Dimension Reduction in R
Essex Summer School in Data Analysis
Lecture 7: Bayesian Models I

Dave Armstrong

Department of Political Science
University of Wisconsin - Milwaukee

e: armstrod@uwm.edu
w: http://www.quantoid.net/teachessex/dimension/

August 19, 2015

Bayesian vs. Frequentist (short version)

Bayesian: Condition on data at hand to produce posterior beliefs that are rational updates of our prior beliefs.

Frequentist: Condition on a null hypothesis to find the plausability of the data (or more extreme data than the ones found) with repeated sampling) with an additional step of reasoning to either reject of fail to reject the null hypothesis.

	Bayesian	Frequentist
Data	Fixed	Variable
Parameters	Variable	Fixed

Why do we care?

In general:

- More straightforward interpretation.
- Statistical Significance is not all it's cracked up to be.
- Easily deal with data that do not represent (repeatable) samples.

This class:

- Allows us to estimate the model of interest on the data of interest.
- Here, it actually doesn't matter that all of the RHS of our regression is unobserved.

Probabilities

Probabilities are defined by Kolmogorov as follows. If Ω is a set of events, $P(A)$ is a function that assigns real numbers to events $A \subset \Omega$, the $P(A)$ are probabilities if:

1. $P(A) \geq 0, \forall A \subset \Omega$
2. $P(\Omega) = \sum_{A}^{\Omega} P(A) = 1$
3. If A and B are disjoint events, then $P(A \cup B) = P(A) + P(B)$.

Objective Probabilities

Objective probabilities are always the function of a long-term relative frequency.

$$P(A) = \lim_{n \to \infty} \frac{m}{n}$$

where m is the number of times event A happens and n is the number of repetitions in which A could have happened.

- Frequentist probabilities are probabilities of objects (e.g., coins, dice, cards, etc...)
- These probabilities are concerned with "the stochastic laws of chance processes" (Hacking 1975, 12, as quoted by Jackman).

Subjective Probabilities

As Hacking (1975, 12, quoted in Jackman) states, subjective probabilities are "dedicated to assessing reasonable degrees of belief in propositions", and as such (need) have no relative frequency interpretation.

- Subjective probabilities are not any set of subjective beliefs, rather they are ones that conform to the axioms of probability (see above).
- Here, subjective beliefs must be updated rationally in light of new information (data) in a manner consistent with the probability axioms.
- Think about how we might update our beliefs about the fairness of a coin over the course of a set of coin flips.
  - This is possible using Bayes' law under the idea of subjective probability.
  - This makes no sense in the objective probability sense because the coin is not expected to be changing its fundamental characteristics.

Bayes Theorem for Continuous Parameters

$$p(\theta|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{\int p(\boldsymbol{y}|\theta)p(\theta)d\theta}$$

$$p(\theta|\boldsymbol{y}) \propto p(\boldsymbol{y}|\theta)p(\theta)$$

where the denominator in the first equation is the "constant of proportionality" ensuring that the posterior is a proper probability distribution (i.e., it integrates to 1).

- The term $p(\boldsymbol{y}|\theta)$ is the likelihood
- The term $p(\theta)$ is the distribution that characterizes our prior beliefs about θ.

Thus, we can say that the posterior is proportional to the prior multiplied by the likelihood.

Priors, Likelihoods and Posteriors

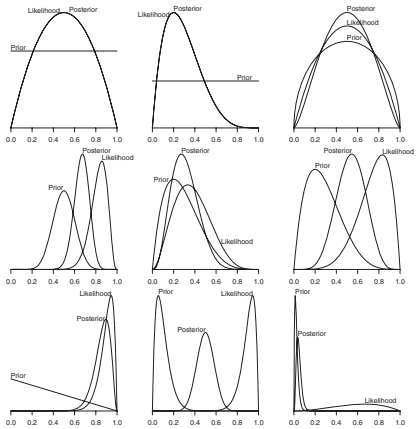


Figure 1.2 Priors, likelihoods and posterior densities. Each panel shows a prior density, a likelihood, and a posterior density over a parameter $\theta \in [0, 1]$. In the top two panels on the left the posterior and the likelihood coincide, since the prior is uniform over the parameter space.

Bayesian Statistics as Information Accumulation (i.e., learning)

The posterior distribution will be more precise than either the prior or the likelihood.

- This precision continues to increase as more data are brought to bear on the problem.
- With enough data, even models with different prior beliefs will converge on the same posterior (provided they are using Bayes Theorem to update beliefs).

Parameters as random variables, Beliefs as distributions

The results of frequentist and Bayesian analysis are quite different.

Frequentist: $\hat{\theta}(\boldsymbol{y})$ is a point-estimate that estimates the population parameter givent the particular dataset (of an infinite set of possible data sets) under consideration.

- The distribution we care about here is the *sampling distribution* which is a purely theoretical distribution describing how the parameters $\hat{\theta}(\boldsymbol{y})$ will change with different $\boldsymbol{y}$.

Bayesian: $p(\theta|\boldsymbol{y})$ is a distribution characterizing our posterior uncertainty about the quantity of interest given the evidence at hand.

- Note that point estimates (e.g., the posterior mean or median) is a loss of a considerable amount of information.

Markov Chains

A Markov chain is a stochastic process:

- physical anaolgy: A particle moving around in space.
- For us, the space is the support of $p(\boldsymbol{\theta}|\text{data})$
- The trajectory is the output of some Monte Carlo algorithm.

Markov chains visit areas of the space with frequencies proportional to the probability of those locations under the density of interest.

Markov Chains

- The state of the chain at time t only depends directly on the state of the chain at time $t-1$.
- Can characterize the probability of taking (a) particular (set of) values with a transition matrix (or kernel). That is, we can characterize $Pr(\theta^{(t)} = i|\theta^{(t-1)} = j)$

Markov Chain Questions of Interest

1. Over successive iterations of the MC, does it gravitate toward one state or the other or to a stable equilibrium distribution over its state space (i.e., the support of $p(\boldsymbol{\theta}|\text{data})$)?
2. If a stationary distribution exists, is it unique?
3. If there is a unique stationary distribution, how long does it take to get there?
4. Can we assess how close we are to the stationary distribution?
5. Can summaries from the trajectory of the MC be taken as summaries of the stationary distribution (i.e., is the observed number of times $\boldsymbol{\theta}$ visits value $\boldsymbol{\theta}^*$ a good estimate of $Pr(\boldsymbol{\theta} = \boldsymbol{\theta}^*)$)?

Markov Chains are...

- Invariant: if $\boldsymbol{\theta}^t \sim p$ then $\boldsymbol{\theta}^{(t+a)} \sim p \quad \forall a > 0$
- Irreducible/Recurrent: The markov chain can reach any region in the state space from $\boldsymbol{\theta}^{(t)}$ with probability greater than zero.
- Reversible: $p(\boldsymbol{\theta}^{(t)})K(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}) = p(\boldsymbol{\theta}^{(t+1)})K(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t+1)})$
- Aperiodic: The chain is not certain to visit the same sequence of states in any given finite amount of time.

A Markov chain with these properties is said to be *ergodic*.

Markov Chain Answers

1. Over successive iterations of the MC, does it gravitate toward one state or the other or to a stable equilibrium distribution over its state space (i.e., the support of $p(\boldsymbol{\theta}|\text{data})$)? Converges to stable distribution - the stationary distribution.
2. If a stationary distribution exists, is it unique? If the chain is irreducible, it is unique.
3. If there is a unique stationary distribution, how long does it take to get there? Can take an arbitrarily long time, but $< \infty$.
4. Can we assess how close we are to the stationary distribution? Yes (total variation norm, but we don't *do* this much).
5. Can summaries from the trajectory of the MC be taken as summaries of the stationary distribution (i.e., is the observed number of times $\boldsymbol{\theta}$ visits value $\boldsymbol{\theta}^*$ a good estimate of $Pr(\boldsymbol{\theta} = \boldsymbol{\theta}^*)$)? Yes.

MCMC

Markov Chain Monte Carlo (MCMC) is a class of techniques that will allow us to use:

- The Monte Carlo principle (indicating that sampling from a density $p(\boldsymbol{\theta})$ can tell us anything we need to know about $\boldsymbol{\theta}$)
- and Markov chain theory, which tells us how that the Monte Carlo principle applies even to non-independent draws
- to figure out how to construct the appropriate transition kernel $K(.,.)$ such that the Markov chain's stationary distribution is $p(\boldsymbol{\theta}|\text{data})$, (the posterior distribution of interest).

We will talk about both the Metropolis-Hastings algorithm and Gibbs Sampling

Questions to Answer

- Are we in the stationary Distribution?
- How many draws from the posterior do we need to properly characterize the posterior distribution (i.e., to get reliable [low MC error] values for the quantities of interest)

No certain answers to these questions.

- Convergence is only ensured to happen before the ∞^{th} iteration.

Potential Problems

- Assumed model (e.g., error distribution assumptions, functional form assumptions, etc...) may not fit well.
- Slow mixing is probably the biggest issue we will face (though some identification problems will arise, too).
- Improper priors can lead to an improper (and meaningless) joint posterior. Proper priors ensure a proper posterior.
- Nonconnected support - the gibbs sampler can give strange results when support over the parameter space is discontinuous.
- Multi-modal posteriors are hard to explore. This is particularly problematic in latent variable models, but can be easily solved with identifying restrictions.

Graphical Diagnostics: Traceplots

A traceplot is a plot that connects the (x, y) pairs of iteration number and sampled chain value.

- If the algorithm has converged, the traceplot will look like a "fuzzy caterpillar".
- Trending indicates a lack of convergence.
- These can be potentially problematic because if you're not looking at a long enough sequence, the chain may look to have converged, whether it has or not.

Example: Running a Regression Model

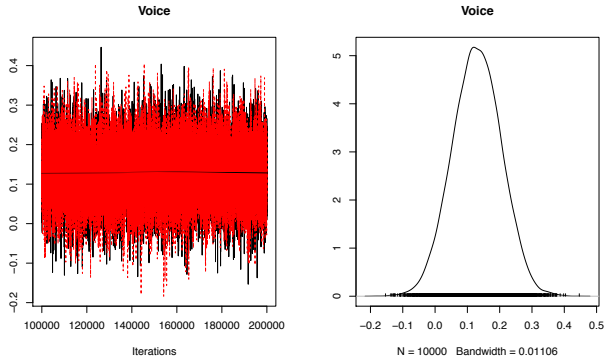
```
> library(foreign)
> dat <- read.dta("http://www.quantoid.net/files/essex/dr.dta")
> library(MCMCpack)
> inits1 <- runif(6,-2,2)
> inits2 <- runif(6,-2,2)
> chain1 <- MCMCregress(rep_mean ~ voice_mean*veto_mean + cwar + iwar,
+     data=dat, burnin = 100000, thin=10, mcmc=100000,
+     beta.start=rep(0,6), seed=1234)
> chain2 <- MCMCregress(rep_mean ~ voice_mean*veto_mean + cwar + iwar,
+     data=dat, burnin = 100000, thin=10, mcmc=100000,
+     beta.start=inits2, seed=5678)
> chains <- mcmc.list(chain1, chain2)
```

Summarizing the MCMC Output

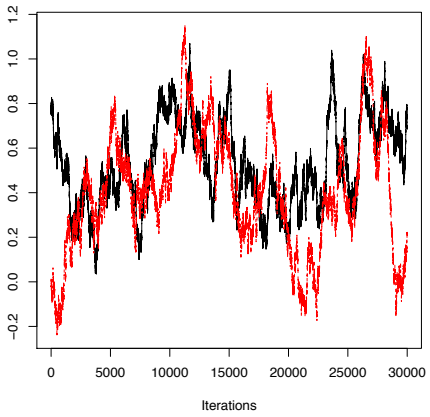
```
> summary(chains)

Iterations = 100001:199991
Thinning interval = 10
Number of chains = 2
Sample size per chain = 10000

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

                         Mean      SD  Naive SE Time-series SE
(Intercept)          -0.7207 0.19828 0.0014021       0.001421
voice_mean            0.1295 0.07563 0.0005348       0.000528
veto_mean             0.5623 0.37236 0.0026330       0.002575
cwar                 -2.1173 0.47534 0.0033612       0.003361
iwar                  0.4029 1.15983 0.0082012       0.008250
voice_mean:veto_mean  0.1562 0.05317 0.0003760       0.000376
sigma2                2.6386 0.31302 0.0022134       0.002194

2. Quantiles for each variable:

                          2.5%      25%     50%     75%   97.5%
(Intercept)          -1.11048 -0.85418 -0.7210 -0.5867 -0.3310
voice_mean           -0.01843  0.07837  0.1296  0.1806  0.2758
veto_mean            -0.16460  0.31353  0.5616  0.8133  1.2993
cwar                 -3.06563 -2.43429 -2.1165 -1.8000 -1.1823
iwar                 -1.86994 -0.37058  0.3933  1.1782  2.6893
voice_mean:veto_mean  0.05179  0.12069  0.1562  0.1917  0.2618
sigma2                2.09488  2.41830  2.6128  2.8296  3.3277
```


Traceplots

```
> plot(chains[,2], main="Voice")
```



Slow mixing chain



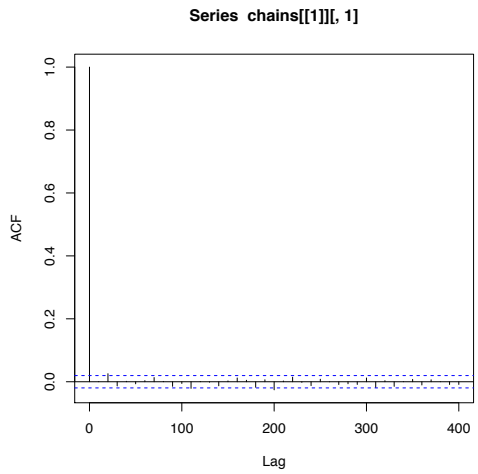
Graphical Diagnostics: Autocorrelation Function

Autocorrelation is a diagnostic that will also tell us how "sticky" the Markov chain is.

- This tells us how correlated draws from one time are with draws from previous times.
- The samplers are most efficient (i.e., able to explore the space in the smallest number of iterations) when the autocorrelation function is a spike at 0, rather than a slowly decaying process.

Two Autocorrelation Functions

```
> acf(chains[[1]][,1])
```



Numerical Diagnostics: Geweke

Geweke tests for stationarity gives essentially a difference of means test across two non-overlapping windows one earlier, one later in the chain.

- Usually the first window is the first 10% of the chain and the second window is the last 50% of the chain.

```
> geweke.diag(chains[[1]])

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

        (Intercept)           voice_mean            veto_mean
            -0.3264               0.1223               0.4864
               cwar                 iwar voice_mean:veto_mean
            -0.6595              -0.4327              -1.5243
             sigma2
             1.3645
```

This is essentially a z-statistic and we should evaluate it as such, considering absolute values above 2ish to be potentially problematic.

Numerical Diagnostics: Heidelberger-Welch

The Heidelberger-Welch diagnostic assesses stationarity and accuracy of the calculated posterior mean.

- Calculates a stationarity test throwing away the first 10%, then 20% up to 50% of the data until the null hypothesis of stationarity is not rejected. If after 50% the null is still rejected, the test fails.
- If the chain passes stationarity, a "half-width" test is conducted which takes the ratio of the half-width of the 95% credible interval to the mean. If this value is greater than 0.1, the test fails.
- The test also identifies how many draws are thought to be draws from the stationary posterior.

Example: Heidelberger-Welch

```
> heidel.diag(chains[[1]])

                    Stationarity start     p-value
                    test         iteration
(Intercept)         passed       1         0.570
voice_mean          passed       1         0.304
veto_mean           passed       1         0.180
cwar                passed       1         0.318
iwar                passed       1         0.712
voice_mean:veto_mean passed      1         0.209
sigma2              passed       1         0.585

                    Halfwidth Mean   Halfwidth
                    test
(Intercept)         passed    -0.722 0.00398
voice_mean          passed     0.129 0.00150
veto_mean           passed     0.562 0.00728
cwar                passed    -2.115 0.00939
iwar                passed     0.390 0.02308
voice_mean:veto_mean passed    0.156 0.00104
sigma2              passed     2.640 0.00604
```

Numerical Diagnostics: Raftery-Lewis

Raftery-Lewis is a run-length diagnostic indicating how many iterations you would need to have from a Markov chain (with the similar dependence structure) to estimate a quantity of interest (the 2.5^{th} percentile) to a certain specified degree of accuracy (± 0.005) with a certain probability (e.g., 0.95).

```
> raftery.diag(chains[[1]])

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

                      Burn-in  Total Lower bound  Dependence
                      (M)      (N)   (Nmin)       factor (I)
 (Intercept)          20       37410 3746          9.99
 voice_mean           20       38020 3746         10.10
 veto_mean            30       40280 3746         10.80
 cwar                 20       37410 3746          9.99
 iwar                 20       37710 3746         10.10
 voice_mean:veto_mean 20       37100 3746          9.90
 sigma2               20       37100 3746          9.90
```


Numerical Diagnostics: Gelman-Rubin

Gelman-Rubin compares within-chain variance to between-chain variance (a classical analysis of variance) with values below 1.2 (or so) deemed 'acceptable'.

```
> gelman.diag(chains)

Potential scale reduction factors:

                      Point est. Upper C.I.
(Intercept)                    1          1
voice_mean                     1          1
veto_mean                      1          1
cwar                           1          1
iwar                           1          1
voice_mean:veto_mean           1          1
sigma2                         1          1

Multivariate psrf

1
```


Summarizing Posteriors: Highest Posterior Density

The highest posterior density is calculated using the empirical CDF.

- It is the shortest distance in the ecdf such that the distance between the two points covers (1-α) probability.

```
> library(runjags)
> both <- combine.mcmc(chains)
> HPDinterval(both)

                           lower      upper
(Intercept)          -1.12982529 -0.3522811
voice_mean           -0.01705858  0.2768510
veto_mean            -0.18279661  1.2768284
cwar                 -3.03799184 -1.1575592
iwar                 -1.88371178  2.6703135
voice_mean:veto_mean  0.05113459  0.2608330
sigma2                2.03102115  3.2499899
attr(,"Probability")
[1] 0.95
```


Factor Analysis

The factor analysis model assumes that the indicators are continuous (or roughly so) and that the underlying relationship between the latent variable and observed variables is linear.

$$y_{ij} = \lambda_j z_i + \delta_{ij}$$

where

- y_{ij} is the observation for the i^{th} observation on the j^{th} observed variable (aka manifest variable, indicators)
- λ_j is often called a "factor loading" - it relates the latent variable to the observed variable. We assume that the data are at least centered, but probably scaled as well.
- z_i is the latent variable estimate for observation i.
- δ_{ij} is an iid idiosyncratic error.

Note that all of the terms on the RHS of the equation are parameters to be estimated.

Factor Analysis: Frequentist vs. Bayesian

Frequentists solve the problem above with an eigen decomposition.

- This works on the correlation matrix rather than the data itself.
- Latent variable estimates are a post-hoc consideration (they are not estimated directly by the model, and there is some debate about the right way to do this)
- Because of the nature of the solution, extensions to hierarchical settings is quite difficult

Bayesians solve the problem by sampling from the joint posterior of all parameters.

- This works on the data directly rather than the correlation/covariance matrix
- Latent variable estimates (with uncertainty measures) are produced necessarily as a function of model estimation.
- Because of the nature of the solution, extensions to hierarchical settings is straightforward.

Identification

Likelihood:

$$\mathcal{L} = p(\boldsymbol{Y}|\boldsymbol{\theta}) \prod_{i=1}^{N} \prod_{j=1}^{J} \phi \left(\frac{y_{ij} - \lambda_j z_i}{\omega_j} \right)$$

where the numerator is just the residual from the equation above and ω_j is the residual standard deviation (also referred to as the uniqueness).

- The z and λ parameters are not jointly identified
- The λ, z and ω parameters are also not jointly identified

Normalization can help fix this lack of identification.

- We can fix the mean and variance of z as well as put a sign restriction on one λ.
- We could fix one λ to a particular value and fix the mean of z.
- Other possibilities also arise.

Identification

Identification in these models is always a bit tricky (or at least something with which you have to be concerned). Here are some common conventions that will be useful for MCMCpack

- Set one coefficient to one for each latent dimension. This allows the variance of the latent variable
- Other coefficients you could (or not) set to zero as you like.

Bayesian FA using MCMCpack

```
> dr <- read.dta("http://www.quantoid.net/files/essex/demrep.dta")
> cons <- list(xconst = c(2,0), xconst = list(1, "+"), polconiii = c(2,0),
+     lgates = c(2,0), log_checks=c(2,0), disap = c(1,0), kill = c(1,0),
+     tort=c(1,0), tort=list(2,"+"))
> X <- as.data.frame(scale(dr[which(dr$year == 2000),-(1:3)]))
> linits1 <- linits2 <- matrix(0, ncol=2, nrow=8)
> linits1[cbind(c(2,3,4,5,5,6,8), c(1,1,1,1,2,2,2) )] <- runif(7, 0,1)
> linits2[cbind(c(2,3,4,5,5,6,8), c(1,1,1,1,2,2,2) )] <- runif(7, 0,1)
> fa1 <- MCMCfactanal(~., X, factors=2, burnin=100000, lambda.start = linits1,
+     mcmc=50000, thin=5, lambda.constraints=cons, seed=2002)
> fa2 <- MCMCfactanal(~., X, factors=2, burnin=100000, lambda.start = linits2,
+     mcmc=50000, thin=5, lambda.constraints=cons, seed=1001)
> chains <- mcmc.list(fa1, fa2)
```

Diagnostics

Heidelberger and Welch Looks pretty good
Geweke Looks not great.
Gelman Looks good.
Trace Plots Look OK.

Saving Factor Scores

You can save the factor scores by setting `save.scores=TRUE`.

- Do this after you've done the convergence checks. If everything else looks OK, so will the distributions of the scores.
- Run for longer burnin and shorter monitoring when you save scores to reduce the pressure on your machine's memory.

Exercise

Using the data below (choose one year to make things go a bit faster),

```
> fh <- read.csv("http://quantoid.net/files/essex/jpr_replication.csv")
```

1. Estimate a one-factor solution for variables `A`, `B` and `C`.
2. Estimate a two factor solution that also includes variables `D`, `E` and `F`.
3. For both models, get check convergence and get the scores.
4. Compare the scores here to those from a conventional FA model.