Toward an environment for Bayesian data analysis in R

Andrew Gelman

8 August 2004

Bayes in R

- I'm a software *user*
- This is my chance to reach the software *developers*!
- I want the best of R, BUGS, and graphical models
- Collaborators:
  - Jouni Kerman, Dept of Statistics, Columbia University
  - (implicitly) the developers of Bugs and R

Bayes in R

- I'm a software *user*
- This is my chance to reach the software *developers*!
- I want the best of R, BUGS, and graphical models
- Collaborators:
  - Jouni Kerman, Dept of Statistics, Columbia University
  - (implicitly) the developers of Bugs and R

Bayes in R

- I'm a software *user*
- This is my chance to reach the software *developers*!
- I want the best of R, BUGS, and graphical models
- Collaborators:
  - Jouni Kerman, Dept of Statistics, Columbia University
  - (implicitly) the developers of Bugs and R

Bayes in R

- I'm a software *user*
- This is my chance to reach the software *developers*!
- I want the best of R, BUGS, and graphical models
- Collaborators:
  - Jouni Kerman, Dept of Statistics, Columbia University
  - (implicitly) the developers of Bugs and R

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Graphical models and Bayesian data analysis

- My view of graphical models:
  - Bayesian data analysis
  - Structured model (not simply $p(\theta), p(y|\theta), p(\theta|y)$)
  - I think of hierarchical (multilevel) models
  - But also time series, spatial, networks, etc.
- BDA: goal is model building and checking, not just "inference"
- Connection between graphical modeling and Bayesian model checking and debugging

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Options for Bayesian data analysis in R

- Why R?
  - Flexibility for data analysis and simulation
  - Open-source
- Programming it yourself (in R or Fortran/C)
- Setting it up in a Gibbs/Metropolis environment (Kerman's `UMACS`)
- Specialized programs for specific models (e.g., Martin and Quinn's `MCMCpack`)
- BUGS (as called from R)

Fully Bayesian computing

- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length $\tilde{n}$
- Goal: direct manipulation of random vectors and arrays

Fully Bayesian computing

- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length n
- Goal: direct manipulation of random vectors and arrays

Fully Bayesian computing

- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length n
- Goal: direct manipulation of random vectors and arrays

Fully Bayesian computing

- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length n
- Goal: direct manipulation of random vectors and arrays

Fully Bayesian computing

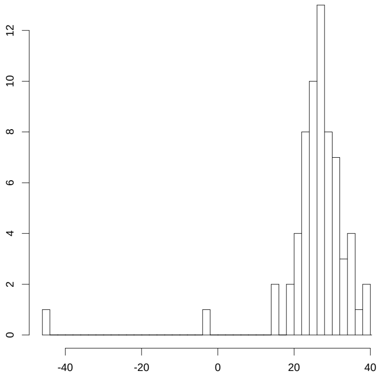
- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length n
- Goal: direct manipulation of random vectors and arrays

Fully Bayesian computing

- All unknowns are random variables
- Potential randomness is implicit in all "random variable" objects
- Example: regression $y = X\beta + \epsilon$, predictions $\tilde{y}$ for new data $\tilde{X}$
  - β is a random vector of length k (uncertainty from regression estimation)
  - $\tilde{y}$ is a random vector of length n
- Goal: direct manipulation of random vectors and arrays

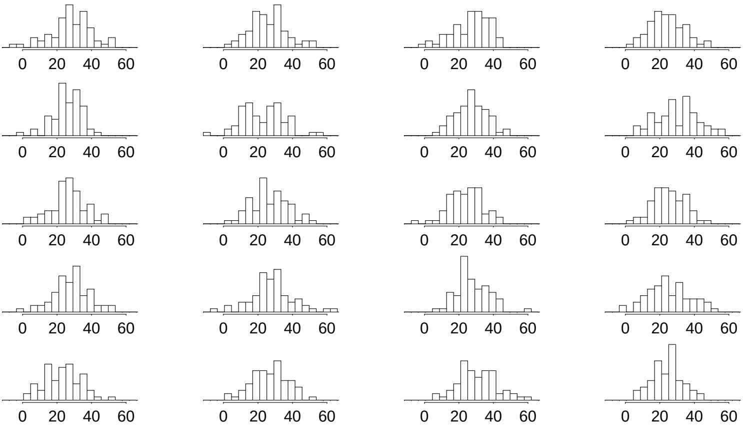
Data y, fit to a normal distribution

```
> hist (y)
```



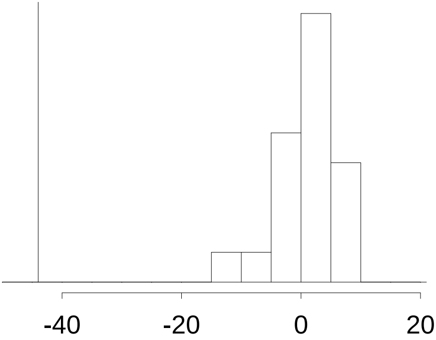
20 posterior predictive replications y^{rep}

```
> hist (y.rep)
```



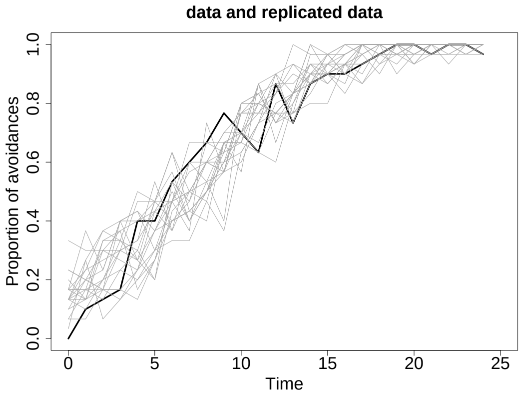
The test statistic, $T(y) = \min_{i=1}^{n} y_i$

```
> rv.hist (T(y), T(y.rep))
```



Another example of a posterior predictive check

```
> plot (y, type="l")
> lines (y.rep)
```



Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Operations with random variables

- Summaries: means, quantiles, etc.
- Plots
- Predictive checking
- No awkward syntax; e.g., we want to say `beta[1]`, not `beta[,1]`
- Some open questions (e.g., how to make plots that show posterior uncertainty)
- More in Jouni Kerman's talk Thursday morning

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\rm rep}|\theta)$
  - Compare y to $y^{\rm rep}$ using (graphical) test variables
  - Graphical structure: y—θ—$y^{\rm rep}$
- More general formulation
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Checking graphical models through predictive replications

- Quick summary of posterior predictive checking
  - Data y, inference from $p(\theta|y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|\theta)$
  - Compare y to y^{rep} using (graphical) test variables
  - Graphical structure: y—θ—y^{rep}
- More general formulation
  - Data y, inference from $p(\theta|X, y)$
  - Predictive replications from $p(y^{\mathrm{rep}}|X, \theta)$
- Connection to graphical models!

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Predictive checking and graphical models

- A posterior predictive check requires:
  - Set of conditioning variables θ
  - Set of fixed design variables X (e.g., sample size)
  - Test variable $T(y)$ (more generally, $T(X, y, \theta)$)
- Simulating posterior predictive replications is a fundamental operation in graphical models
- Requires a new node, y^{rep} whose distribution is implied by the existing model

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Fake-data debugging

- Models can be debugged by simulating fake data:
  - Sample θ^{true} from the prior distribution $p(\theta)$
  - Sample y from the model $p(y|\theta)$
  - Perform Bayesian inference, simulations from $p(\theta|y)$
  - Check calibration of posterior means, predictive intervals, etc. compared to θ^{true}
- Fake-data simulation is a fundamental operation in graphical models
- θ^{true} is a new node

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Predictive checking and fake-data debugging

- Model checking or debugging in ideal graphical model software ("DreamBUGS"):
  - Set an on/off switch for each node: is it conditioned on or averaged over?
  - Specify a test summary (numerical or graphical) of data and parameters
  - Various off-the-shelf test summaries will be available
  - Run and look at the results!
- Design of data collection is integrated with graphical modeling

Great things about BUGS

- It really works! I use it in my own applied research
- Easy to use and to teach, intuitive syntax
- Free
- Can be called directly from R

Great things about BUGS

- It really works! I use it in my own applied research
- Easy to use and to teach, intuitive syntax
- Free
- Can be called directly from R

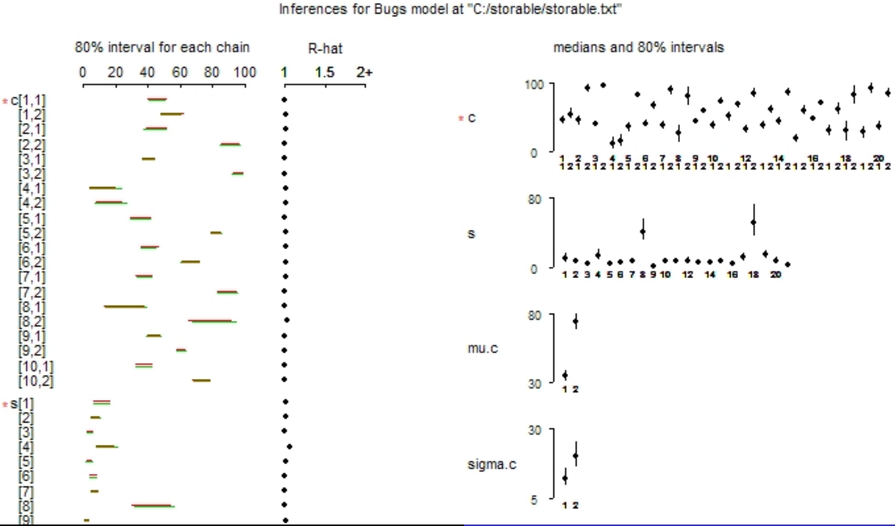
Great things about BUGS

- It really works! I use it in my own applied research
- Easy to use and to teach, intuitive syntax
- Free
- Can be called directly from R

Great things about BUGS

- It really works! I use it in my own applied research
- Easy to use and to teach, intuitive syntax
- Free
- Can be called directly from R

Running Bugs from R



Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Problems with BUGS

- Often needs lots of "hand-holding" to work
- Efficiently-programmed models can get really long
- Can't debug by running interactively (as in R)
- Need to use work-arounds when it crashes
- Not open-source; can't go inside and improve/fix it
- You have to stop it to check convergence

Potential improvements to BUGS

- Functions or macros
  - Instead of:
    ```
    for (i in 1:n){
      y[i] ~ dnorm (y.hat[i], tau.y)
      y.hat[i] <- a[county[i]] + b[county[i]]*x[i]
      e.y[i] <- y[i] - y.hat[i]
    }
    tau.y <- pow(sigma.y, -2)
    sigma.y ~ dunif (0, 1000)
    ```
  - We want something like:
    ```
    y ~ norm (a[county] + b[county]*x, sigma.y)
    ```
- Lots more examples

Potential improvements to BUGS

- Functions or macros
  - Instead of:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[county[i]] + b[county[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 1000)
```

  - We want something like:

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

- Lots more examples

Potential improvements to BUGS

- Functions or macros
  - Instead of:

```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[county[i]] + b[county[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 1000)
```

  - We want something like:

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

- Lots more examples

Potential improvements to BUGS

- Functions or macros
  - Instead of:
```
for (i in 1:n){
  y[i] ~ dnorm (y.hat[i], tau.y)
  y.hat[i] <- a[county[i]] + b[county[i]]*x[i]
  e.y[i] <- y[i] - y.hat[i]
}
tau.y <- pow(sigma.y, -2)
sigma.y ~ dunif (0, 1000)
```
  - We want something like:
```
y ~ norm (a[county] + b[county]*x, sigma.y)
```
- Lots more examples

More potential improvements to BUGS

- Automatic convergence monitoring (run until the sequences have mixed)
- Model building, using simulations from previous simpler models as starting points
- Correlation modeling (e.g., Daniels/Kass, Barnard/Meng/McCulloch)
- Automatic data subsetting
- Going beyond the "production run" mentality

More potential improvements to BUGS

- Automatic convergence monitoring (run until the sequences have mixed)
- Model building, using simulations from previous simpler models as starting points
- Correlation modeling (e.g., Daniels/Kass, Barnard/Meng/McCulloch)
- Automatic data subsetting
- Going beyond the "production run" mentality

More potential improvements to BUGS

- Automatic convergence monitoring (run until the sequences have mixed)
- Model building, using simulations from previous simpler models as starting points
- Correlation modeling (e.g., Daniels/Kass, Barnard/Meng/McCulloch)
- Automatic data subsetting
- Going beyond the "production run" mentality

More potential improvements to BUGS

- Automatic convergence monitoring (run until the sequences have mixed)
- Model building, using simulations from previous simpler models as starting points
- Correlation modeling (e.g., Daniels/Kass, Barnard/Meng/McCulloch)
- Automatic data subsetting
- Going beyond the "production run" mentality

More potential improvements to BUGS

- Automatic convergence monitoring (run until the sequences have mixed)
- Model building, using simulations from previous simpler models as starting points
- Correlation modeling (e.g., Daniels/Kass, Barnard/Meng/McCulloch)
- Automatic data subsetting
- Going beyond the "production run" mentality

Graphical models for Bayesian data analysis

- Direct computation and graphing of “random variable” objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating

Graphical models for Bayesian data analysis

- Direct computation and graphing of "random variable" objects
- Work with posterior *simulations*, not means and medians
- Generalization to model checking and fake-data debugging
  - Should be easy to do
  - Generalizes graphical model structure
- Using graphical models as a structure for building up from simple models
- Open source to allow modules for Gibbs and Metropolis updating