Bayesian Structural Equation Models for Cumulative Theory Building in Information Systems

A Brief Tutorial using BUGS and R

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca

December 5, 2014

This tutorial is based on the following publication:

- Evermann, Joerg and Tate, Mary (2014).
  Bayesian Structural Equation Models for Cumulative Theory Building in Information Systems – A Brief Tutorial using BUGS and R.
  Communications of the AIS, 34(1) , article 76.

Tutorial Materials

http://joerg.evermann.ca/bayesiantutorial.html

Software to Download and Install

- Zip file with examples
- R
- JAGS
- OpenBUGS

Windows

- Notepad++ (text editor, only if you do not have one)

MacOS X

- OpenBUGS not available for MacOS X

Linux

- Use your package manager to get the software

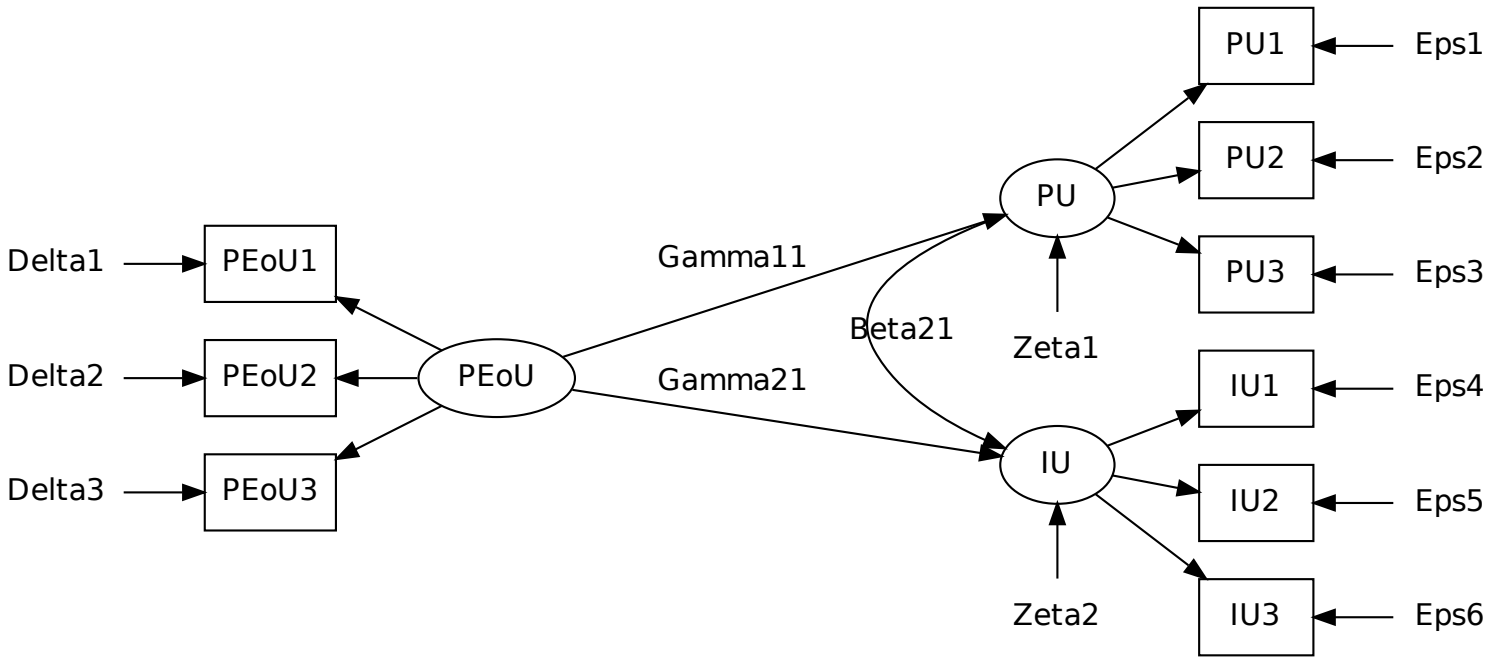
R Packages to Install

- `install.packages('coda')`

Introduction

Structural Equation Modeling

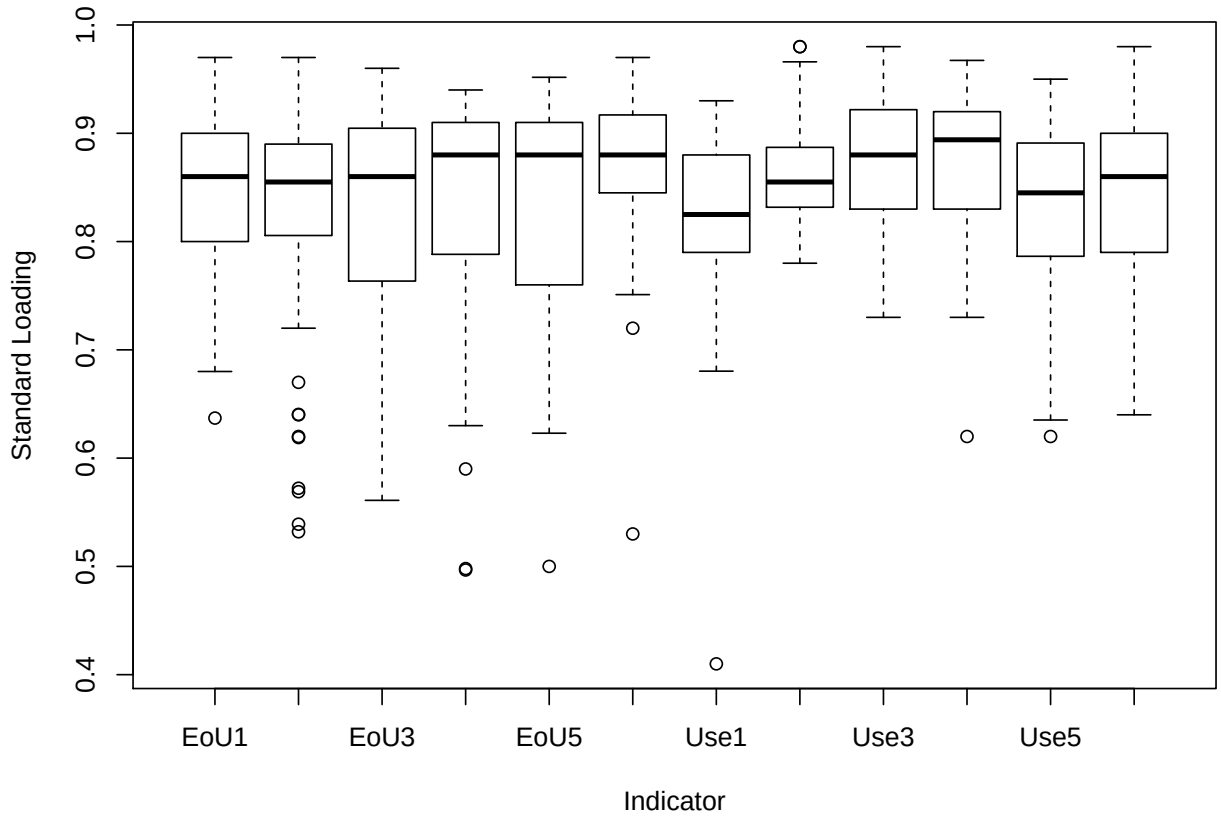
- General framework for regression relationships, typically including
  - Linear regressions
  - Unobserved variables (latent variables, error terms)
- Different ways of estimating, including
  - Covariance-based methods
  - Partial least squares


PU1
Eps1
PU2
Eps2
PU
PU3
Eps3
Delta1
PEoU1
Gamma11
Beta21
Zeta1
Delta2
PEoU2
PEoU
Gamma21
IU1
Eps4
IU
IU2
Eps5
Delta3
PEoU3
Zeta2
IU3
Eps6

Cumulative Theory Building

- Regressions between the same variables are repeatedly estimated
  - In different models
  - With different samples
- Researchers advised to reuse instruments
- Researchers advised to extend theories
- Example: 43 TAM studies in top IS journals between 2004 and 2011

Example – TAM Loadings



Source:

43 empirical TAM papers that cite Davis (1989) or Davis et al. (1989), use their instrument and report sufficient information

Cumulative Theory Building

What to do when your parameter estimate differs?

- Ignore it ?
- ”Fix” your model (typically: omit variable) ?
- Claim that your estimate is the ”right” one ?
- Discuss it ?
- Integrate previous estimate in your estimation!

Bayesian Statistics

- Based on Bayes´ principle of conditional probabilities
- Allows modeling of prior probabilities
- Probability distribution of parameters, rather than NHST
- Flexible method for different models

BUGS and R

BUGS

- Bayesian inference using Gibbs Sampling
- Software using Gibbs sampling of probability distributions
- Different implementations: WinBUGS, OpenBUGS, JAGS
- ”Simple”, explicit syntax to describe statistical model
- Interactive or scripted

R

- General statistical software
- Open-source and cross-platform
- Interactive or scripted

Agenda

- Introduction to Bayesian statistical principles
- Introduction to BUGS and R
- Best practices for estimating Bayesian models
- Exploring different Bayesian Models

Introduction to Bayesian Principles

Conditional Probabilities ("Bayes Principle")

$$p(\theta|x)p(x) = p(x|\theta)p(\theta)$$

$p(x)$	Probability of observing data x (unconditional) (prior likelihood of x)
$p(\theta)$	(Prior) probability of parameters θ
$p(x\|\theta)$	Probability of observing data x, conditional on parameters θ (likelihood of x)
$p(\theta\|x)$	(Posterior) probability of parameters, conditional on observing data x

Updating Beliefs

$$p(\theta|x) \propto p(x|\theta)p(\theta)$$

The posterior probability is the prior probability *updated* after observing the data.

Frequentist Research (NHST)

- Focuses on the likelihood of observing data $p(x|\theta)$
- Reject the hypothesis that $\theta = \theta_0$ if $p(x|\theta_0) < \alpha$
- Data assumed as random from population with $\theta = \theta_0$
- Model parameters assumed as fixed $\theta = \theta_0$
- The likelihood $p(x|\theta_0)$ is useless for making any statements about the observed θ (or any other value for θ)

Bayesian Inference

- Focuses on the probability of effect $p(\theta|x)$
  - *Credibility intervals* around $p(\theta|x)$
- Model parameters also assumed as random from $p(\theta)$
- The probabality $p(\theta|x)$ allows statements about the parameter values θ

Bayesian Advantages

Integration of Prior Knowledge

- Distributional assumptions with mean and variance
- Express point ”belief” and certainty or precision

Integrated treatment of missing values

- No separate imputation necessary
- Missing values treated as any other variable in the model
  - Assumed random with distributional assumption
- Missing information treatment uses model information (MCAR, MAR)
- Allows specification of missingness mechanisms (NMAR)

Bayesian Advantages

Estimation of Latent Variable Scores

- Treatment of latent variables as missing values

Relaxation of Model Identification Requirements

- Allows strictly-non-identified models to be estimated, provided the prior parameter distributions sufficiently restrict possible values
- For example, allows all cross-loadings in a CFA model to be estimated, when constrained (BSEM)

Bayesian Advantages

Small-Sample Accuracy

- CBSEM makes assumptions about asymptotic distribution, valid only for large samples
- PLS has 'consistency-at-large' theorem, estimates unbiased onl for large samples
- Bayesian estimation makes no such assumptions
- More accurate than ML estimation for small samples

Relaxation of Normality Assumptions

- CBSEM and PLS make assumptions about normality of data or parameter estimates
- Bayesian estimation allows other than normal distribution
- Based on prior knowledge or theoretical considerations
- Often restricted to conjugate distributations

Bayesian Advantages

Non-continuous Data

- Similar to non-normal data, Bayesian estimation can deal with discrete data
- Assign appropriate probability distributions for priors

Easy Extensibility to Multi-level models

- Individuals in Teams in Firms in Industries in Countries
- Explicit modeling of within- and between-level relationships
- May lead to better adoption of multi-level models in IS

Bayesian Advantages

Non-Linear Latent Models

- Allows direct modeling of interaction/moderation

Convergence with Traditional Methods

- With increasing sample size, Bayesian estimates converge to ML ones
  - Increasing ”weight of evidence” over prior assumptions
- Using non-informative priors, Bayesian estimates converge to ML ones

Bayesian Drawbacks

Computational Requirements

- Iterative method, frequently tens of thousands of iterations
- Repeated sampling of hundreds of model parameters and latent variable values
- Modern PC can estimate Bayesian SEM in a few minutes

Dependence on Prior Distributions

- Especially for small samples, Bayesian estimates depend on choice of prior distribution
- Even and especially for uninformative priors distribution
- Requires checking results using different priors

Bayesian Drawbacks

Lack of Model Fit Tests

- Lack of easy χ^2 test of model fit
- Instead, the ”posterior predictive p-value” (PPP) is argued to serve similar role

What Bayesian Estimation is Not

- It is not a research methodology
- It is not specific to survey research
- It does not affect concepts of measurement validity and reliability
- It is not meta-analysis, but can be used after a meta-analysis
- It is not a ”silver bullet” that fixes shortcomings of existing methods

A Simple Illustration

Regression Example

$$y_i = \beta x_i + \epsilon_i$$

With the usual assumptions about normally distributed errors

$$\epsilon_i \sim N(0, \sigma^2)$$

Rewriting the above shows that Y are normally distributed:

$$y_i \sim N(\beta x_i, \sigma^2)$$

Hence, the joint likelihood is the following normal density:

$$p(y_i, x_i|\beta, \sigma^2) \propto (\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{(y_i - \beta x_i)^2}{2\sigma^2}\right)$$

Priors for Regression

- Need to specify a distribution for $p(\beta, \sigma^2)$
- Assuming that prior mean and variance are independent, we can simplify to

$$p(\beta, \sigma^2) = p(\beta|\sigma^2)p(\sigma^2)$$

- Choice of
  - Informative prior distribution (low variance, high precision)
  - Uninformative prior distribution (high variance, low precision)
- Typically, *conjugate prior* distributions are used

Conjugate Priors

The product of likelihood $p(x|\theta)$ and prior $p(\theta)$ (i.e. the posterior) is of the same form as the prior

Likelihood function	*Conjugate prior*	*Mean*	*Variance*	*Example uninformative*
Normal (known var.)	Normal $N(\mu, \eta)$	μ	η	$N(0, 10^{10})$
Normal (known mean)	Inverse Gamma $IG(a, b)$	$\frac{b}{a-1}$	$\frac{b^2}{(a-1)^2(a-2)}$	$IG(0, 0)$ $IG(-1, 0)$
Normal (known mean)	Gamma $G(a, b)$	$\frac{a}{b}$	$\frac{a}{b^2}$	$G(.001, .001)$

Conjugate Priors

Likelihood function	*Conjugate prior*	*Mean*	*Variance*	*Example uninformative*
MV Normal (means, cov)	Inverse Wishart $IW(\Omega_p, d)$	$\frac{\Omega}{d-p-1}$	$\frac{1}{(d-p)(d-p-1)^2(d-p-3)}$	$IW(0, -p-1)$ $IW(I, p+1)$ $IW(0, 0)$
MV Normal (means, prec)	Wishart $W(\Omega_p, d)$	$d\Omega$		
Exponential, Gamma	Gamma $G(a, b)$	$\frac{a}{b}$	$\frac{a}{b^2}$	$G(.001, .001)$
	Uniform $U(a, b)$	$\frac{1}{2}(a+b)$	$\frac{1}{12}(b-a)^2$	$U(-10^{10}10^{10})$ $U(0, 10^{10})$

Priors and Hyper-Parameters

$$p(\beta|\sigma^2) \sim N(\mu, \eta)$$
$$p(\sigma^2) \sim InvGamma(a, b)$$

- Hyper-parameters μ and η express our point belief and certainty about the regression coefficient β, e.g.

$$\mu = 0.5, \eta = 5$$

- Hyper-parameters a and b express our point belief and certainty about the error/residual variance, e.g.

$$a = 3, b = 1 \rightarrow \text{mean of 0.5 and variance of 0.25}$$

MCMC with Gibbs Sampling

MCMC

- Markov Chain Monte Carlo
- Current sample depends only on prior sample, not on entire history

Gibbs Sampler

- Iteratively sample from posterior distribution. In our example:
  1. Sample from $p(\beta|\sigma^2, Y, X, a, b, \hat{\beta}, S)$
  2. Sample from $p(\sigma^2|\beta, Y, X, a, b, \hat{\beta}, S)$
- Continue iteration until stable sample of sufficient size
- Multiple sampling chains from different starting values
- Retain only samples from stable chains, discard ”burn-in” samples

Introduction to BUGS and R

BUGS

Main Files for BUGS

- Model
- Data
- Initial values (may be omitted)
- Script

BUGS Syntax

Deterministic Nodes

- Defined using `<-`, e.g. `a <- b + c`
- Computed

Stochastic Nodes

- Assigned a distribution using `~`, e.g. `a ~ dnorm(0,1)`
- Sampled / Estimated

Iterators and Arrays

- Arrays are declared using `[]` brackets, e.g. `a[1]`
- Iterate using `for`, e.g. `for(i in 1:100)`

A First Regression Example

Model Definition

```
model {
 for(i in 1:N) {
  mu[i] <- beta * x[i]
  y[i] ~ dnorm(mu[i],psi)
 }
 beta ~ dnorm(0.5, 5)
 psi ~ dgamma(3, 1)
 errvar <- 1/psi
}
```

Data File (OpenBUGS)

```
list(
 x=c(1.3168, 0.9062, 0.8608, 1.2683, ... ),
 y=c(0.6145, 1.5050, 0.6581, 0.9294, ... ),
 N=100
)
```

Initial Value File(s) (OpenBUGS)

```
list(beta=0.75, psi=0.1)
```

Script File (OpenBUGS)

```
modelSetWD('f:\BayesianTutorial\example1\')
modelCheck('example1model.txt')
modelData('example1data.txt')
modelCompile(3)
modelInits('example1inits1.txt', 1)
modelInits('example1inits2.txt', 2)
modelInits('example1inits3.txt', 3)
modelInitialize
samplesSet('beta')
samplesSet('psi')
samplesSet('errvar')
dicSet()
modelUpdate(5000, 1, 1, 'F')
samplesCoda('*', 'codaoutput')
samplesStats('*')
dicStats()
```

Data File (JAGS)

```
x <- c(1.3168, 0.9062, 0.8608, 1.2683, ... )
y <- c(0.6145, 1.5050, 0.6581, 0.9294, ... )
N <- 100
```

Initial Value File(s) (JAGS)

```
beta <- 0.75
psi <- 0.1
```

Script File (JAGS)

```
cd '.'
load dic
model in 'example1model.txt'
data in 'example1data.jags.txt'
compile, nchains(3)
parameters in 'example1inits1.jags.txt', chain(1)
parameters in 'example1inits2.jags.txt', chain(2)
parameters in 'example1inits3.jags.txt', chain(3)
initialize
monitor beta
monitor psi
monitor errvar
monitor pD, type(mean)
monitor deviance, type(mean)
update 5000
coda *, stem('jagsoutputCODA')
coda pD, stem('PD')
coda deviance, stem('DEV')
exit
```

Running the Example

OpenBUGS

- On the command line:
  `OpenBUGS example1script.openbugs.txt`
- In OpenBUGS: Model → Script

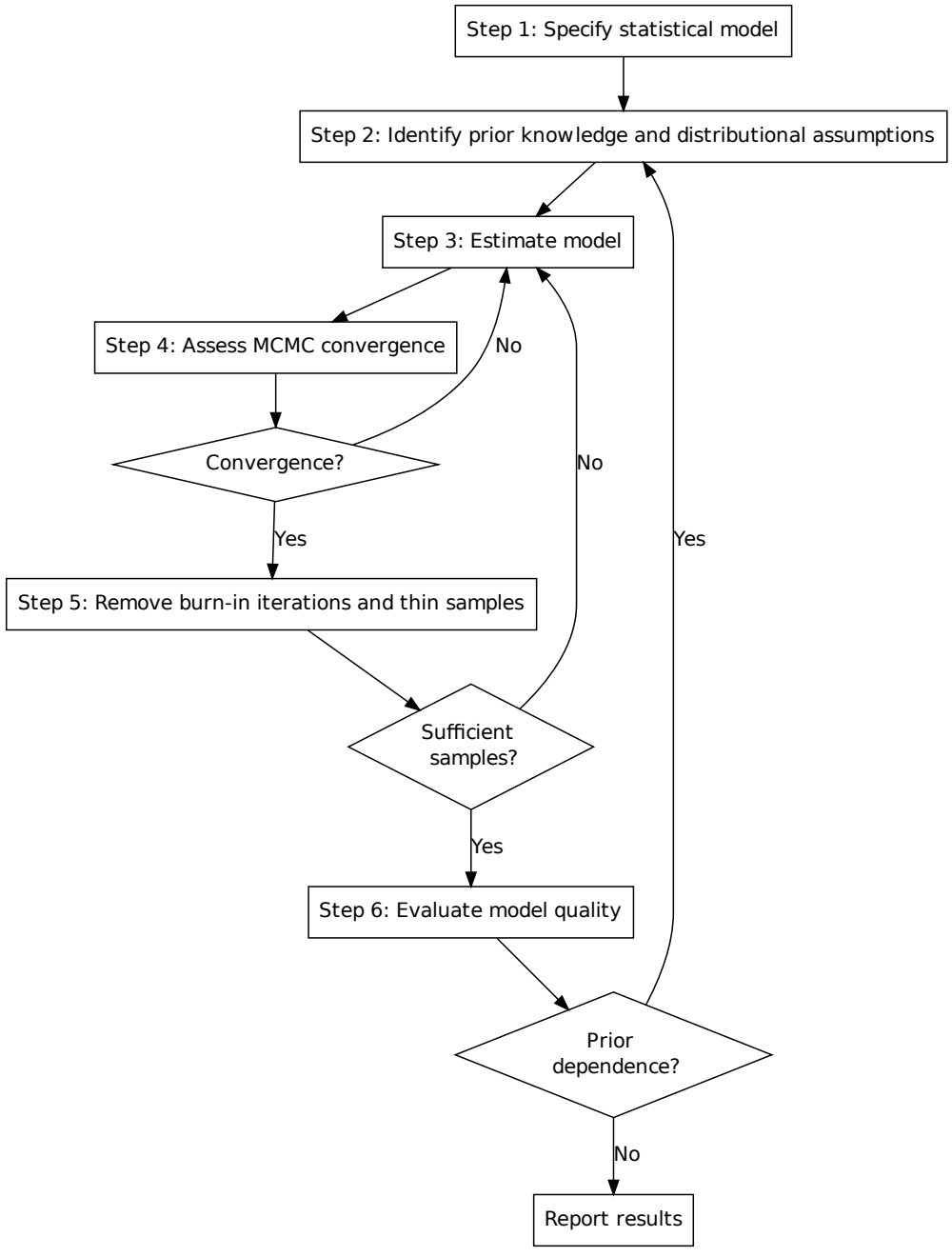
JAGS

- On the command line:
  `jags example1script.jags.txt`

Best Practices Example for Estimating Bayesian Models

Recommended Process

Step 1	Specify the statistical model
Step 2	Identify prior knowledge and distributional assumptions
Step 3	Estimate the model
Step 4	Assess MCMC convergence
Step 5	Remove burn-in iterations and thin samples
Step 6	Evaluate model quality



Step 1: The Statistical Model

Example: 2-Latent Factor Model

- Straightforward extension of regression model
- (Except for the issue with the pesky Wishart priors)

```
model {
  for(i in 1:N) {
  #measurement model
    for(j in 1:P) {
      y[i,j] ~ dnorm(mu[i,j],errorprec[j])
    }
    mu[i,1]<-xi[i,1]
    mu[i,2]<-lam[1]*xi[i,1]
    mu[i,3]<-lam[2]*xi[i,1]
    mu[i,4]<-xi[i,2]
    mu[i,5]<-lam[3]*xi[i,2]
    mu[i,6]<-lam[4]*xi[i,2]
  #structural model
    xi[i,1:2] ~ dmnorm(u[1:2],latprec[1:2,1:2])
  } #end of i
```

Step 2: Identify Prior Knowledge and Assumptions

- Research the literature for previous estimates and assumptions
  - Use existing meta-analyses
  - Ensure estimates are comparable
    - Same/similar samples
    - Same/similar items
    - Same/similar models
- Use the appropriate conjugate prior distribution to the type of assumed likelihood
- Use informative prior distributions when sufficient knowledge exists
- Use uninformative prior distributions when no previous knowledge exists
  - Uninformative priors should be skeptical
  - Reflect a Null-Hypothesis of ”no effect”
  - For example, means of zero for regression parameters

Model Definition Continued

```
#priors on loadings
  lam[1]~dnorm(0,0.0001)
  lam[2]~dnorm(0,0.0001)
  lam[3]~dnorm(0,0.0001)
  lam[4]~dnorm(0,0.0001)
#priors on errors
  for(j in 1:P) {
    errorprec[j] ~ dgamma(1, 1)
    errorvar[j]<-1/errorprec[j]
  }
```

Model Definition Continued

```
#priors on latent (co-)variances
  latprec[1:2,1:2] ~ dwish(V[,], 4)
  latcov[1:2,1:2] <- inverse(latprec[,])
#compute latent correlation to monitor it
  latcor <- latcov[1,2] /
                (sqrt(latcov[1,1])*sqrt(latcov[2,2]))
#construct the V matrix for the Wishart distrib
  V[1,1] <- 1
  V[1,2] <- 0
  V[2,1] <- 0
  V[2,2] <- 1
#construct the u matrix for the latent means
  u[1] <- 0
  u[2] <- 0
} #end of model
```

Data – JAGS

```
y <-
structure(c(0.954296981154502, -1.4780727262838 6,
-2.18322025551015, 2.02660110947426, 0.703106236064238,
-1.63635998174368, 1.95179251524944, 1.18497846785358,
2.03344373450836, 0.490172251893674, -0.442571445072999,
...
), .Dim = c(1000, 6) )
N <- 1000
P <- 6
```

Created from R using

```
dump(c('y', 'N', 'P'), 'example2data.jags')
```

Missing values can be specified using `NA`

Data – OpenBUGS

```
list(
N=1000,
P=6,
y=structure(.Data=c(0.954296981154502, 0.112291234691847, 1.32
1.32584263559602, 1.02601776794446, 0.218156814188609, -1.4780
0.594570552962093, -1.69912974905603, 0.928509780830628, 2.281
0.432304507895 28, -0.683921633913079, -3.55558968133591, 0.314
-1.92142152759482, -1.41469999245265, -2.10978116861893, -2.18
),
.Dim=c(1000, 6))
)
```

Created from R using `dput(list(y=aperm(y), ...))` and adjusting dimensions

Missing values can be specified using `NA`

Step 3: Estimate the Model (JAGS)

```
cd '.'
load dic
model in 'example2model.txt'
data in 'example2data.jags.txt'
compile, nchains(3)
initialize
monitor lam
monitor errorvar
monitor latcor
monitor pD, type(mean)
monitor deviance, type(mean)
update 5000
coda *, stem('jagsoutputCODA')
coda pD, stem('PD')
coda deviance, stem('DEV')
exit
```

Notice no initial values specified!

Step 3: Estimate the Model (OpenBUGS)

```
modelSetWD('.')
modelCheck('example2model.txt')
modelData('example2data.openbugs.txt')
modelCompile(3)
modelInits('example2inits1.txt', 1)
modelInits('example2inits2.txt', 2)
modelInits('example2inits3.txt', 3)
modelGenInits()
modelInitialize
samplesSet('lam')
samplesSet('errorvar')
samplesSet('latcor')
dicSet()
modelUpdate(5000, 1, 1, 'F')
samplesCoda('*', 'openbugsoutput')
samplesStats('*')
dicStats()
```

Initial values specified!

Sampling Values

Latent Variable Scores

- Use `monitor xi` (JAGS) or `samplesSet('xi')` (OpenBUGS) in script
- Use `monitor xi[243,2]` (JAGS) or `samplesSet('xi[243,2]')` (OpenBUGS) in script

Missing Values

- Use e.g. `monitor y[198,3]` (JAGS) or `samplesSet('y[198,3]'` (OpenBUGS) in script

Step 3: Recommendations

- Use at least 3 MCMC chains
- Use at least 5000 sampling iterations
- Specify initial values, rather than generating them

Step 4: Assess MCMC Convergence

- Using the `coda` package in R

Within Chain Convergence

- Trace and density plots
- Geweke's test
- Heidelberger and Welch's test

Between Chain Convergence

- Trace and density plots
- Gelman's PSR (potential scale reduction) factor

Reading BUGS Output into R

OpenBUGS & JAGS

```
library(coda)
mcmc.list <- read.openbugs('jagsoutput')
```

Requires JAGS output be named appropriately, e.g.

```
coda *, stem('jagsoutputCODA')
```

Basic Results Information

```
summary(mcmc.list)

Iterations = 1:5000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 5000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                Mean      SD  Naive SE Time-series SE
lam[1]        0.9523 0.06417 0.0005239      0.0022556
lam[2]        0.8607 0.05713 0.0004665      0.0018833
...
errorvar[1] 0.9069 0.07071 0.0005773      0.0021549
errorvar[2] 0.9669 0.06819 0.0005568      0.0013738
...
```

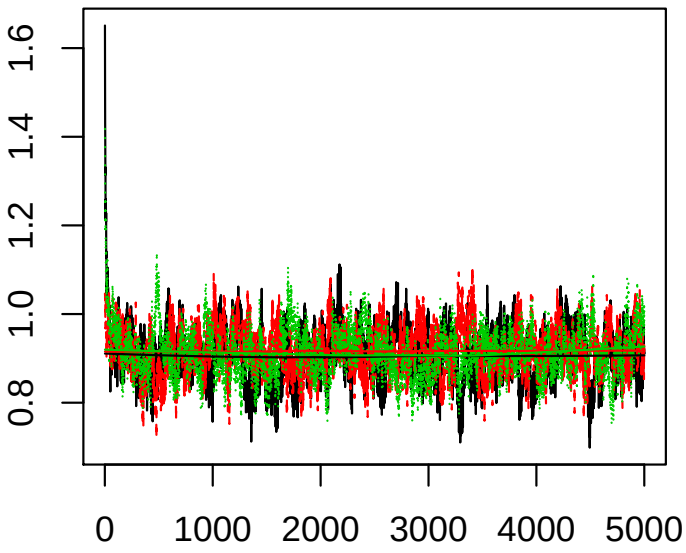
Can we trust these results already?

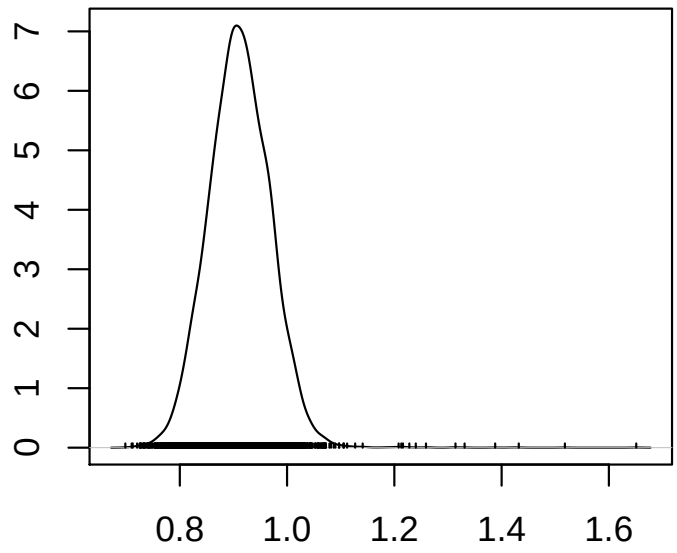
Eyeballing with Trace and Density Plots

```
plot(mcmc.list, ask=TRUE)
```

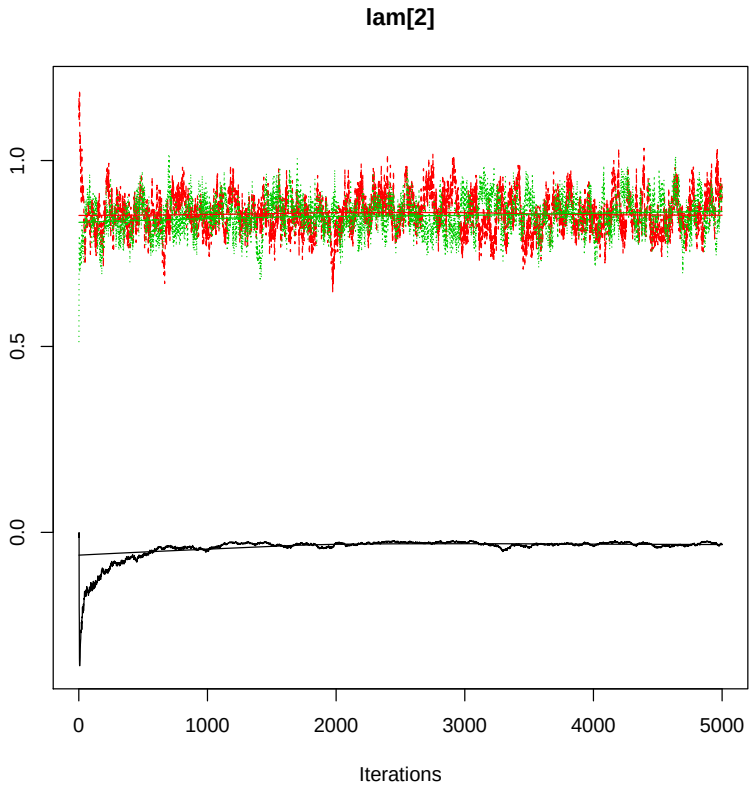
- Density plots should reflect the expected posterior distribution based on the choice of likelihood and prior
- Sampling means of each chain becomes stable
- Sampling means of all chains converge

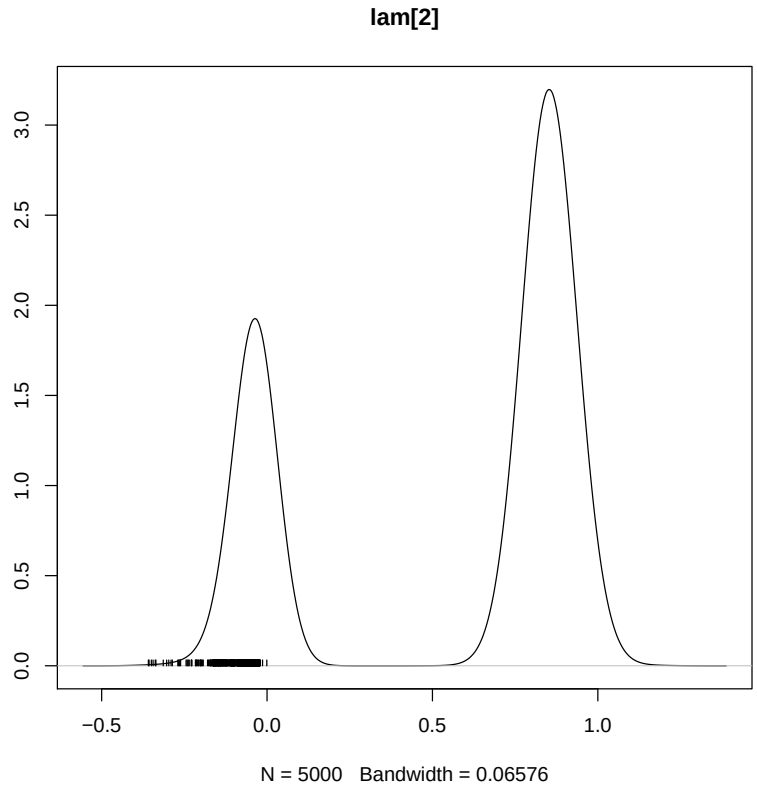
Example Trace and Density Plot



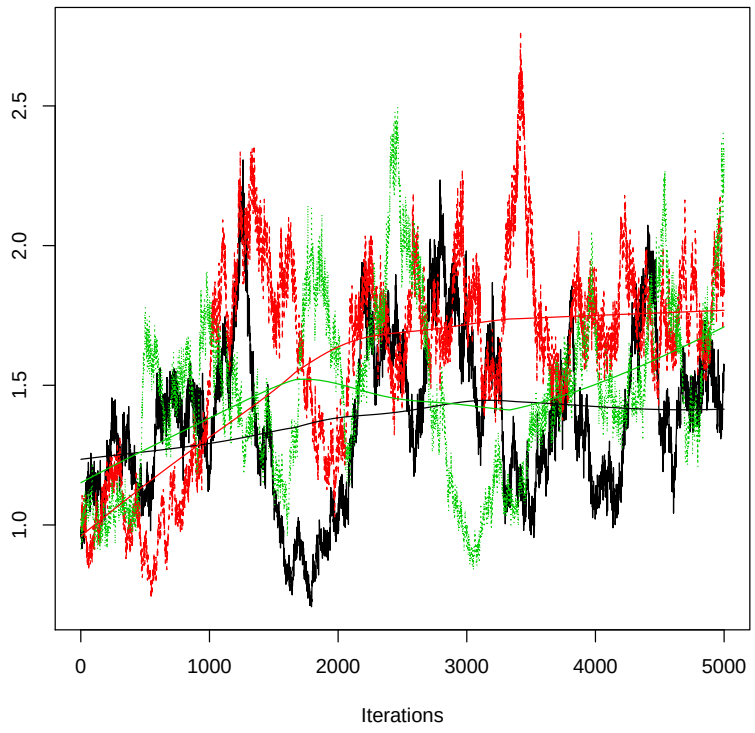


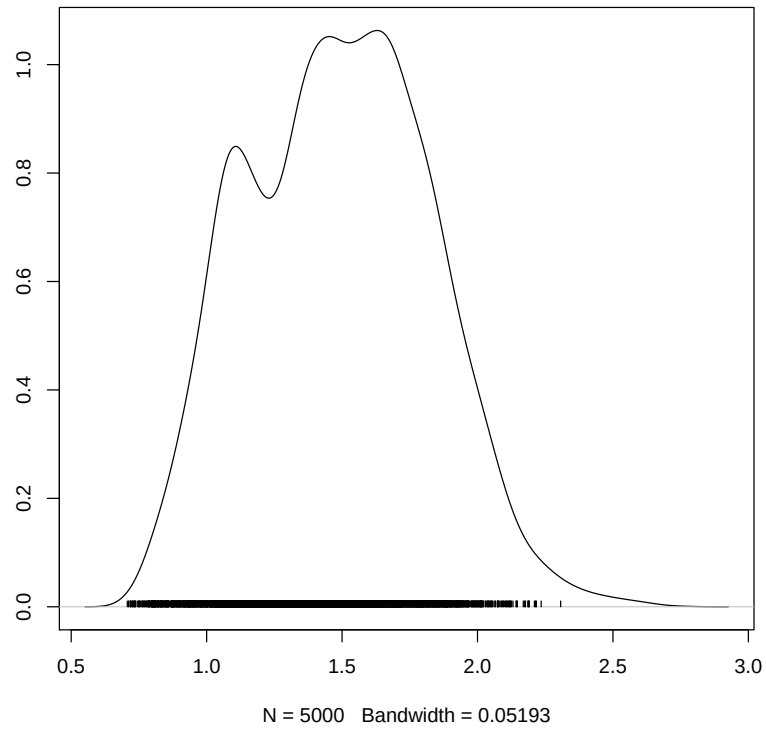
Example Trace and Density Plot





Example Trace and Density Plot





Geweke's Diagnostic

Principle

- Test equality of means of different parts of the trace (z-test)
- Repeat until stationary "tail" of trace is identified

R

- `geweke.diag(mcmc.list, 0.1, 0.5)`

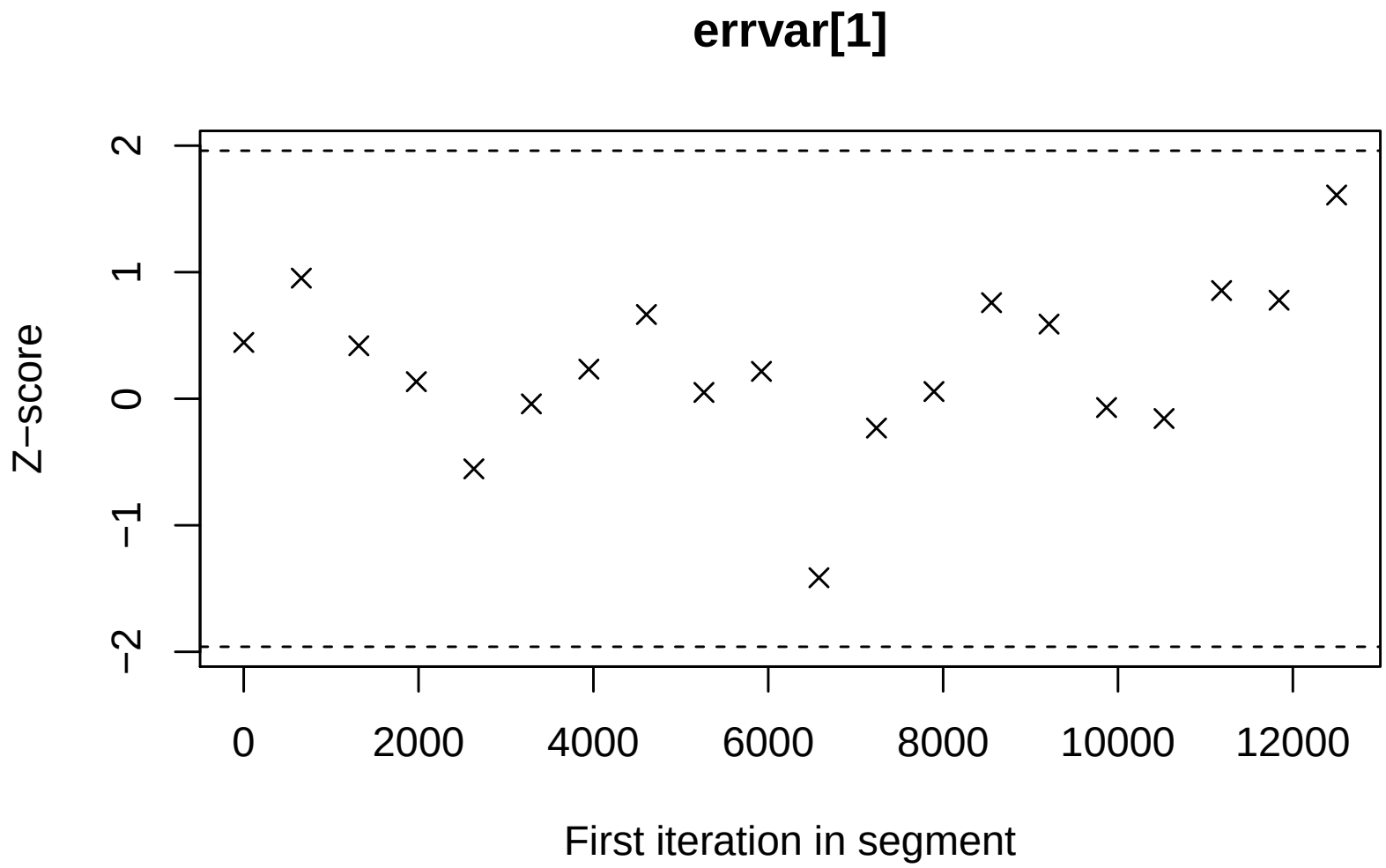
Output

```
[[1]]

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

     lam[1]      lam[2]      lam[3]      lam[4] errorvar[1] errorvar[2
    1.53419     1.72431     1.26276     1.31907     1.48009    -0.3742
errorvar[3] errorvar[4] errorvar[5] errorvar[6]      latcor
   -0.08549     1.45678     0.04935     0.08728     1.86148
... (Repeated for 2nd and 3rd chain) ...
```

Plotting Geweke's Diagnostic

```
geweke.plot(mcmc.list)
```

Output



Heidelberger & Welch's Diagnostic

Principle

- Apply Cramer-von-Mises test to assess whether sample data come from stationary distribution
- Apply successively first to entire chain, then discard then first 10%, 20%, etc. until stationarity test passed (or failed)

R

- `heidel.diag(mcmc.list)`

Output

```
[[1]]

            Stationarity start     p-value
            test         iteration
lam[1]      passed       1         0.642
lam[2]      passed       1         0.343
...
errorvar[1] passed       1         0.376
errorvar[2] passed       1         0.841
...
latcor      passed       1         0.152

... (Repeated for other chains) ...
```

Interchain Convergence Evaluation using PSRF

Between chain variance:

$$B = \frac{n}{m-1} \sum_{j=1}^{m} \left(\bar{\psi}_{.j} - \bar{\psi}_{..}\right)^2$$

Within chain variance:

$$W = \frac{1}{m} \sum_{j=1}^{m} \left[\frac{1}{n-1} \sum_{i=1}^{n} \left(\bar{\psi}_{ij} - \bar{\psi}_{.j}\right)^2 \right]$$

PSRF:

$$PSRF = \sqrt{\frac{\frac{n-1}{n} W + \frac{1}{n} B}{W}}$$

R:

```
gelman.diag(mcmc.list)
```

Output:

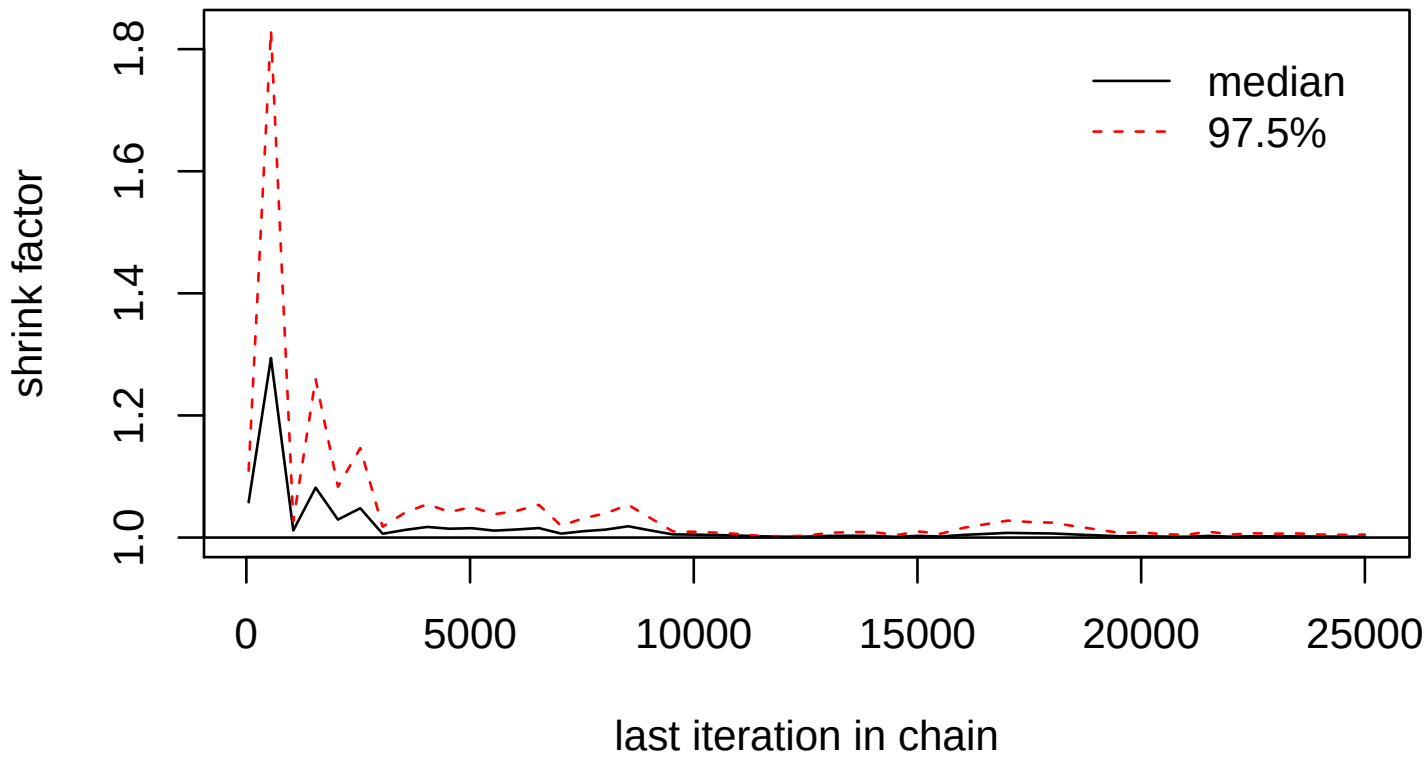
```
Potential scale reduction factors:

            Point est. Upper C.I.
lam[1]            1.00       1.01
lam[2]            1.00       1.02
lam[3]            1.01       1.03
lam[4]            1.01       1.02
errorvar[1]       1.01       1.02
errorvar[2]       1.00       1.00
errorvar[3]       1.00       1.01
errorvar[4]       1.00       1.00
errorvar[5]       1.00       1.01
errorvar[6]       1.00       1.01
latcor            1.00       1.01

Multivariate psrf

1.01
```

Plotting PSRF

R:

```
gelman.plot(mcmc.list)
```

Output:



Step 4: Recommendations

- Geweke's z-statistic should be less than 1.96 for all parameters
- Heidelberger and Welch's stationarity test should be passed by all parameters (note the starting iteration)
- Heidelberger and Welch's half-width-test should be passed by all parameters
- Gelman's PSRF should be less than 1.1. for all parameters

Step 4: Possible Remedies

- Increase the number of MCMC iterations
- Set different starting values
- Then repeat step 4 (assess MCMC convergence)

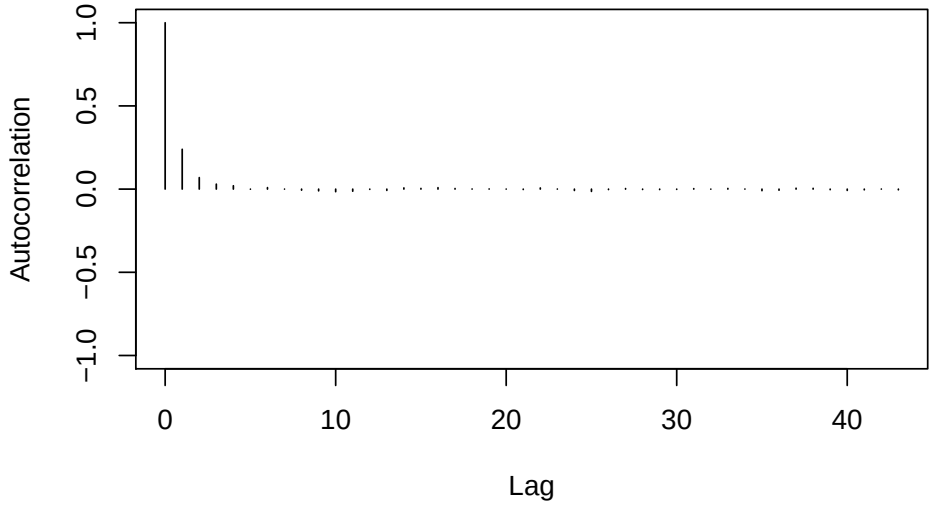
Step 5: Remove Burn-In Iterations and Thin Samples

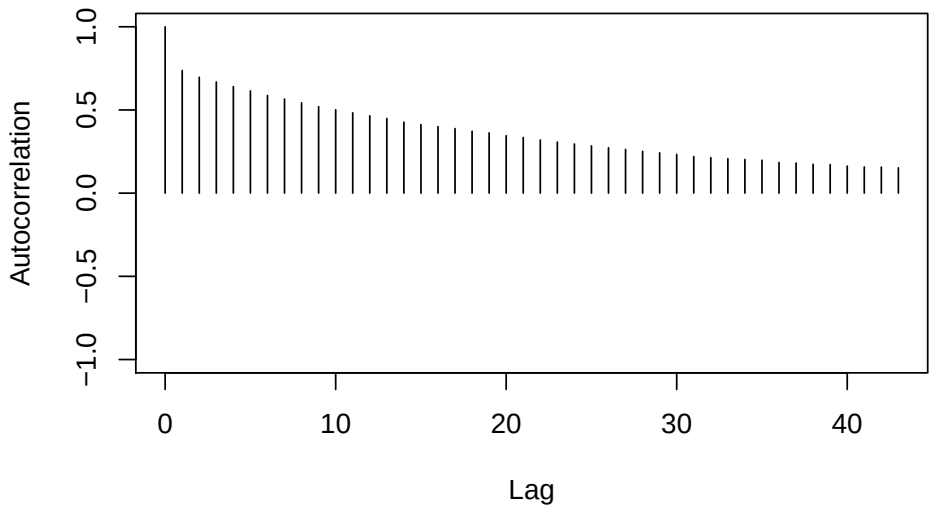
- Beginnings of chains are unstable and should be discarded
  $\Rightarrow$ Discard the "burn-in" iterations at beginning of chain
  - Geweke's diagnostic
  - Heidelberger & Welch's diagnostic
  - Gelman's PSRF
- Inference assumes independence of observations, but chains are autocorrelated
  $\Rightarrow$ Remove sufficient samples so that remainder is not autocorrelated
  - Compute and plot autocorrelations
  - Raftery-Lewis diagnostic

```
autocorr.diag(mcmc.list)
```

```
          lam[1] lam[2]  lam[3] lam[4] errorvar[1]
Lag 0      1.00  1.000  1.0000  1.000        1.000
Lag 1      0.76  0.707  0.7514  0.731        0.662
Lag 5      0.47  0.403  0.4470  0.416        0.333
Lag 10     0.30  0.250  0.2609  0.231        0.220
Lag 50     0.01  0.012 -0.0091  0.023        0.019
...
```

autocorr.plot(mcmc.list)





raftery.diag(mcmc.list, q=0.5, r=0.05)

```
Quantile (q) = 0.025
Accuracy (r) = +/- 0.005
Probability (s) = 0.95
```

	Burn-in (M)	Total (N)	Lower bound (Nmin)	Dependence factor (I)
lam[1]	10	12522	3746	3.34
lam[2]	10	13936	3746	3.72
lam[3]	15	16734	3746	4.47
lam[4]	10	11436	3746	3.05
errorvar[1]	9	11949	3746	3.19
errorvar[2]	4	4955	3746	1.32
errorvar[3]	8	10242	3746	2.73
errorvar[4]	8	11354	3746	3.03
errorvar[5]	10	11862	3746	3.17
errorvar[6]	7	7613	3746	2.03
latcor	5	5716	3746	1.53

Thinning in R

```
thinned <- window(mcmc.list, 500, 5000, 5)
```

Step 5: Recommendations

First

- Discard (generously) the burn-in iterations from the sample,
- Assess autocorrelation within MCMC series,
- Use Raftery and Lewis' method to identify the required number of samples for desired accuracy,

Then

- Thin the series to avoid autocorrelation, and/or
- Increase the sample size (MCMC iterations) and rerun model estimation

Step 6: Evaluate Model Quality

DIC

- Deviance Information Criterion
- Not an absolute criterion
- Useful for model comparisons
- Should be used with a measure of model complexity

PPP

- Posterior Predictive Probability
- Probability that the original data fits the model better than the predicted data
- Requires a fit function, typically the traditional χ^2 fit function

DIC using OpenBUGS

Scriptfile

```
...
dicSet()
...
dicStats()
```

Output

```
Deviance information
          Dbar      Dhat      DIC       pD
y         16460.0   1.5E+4    17910.0   1457.0
total     16460.0   1.5E+4    17910.0   1457.0
```

DIC using JAGS

Scriptfile

```
...
monitor pD, type(mean)
monitor deviance, type(mean)
...
coda pD, stem('PD')
coda deviance, stem('DEV')
```

R

```
dev.data <- read.csv('DEVtable0.txt', row.names=1, sep=' ')
pd.data <- read.csv('PDtable0.txt', row.names=1, sep=' ')
Dbar = sum(dev.data)
pD <- sum(pd.data)
DIC <- Dbar + pD
```

PPP

$$PPP = p(f(Y, \theta) < f(\tilde{Y}, \theta))$$

- Low PPP values indicate that original data fits the model worse than data predicted from the model
  - PPP of 0.5 is excellent fit
  - PPP of 0.05 is acceptable fit [Muthen and Asparouhov, 2012]
- Less bias than classic χ^2 for small samples
- Less powerful to reject misspecified modles than classic χ^2 (but this effect diminishes with increasing sample size)

PPP using R

Code for CFA models in script file ”BayesianCFAPPP.R”

Step 6: Recommendations

- Use the PPP to assess model fit, aim for PPP of > 0.05
- Use the DIC to compare alternative/competing models
- Especially for small samples, re-estimate model with different uninformative priors

Step 7: Reporting Results

R

```
summary(thinned)
```

Recommendations

- Report and justify the choice of informative prior distributions
- Report all diagnostics and the estimation decisions based on them
- Report the estimated mean or mode of the posterior distribution and credibility intervals for important parameters

Exploring different SE Models

Example 3: Fixing exogenous latent variances

```
...
    mu[i,1]<-lam[1]*xi[i,1]
    mu[i,2]<-lam[2]*xi[i,1]
    mu[i,3]<-lam[3]*xi[i,1]
    mu[i,4]<-lam[4]*xi[i,2]
    mu[i,5]<-lam[5]*xi[i,2]
    mu[i,6]<-lam[6]*xi[i,2]
    xi[i,1:2]~dmnorm(u[1:2],V[1:2,1:2])
...
  lam[1]~dnorm(0,0.0001)
  lam[2]~dnorm(0,0.0001)
  lam[3]~dnorm(0,0.0001)
  lam[4]~dnorm(0,0.0001)
  lam[5]~dnorm(0,0.0001)
  lam[6]~dnorm(0,0.0001)
...
```

Notice all loadings are free and the variance/covariance of the `xi` is not sampled from a Wishart prior, but set to matrix `V`.

Example 4: Freeing all coefficients

```
...
    xi[i,1:2]~dmnorm(u[1:2],latprec[1:2,1:2])
...
  lam[1]~dnorm(0.9,10)
  lam[2]~dnorm(0.9,10)
  lam[3]~dnorm(0.9,10)
  lam[4]~dnorm(0.9,10)
  lam[5]~dnorm(0.9,10)
  lam[6]~dnorm(0.9,10)
...
  latprec[1:2,1:2] ~ dwish(V[,], 10)
  V[1,1] <- 7
  V[1,2] <- 3.5
  V[2,1] <- 3.5
  V[2,2] <- 7
...
```

- Note the large precisions on the `lam[]` priors to make this converge reasonably well.
- Compare the convergence and results of the models with and without those priors

Example 5: Free cross-loadings

```
...
    mu[i,1]<-lam[1]*xi[i,1] + lam[7]*xi[i,2]
    mu[i,2]<-lam[2]*xi[i,1] + lam[8]*xi[i,2]
    mu[i,3]<-lam[3]*xi[i,1] + lam[9]*xi[i,2]
    mu[i,4]<-lam[4]*xi[i,2] + lam[10]*xi[i,1]
    mu[i,5]<-lam[5]*xi[i,2] + lam[11]*xi[i,1]
    mu[i,6]<-lam[6]*xi[i,2] + lam[12]*xi[i,1]
...
  lam[7]~dnorm(0.0,10)
  lam[8]~dnorm(0.0,10)
  lam[9]~dnorm(0.0,10)
  lam[10]~dnorm(0.0,10)
  lam[11]~dnorm(0.0,10)
  lam[12]~dnorm(0.0,10)
...
```

- Proposed by Muthen and Asparouhov (PsychMeth 2012) to improve fit.
- But see Stromeyer et al. (JOM 2014) for critique.

Example 6: Latent Regressions (Structural Model) and Intercepts

```
model {
  for(i in 1:N) {
    mu[i,1]<-a[1] + xi[i,1]
    mu[i,2]<-a[2] + lam[1]*xi[i,1]
    mu[i,3]<-a[3] + lam[2]*xi[i,1]
    mu[i,4]<-a[4] + xi[i,2]
    mu[i,5]<-a[5] + lam[3]*xi[i,2]
    mu[i,6]<-a[6] + lam[4]*xi[i,2]
    mu[i,7]<-a[7] + eta[i]
    mu[i,8]<-a[8] + lam[5]*eta[i]
    mu[i,9]<-a[9] + lam[6]*eta[i]
    #structural model
    xi[i,1:2]~dmnorm(u[1:2],latprec[1:2,1:2])
    nu[i] <- gamma[1]*xi[i,1] + gamma[2]*xi[i,2]
    eta[i] ~ dnorm(nu[i], preceta)
  }
  ...
```

```
  ...
  for(j in 1:P) {
    # uninformative priors on intercepts
    a[j]~dnorm(0,0.0001)
  }
  ...
  #priors on latent errors
  preceta ~ dgamma(1, 1)
  vareta <- 1/preceta
  #priors on structural coefficients
  gamma[1] ~ dnorm(0, 0.0001)
  gamma[2] ~ dnorm(0, 0.0001)
}
```

Example 7: Ordered Categorical Variables

```
model {
  for(i in 1:N) {
    for(j in 1:P) {
       y[i,j]~dnorm(mu[i,j],errorprec[j])
       z[i,j]~dinterval(y[i,j], cutoffs)   }
    mu[i,1]<-xi[i]
    mu[i,2]<-lam[1]*xi[i]
    mu[i,3]<-lam[2]*xi[i]
    mu[i,4]<-eta[i]
    mu[i,5]<-lam[3]*eta[i]
    mu[i,6]<-lam[4]*eta[i]

    xi[i]  ~ dnorm(0,phi)
    eta[i] ~ dnorm(nu[i],laterrprec)
    nu[i] <- gam * xi[i]
  }
```

OpenBUGS

```
y[i,j]~dnorm(mu[i,j],errorprec[j])
          I(cutoffs[z[i,j]],cutoffs[z[i,j]+1])
```

Example 7: Data

- The `z` are actually observed
- The `z` are interval distributed over an underlying continuous variable `y`
- Requires cutoffs defined on the underlying continuous variable
- Requires specification of initial values on the underlying continuous variable

```
cutoffs <- c(-200, -2.5, -1, 1, 2.5, 200)
```

An easy extension:

- Different cut-off points for different items

Example 8: Nonlinear Model with Covariates

```
model {
  for(i in 1:N) {
    #measurement model
     ...
    #structural model
    xi[i,1:2]~dmnorm(u[1:2],latprec[1:2,1:2])
    nu[i] <- gamma[1]*xi[i,1] + gamma[2]*xi[i,2] +
             gamma[3]*xi[i,1]*xi[i,2] + gamma[4]*z[i]
    eta[i] ~ dnorm(nu[i], preceta)
  }
  ...
}
```

- Allows modelling of latent variable interactions (moderation)
- Allows ”formative” models

Example 9: Multi-Sample Model with Ordered Categorical Variables and Cross-Group Constraints

```
model {
  for(i in 1:N1) {
    #measurement equation model
    for(j in 1:P) {
      y1[i,j]~dnorm(mu1[i,j],errorprec1[j])
      z1[i,j]~dinterval(y1[i,j], cutoffs)
    }
    mu1[i,1]<-xi1[i]
    mu1[i,2]<-lam1[1]*xi1[i]
    mu1[i,3]<-lam1[2]*xi1[i]
    mu1[i,4]<-eta1[i]
    mu1[i,5]<-lam1[3]*eta1[i]
    mu1[i,6]<-lam1[4]*eta1[i]

    xi1[i]  ~ dnorm(0,phi)
    eta1[i] ~ dnorm(nu1[i],laterrprec1)
    nu1[i] <- gam1 * xi1[i]

  }
```

```
for(i in 1:N2) {
  #measurement equation model
  for(j in 1:P) {
    y2[i,j]~dnorm(mu2[i,j],errorprec2[j])
    z2[i,j]~dinterval(y2[i,j], cutoffs)
  }
  # Equality constraints on lambdas
  mu2[i,1]<-xi2[i]
  mu2[i,2]<-lam1[1]*xi2[i]
  mu2[i,3]<-lam1[2]*xi2[i]
  mu2[i,4]<-eta2[i]
  mu2[i,5]<-lam1[3]*eta2[i]
  mu2[i,6]<-lam1[4]*eta2[i]
  #structural equation model
  xi2[i]  ~ dnorm(0,phi)
  eta2[i] ~ dnorm(nu2[i],laterrprec2)
  nu2[i] <- gam2 * xi2[i]

}
```

```
  for(j in 1:4) {
    lam1[j] ~ dnorm(1,1)
  }
  gam1 ~ dnorm(0.8,1)
  gam2 ~ dnorm(0.8,1)

  for(j in 1:P) {
    errorprec1[j]~dgamma(3,1)
    errorvar1[j]<-1/errorprec1[j]
    errorprec2[j]~dgamma(3,1)
    errorvar2[j]<-1/errorprec2[j]
  }

  laterrprec1 ~ dgamma(3,1)
  laterrvar1 <- 1/laterrprec1
  laterrprec2 ~ dgamma(3,1)
  laterrvar2 <- 1/laterrprec2

  phi ~ dgamma(1,1)
}
```

Example 9: Easy Extensions

- Equality constraints on error variance
- Equality constraints on latent (co-)variances
- Equality constraints on structural coefficients
- Equality constraints *within* a sample

- As before, truncated or censored variables require initial values for the unobserved continuous underlying variable

Example 10: Missing Data Mechanisms

```
model {
  for(i in 1:N) {
    ...
    #missingness mechanism model
    for (j in 1:P) {
      R[i,j] ~ dbern(pi[i,j])
      logit(pi[i,j])
         <- b[1]+b[2]*y[i,1]+b[3]*y[i,2]+b[4]*y[i,3]
    }
    ...
  }
  for(j in 1:4) {
    b[j] ~ dnorm(0, 1)
  }
  ...
}
```

- Set monitors for `y` values and write to CODA output
- Use coda statistics in R to see Bayesian estimates for missing values

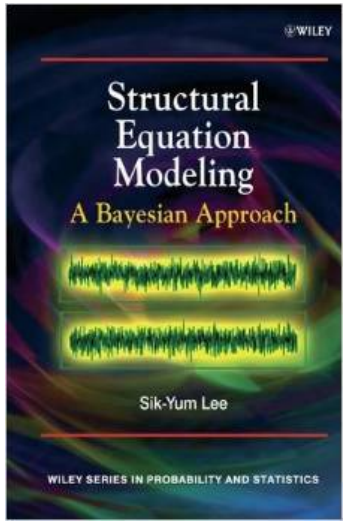
Example 11: Binomial Distributions

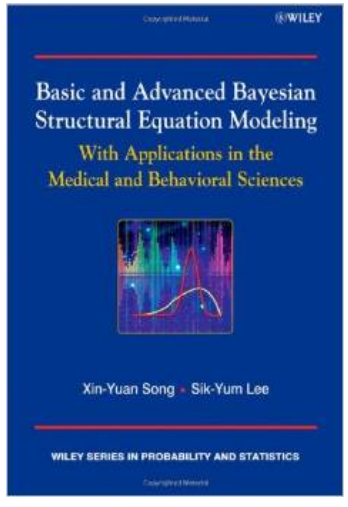
```
model {
  for(i in 1:N1) {
    #measurement equation model
    for(j in 1:P) {
      z1[i,j]~dbin(pb1[i,j], 5)
      logit(pb1[i,j]) <- mu1[i,j]
    }
    ...
}
```

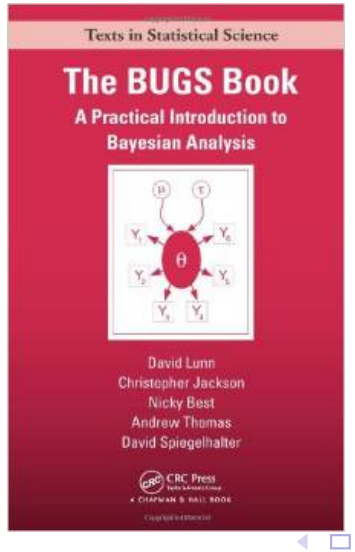
Resources

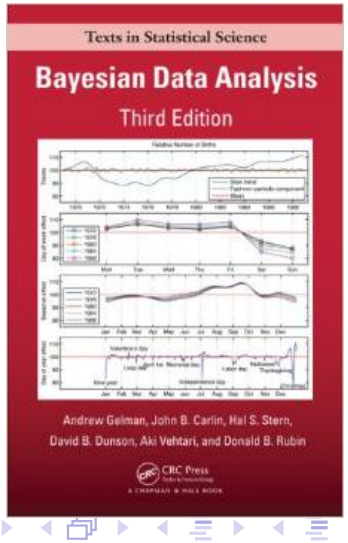
Books

- Lee, S-Y: Structural Equation Modeling: A Bayesian Approach (Wiley, 2007)
- Song, X-Y and Lee, S-Y: Basic and Advanced Bayesian Structural Equation Modeling: With Applications in the Medical an Behavioural Sciences (Wiley, 2012)
- Lunn, D et al: The BUGS book: A Practical Introduction to Bayesian Analysis (CRC Chapman & Hall, 2012)
- Gelman, A et al: Bayesian Data Analysis, 3rd ed (CRC Press, 2013)









On the Web

- BUGS mailing list: `bugs@jiscmail.ac.uk` (discussion, help)
- WinBUGS site: `http://www.mrc-bsu.cam.ac.uk/software/bugs` (examples, manuals)
- OpenBUGS site: `http://www.openbugs.net` (examples, manuals)
- JAGS site: `http://mcmc-jags.sourceforge.net` (examples, manuals, discussion, help)

- MPlus site: `http://statmodel.com/BSEM.shtml`

Miscellaneous Helpful Stuff

- http://statacumen.com/2009/07/02/wishart-distribution-in-winbugs-nonstandard-para
- http://web.as.uky.edu/statistics/users/pbreheny/701/S13/notes/3-28.pdf
- http://www.tc.umn.edu/~nydic001/docs/unpubs/Wishart_Distribution.pdf
- http://en.wikipedia.org/wiki/Inverse-Wishart_distribution
- http://en.wikipedia.org/wiki/Wishart_distribution