New code with extra parameters ξ_a, ξ_b:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- xi.a*a[state[i]] + xi.b*b[state[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 100)
xi.a ~ dnorm (0, .01)
xi.b ~ dnorm (0, .01)
for (j in 1:J){
  a.adj[j] <- xi.a*a[j]
  b.adj[j] <- xi.b*b[j]
}
```

- We would prefer:

```
y ~ norm (a[state] + b[state]*x, sigma.y)
```

Priors needed for hierarchical variance parameters

Priors needed for hierarchical variance parameters

- Problems

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates
    - Priors for variances that are bounded away from 0

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates
    - Priors for variances that are bounded away from 0
    - Priors for covariance matrices that are bounded away from degeneracy

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates
    - Priors for variances that are bounded away from 0
    - Priors for covariance matrices that are bounded away from degeneracy
  - Weakly-informative priors for full Bayes

Priors needed for hierarchical variance parameters

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates
    - Priors for variances that are bounded away from 0
    - Priors for covariance matrices that are bounded away from degeneracy
  - Weakly-informative priors for full Bayes
    - Half-Cauchy priors for hierarchical variances

- Problems
  - Variances estimated at 0
  - Correlations estimated at ± 1; non-positive-definite covariance matrices
- Solutions
  - Regularization priors for point estimates
    - Priors for variances that are bounded away from 0
    - Priors for covariance matrices that are bounded away from degeneracy
  - Weakly-informative priors for full Bayes
    - Half-Cauchy priors for hierarchical variances
    - Scaled-inverse-Wishart priors for covariances: $\Sigma_B = \text{Diag}(\xi) Q \text{Diag}(\xi)$

Models for deep interactions

Models for deep interactions

- Main effects, 2-way, 3-way, etc.

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states
- $4 + 5 + 50 + 4 \times 5 + 4 \times 50 + 5 \times 50 + 4 \times 5 \times 50$ parameters (“effects”)

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states
- $4 + 5 + 50 + 4 \times 5 + 4 \times 50 + 5 \times 50 + 4 \times 5 \times 50$ parameters ("effects")
- Also, group-level predictors (linear trends for age and income, previous voting patterns for states)

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states
- $4+5+50+4\times 5+4\times 50+5\times 50+4\times 5\times 50$ parameters ("effects")
- Also, group-level predictors (linear trends for age and income, previous voting patterns for states)
- Need a richer modeling language

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states
- $4 + 5 + 50 + 4 \times 5 + 4 \times 50 + 5 \times 50 + 4 \times 5 \times 50$ parameters ("effects")
- Also, group-level predictors (linear trends for age and income, previous voting patterns for states)
- Need a richer modeling language
  - ```
    glmer (y ~ z.age*z.inc*rvote.st + (z.age*z.inc | st) +
      (z.age*rvote.st | inc) + (z.inc*rvote.st | age) +
      (z.age | inc*st) + (z.inc | age*st) + (z.st |age*inc) +
      (1 | age*inc*st), family=binomial(link="logistic"))
    ```
    ```

Models for deep interactions

- Main effects, 2-way, 3-way, etc.
- Example: predicting public opinion given 4 age categories, 5 income categories, 50 states
- $4 + 5 + 50 + 4 \times 5 + 4 \times 50 + 5 \times 50 + 4 \times 5 \times 50$ parameters ("effects")
- Also, group-level predictors (linear trends for age and income, previous voting patterns for states)
- Need a richer modeling language
  - ```
    glmer (y ~ z.age*z.inc*rvote.st + (z.age*z.inc | st) +
     (z.age*rvote.st | inc) + (z.inc*rvote.st | age) +
     (z.age | inc*st) + (z.inc | age*st) + (z.st |age*inc) +
     (1 | age*inc*st), family=binomial(link="logistic"))
    ```
  - No easy way to write this in Bugs or to program it oneself!
    ```

Automatic confidence building

Automatic confidence building

- Monitoring convergence

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data
- Model checking

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data
- Model checking
  - Solution: posterior predictive simulation by default

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data
- Model checking
  - Solution: posterior predictive simulation by default
  - For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.rep[i] <- dnorm (y.hat[i], tau.y)
  . . .
```

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data
- Model checking
  - Solution: posterior predictive simulation by default
  - For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.rep[i] <- dnorm (y.hat[i], tau.y)
  . . .
```

  - But y^{rep} should be included automatically

Automatic confidence building

- Monitoring convergence
  - Solution: check mixing within and between chains; it's easy
- Fake-data debugging
  - Solution: set up Bugs (or whatever) to automatically run itself with simulated data
- Model checking
  - Solution: posterior predictive simulation by default
  - For example:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.rep[i] <- dnorm (y.hat[i], tau.y)
  . . .
```

  - But y^{rep} should be included automatically
  - Implicit graphical structure for model checking: y—θ—y^{rep}

Où allons-nous?

Où allons-nous?

- Work in parallel on different software

Où allons-nous?

- Work in parallel on different software
- lmer/glmer

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?
- Bugs

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?
- Bugs
  - Subroutines, automatic monitoring, other improvements discussed above

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?
- Bugs
  - Subroutines, automatic monitoring, other improvements discussed above
- HBC (hierarchical Bayes compiler)

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?
- Bugs
  - Subroutines, automatic monitoring, other improvements discussed above
- HBC (hierarchical Bayes compiler)
  - Closer to programming it myself

Où allons-nous?

- Work in parallel on different software
- lmer/glmer
  - Reglarizing priors
  - Approximate full Bayes (a few steps of Metropolis)
  - Use congjugate gradient to improve speed?
- Bugs
  - Subroutines, automatic monitoring, other improvements discussed above
- HBC (hierarchical Bayes compiler)
  - Closer to programming it myself
  - A better modeling language?

Vos pensées??

- Where to go on Bugs?
- How to work efficiently when so many research groups around the world are fitting these models?
- How to move from "The program converged!" to "The model makes sense"?