Gibbs sampling cont'd

Dr. Olanrewaju Michael Akande

Feb 7, 2020

Announcements

- Homework 4 now online.
- Quiz I next Wednesday, Feb 12.
- Survey I for the course coming soon.

Outline

- Gibbs sampler for normal model
- Inference for Pygmalion data
- MCMC diagnostics
- Chat on Quiz I

Recap of normal model

- Suppose we have a normal model as our sampling distribution and wish to specify our uncertainty about μ as independent of τ.
- That is, we want $\pi(\mu, \tau) = \pi(\mu)\pi(\tau)$.
- For example,

$$\begin{aligned} \mu &\sim \mathcal{N}\left(\mu_0, \sigma_0^2\right). \\ \tau &\sim \text{Gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0}{2\tau_0}\right). \end{aligned}$$

- When σ_0^2 is not proportional to $\frac{1}{\tau}$, the marginal density of τ is not a gamma density (or a density we can easily sample from).
- We can't sample from the joint posterior like we are used to, we need to do Gibbs sampling.

Full conditionals

- That is, we need

$$\mu | Y, \tau \sim \mathcal{N}(\mu_n, \tau_n^{-1}),$$

where

- $\mu_n = \dfrac{\frac{\mu_0}{\sigma_0^2} + n\tau\bar{y}}{\frac{1}{\sigma_0^2} + n\tau}$; and
- $\tau_n = \frac{1}{\sigma_0^2} + n\tau$.

Full conditionals

- and

$$\tau | \mu, Y \sim \text{Gamma}\left(\frac{\nu_n}{2}, \frac{\nu_n \sigma_n^2(\mu)}{2}\right),$$

where

$$\nu_n = \nu_0 + n$$

$$\sigma_n^2(\mu) = \frac{1}{\nu_n}\left[\frac{\nu_0}{\tau_0} + \sum_{i=1}^{n}(y_i - \mu)^2\right] = \frac{1}{\nu_n}\left[\frac{\nu_0}{\tau_0} + n s_n^2(\mu)\right]$$

$$\text{with} \quad s_n^2(\mu) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2 \quad \Rightarrow \quad n s_n^2(\mu) = (n-1)s^2 + n(\bar{y} - \mu)^2$$

Recall the Pygmalion data

- For now, let's focus only on the accelerated group for the Pygmalion data.
- Data for accelerated group (A): 20, 10, 19, 15, 9, 18.
- Summary statistics: $\bar{y}_A = 15.2$; $s_A = 4.71$.
- Suppose we assume these improvement scores are normal with mean μ and variance $\frac{1}{\tau}$.
- Suppose for μ, we use a $\mathcal{N}(0, 100)$ prior, and for τ we use a $\mathrm{Ga}(\frac{1}{2}, 50)$ prior.
- Matching with

$$\begin{aligned} \mu &\sim \mathcal{N}\left(\mu_0, \sigma_0^2\right). \\ \tau &\sim \mathrm{Gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0}{2\tau_0}\right), \end{aligned}$$

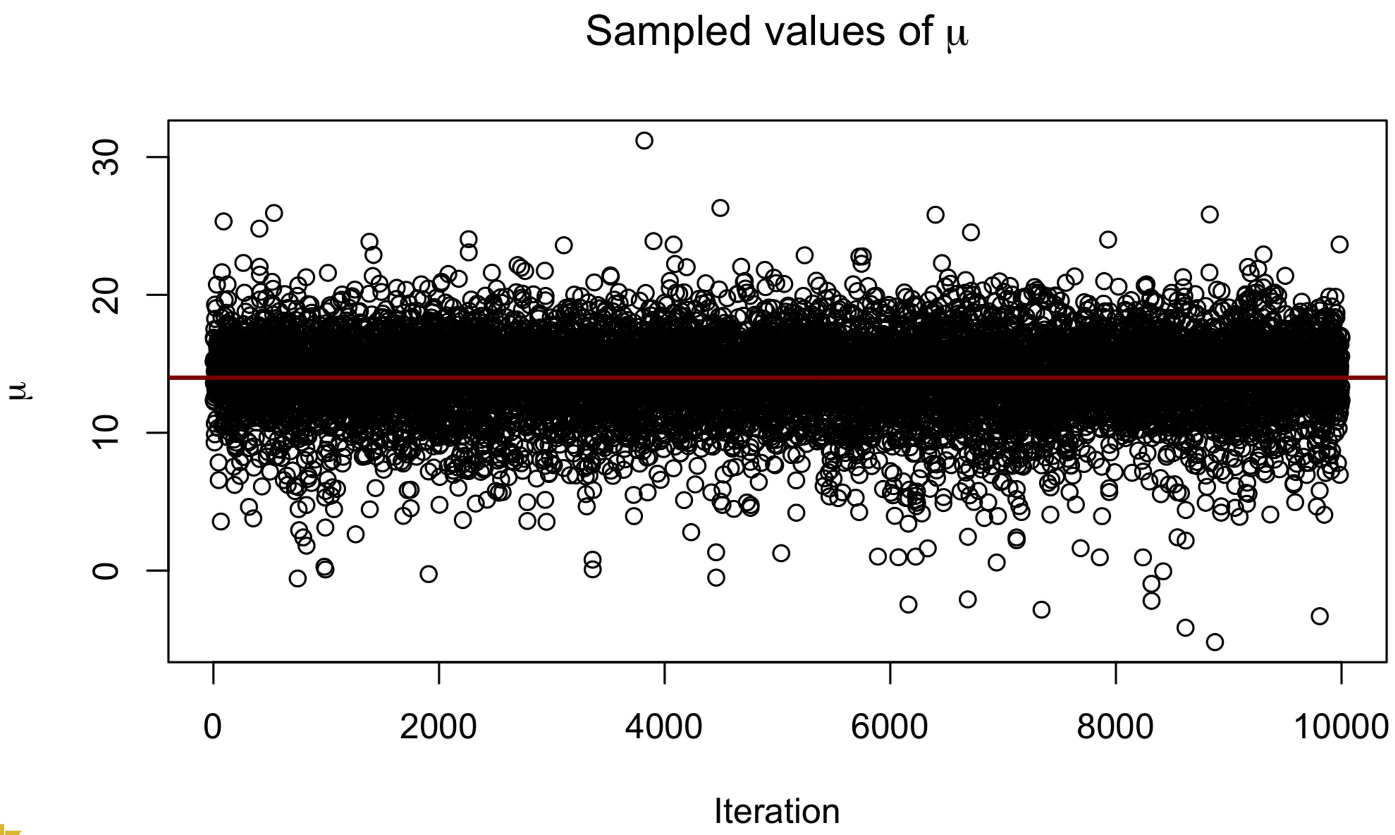
we have: $\mu_0 = 0$, $\sigma_0^2 = 100$, $\nu_0 = 1$ and $\tau_0 = 1/100$.

Gibbs sampling for Pygmalion data

```r
y <- c(20,10,19,15,9,18) #data
y_bar <- mean(y); s2 <- var(y); n <- length(y) #sample statistics you'll need

S <- 10000 # number of samples to draw

PHI <- matrix(nrow=S,ncol=3); #matrix to save results
colnames(PHI) <- c("mu","tau","sigma2")
PHI[1,] <- phi <- c(y_bar,1/s2,s2) #starting values are MLEs

mu0 <- 0; sigma02 <- 100; nu0 <- 1; tau0 <- 1/100 #hyperparameters


###### Gibbs sampler ######
set.seed(1234) #to replicate results exactly
for(s in 2:S) {
#first, draw new mu
taun <- 1/sigma02 + n*phi[2]
mun <- (mu0/sigma02 + n*y_bar*phi[2])/taun
phi[1] <- rnorm(1,mun,sqrt(1/taun))

#now, draw new tau/sigma2
nun <- nu0+n
#trick to speed up calculation of sum(y_i-\mu)^2
s2nmu <- (nu0/tau0 + (n-1)*s2 + n*(y_bar-phi[1])^2)/nun
phi[2] <- rgamma(1,nun/2,nun*s2nmu/2)
phi[3] <- 1/phi[2] #save sigma2

#save the current joint draws
PHI[s,] <- phi
}
###### End of Gibbs sampler ######
```

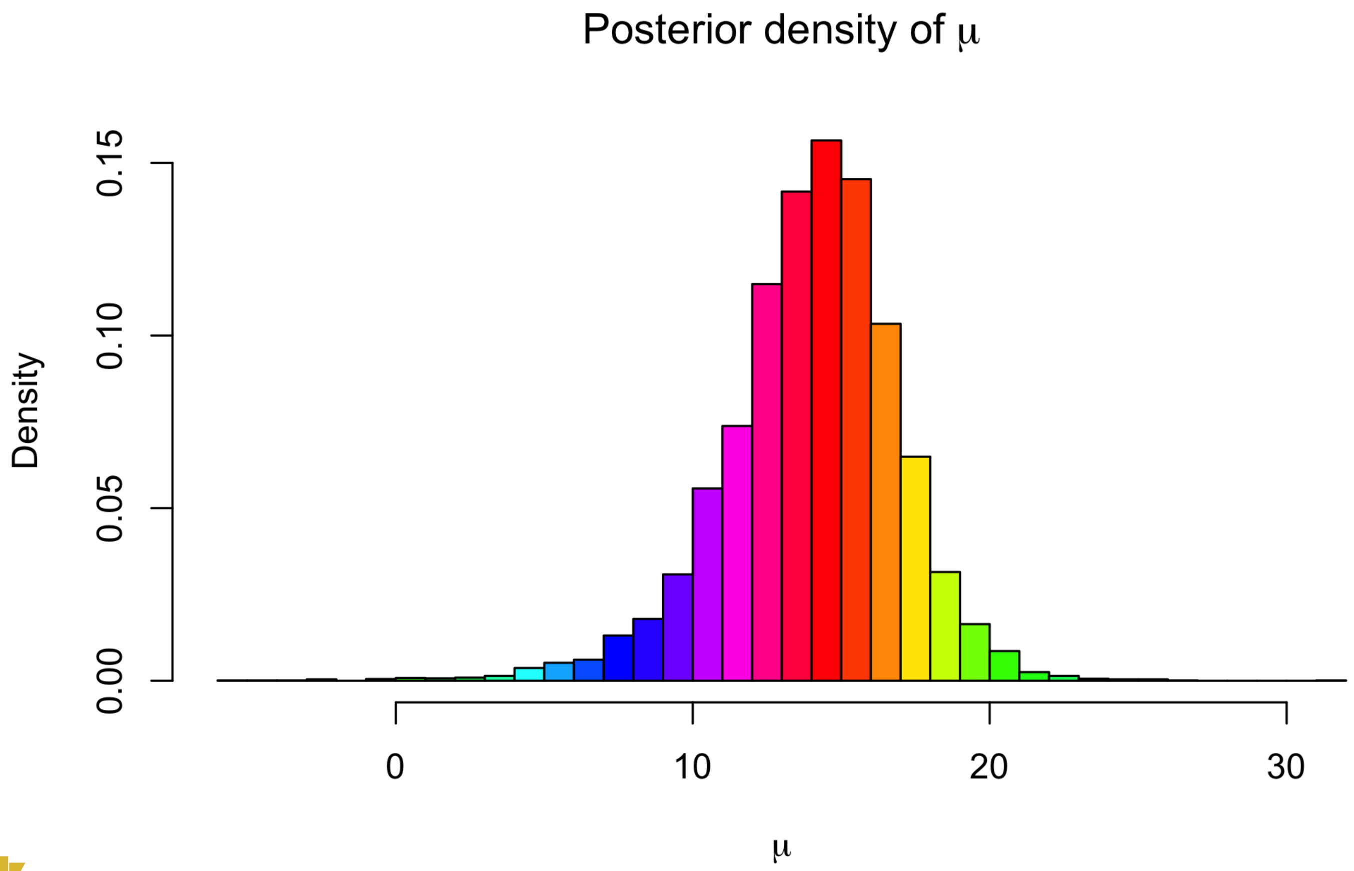
Pygmalion data: mean

```r
plot(PHI[,1],ylab=expression(mu),xlab="Iteration",
     main=expression(paste("Sampled values of ",mu)))
abline(a=mean(PHI[,1]),b=0,col="red4",lwd=2)
```



Pygmalion data: mean

```
hist(PHI[,1],col=rainbow(20),xlab=expression(mu),ylab="Density",freq=F,breaks=50,
     main=expression(paste("Posterior density of ",mu)))
```



Pygmalion data: mean

```r
round(mean(PHI[,1]),3)
```

```
## [1] 13.99
```

```r
round(quantile(PHI[,1],c(0.025,0.5,0.975)),3)
```

```
##   2.5%    50%  97.5%
##  7.520 14.217 19.277
```

Posterior summaries for μ:

- Posterior mean $= 14$.
- Posterior median $= 14.22$.
- 95% credible interval $= (7.52, 19.28)$.

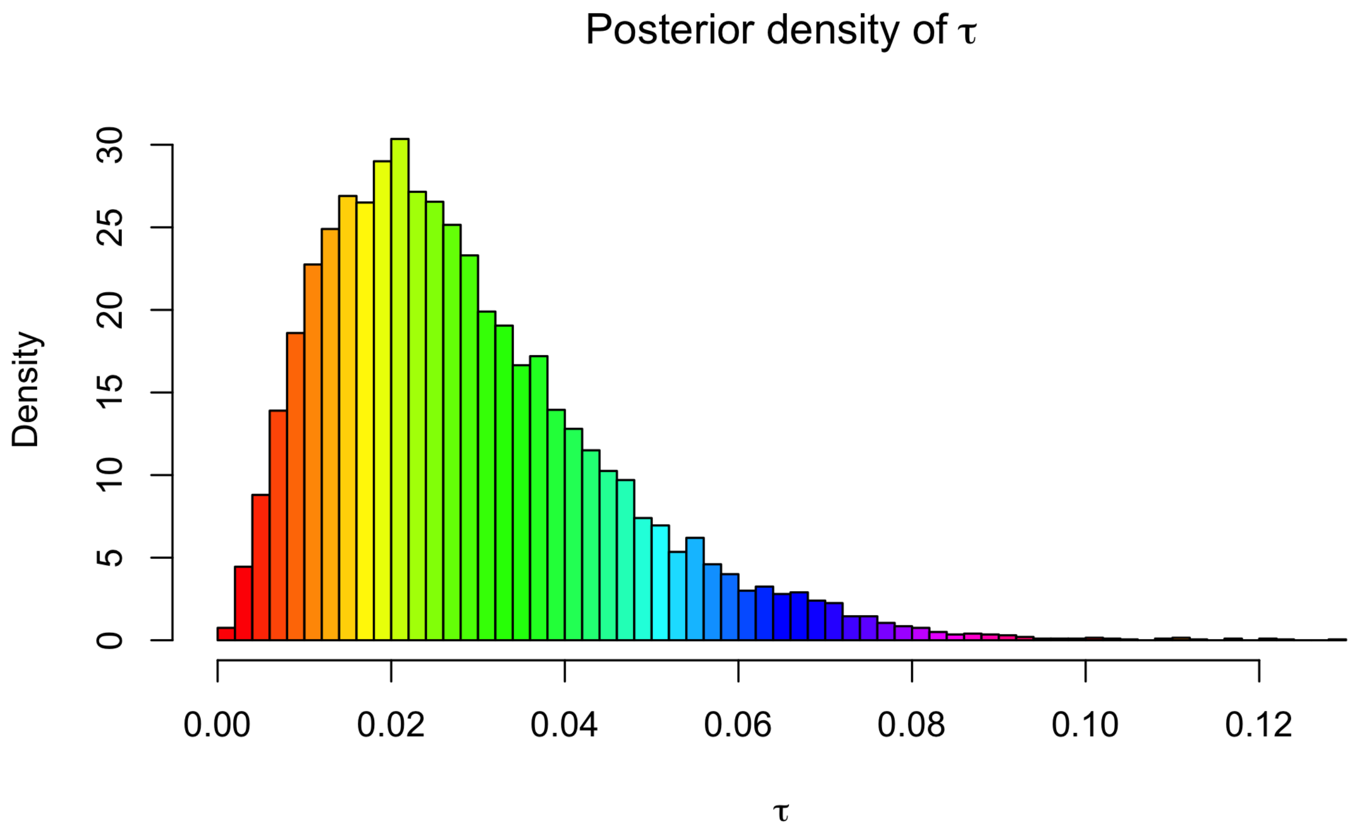
For context, $\bar{y}_A = 15.2$, and we used a $\mathcal{N}(0, 100)$ prior for μ.

Pygmalion data: precision

```r
plot(PHI[,2],ylab=expression(tau),xlab="Iteration",
     main=expression(paste("Sampled values of ",tau)))
abline(a=mean(PHI[,2]),b=0,col="red4",lwd=2)
```

Pygmalion data: precision

```
hist(PHI[,2],col=rainbow(50),xlab=expression(tau),ylab="Density",freq=F,breaks=50,
     main=expression(paste("Posterior density of ",tau)))
```



Pygmalion data: precision

```r
round(mean(PHI[,2]),3)
```

```
## [1] 0.028
```

```r
round(quantile(PHI[,2],c(0.025,0.5,0.975)),3)
```

```
##  2.5%   50% 97.5% 
## 0.006 0.025 0.069 
```

Posterior summaries for τ:

- Posterior mean $= 0.028$.
- Posterior median $= 0.025$.
- 95% credible interval $= (0.006, 0.069)$.

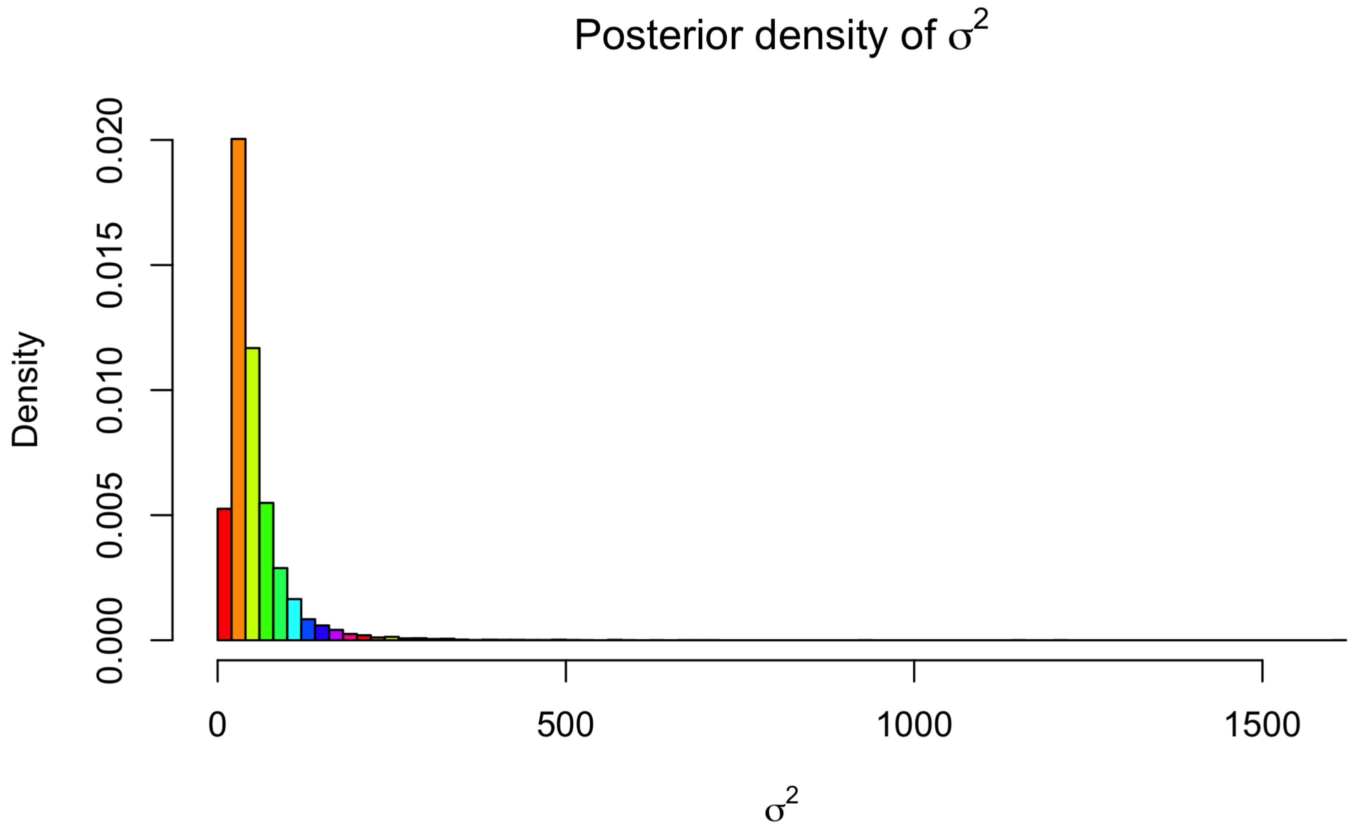
For context, $s_A = 4.71$, which means sample precision $= 1/4.71^2 = 0.045$. Also, we used a $\text{Ga}(\frac{1}{2}, 50)$ prior for τ.

Pygmalion data: variance

```r
plot(PHI[,3],ylab=expression(sigma^2),xlab="Iteration",
     main=expression(paste("Sampled values of ",sigma^2)))
abline(a=mean(PHI[,3]),b=0,col="red4",lwd=2)
```

Pygmalion data: variance

```
hist(PHI[,3],col=rainbow(10),xlab=expression(sigma^2),ylab="Density",freq=F,breaks=100,
     main=expression(paste("Posterior density of ",sigma^2)))
```



Pygmalion data: variance

```r
round(mean(PHI[,3]),2)
```

```
## [1] 53.34
```

```r
round(quantile(PHI[,3],c(0.025,0.5,0.975)),2)
```

```
##   2.5%    50%  97.5%
##  14.52  39.60 174.11
```

Posterior summaries for σ^2:

- Posterior mean $= 53.34$.
- Posterior median $= 39.60$.
- 95% credible interval $= (14.52, 174.11)$.

For context, $s_A = 4.71$, which means sample variance $4.71^2 = 22.18$. Again, we used a $\mathrm{Ga}(\frac{1}{2}, 50)$ prior for τ.

Some terminology

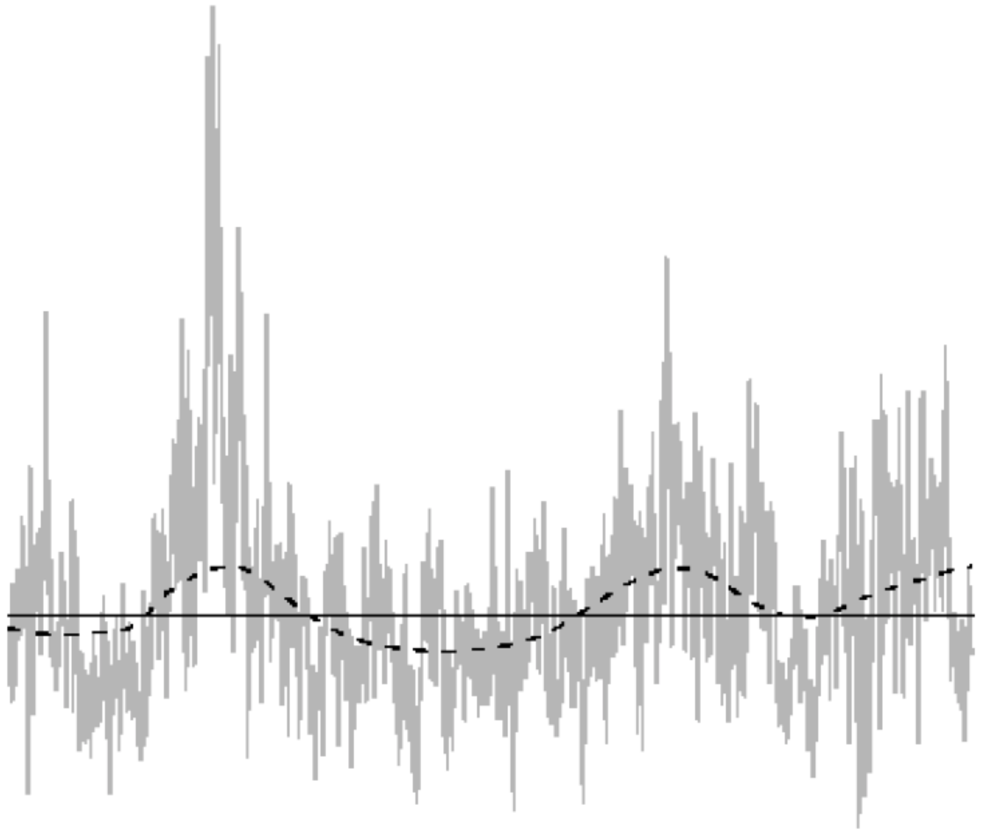
- Convergence: bypassing initial drift in the samples towards a stationary distribution.
- Burn-in: samples at start of the chain that are discarded to allow convergence.
- Trace plot: plot of sampled values of a parameter vs iterations.
- Slow mixing: tendency for high autocorrelation in the samples.
- Thinning: practice of collecting every kth iteration to reduce autocorrelation. It gets you a little closer to iid draws and saves memory (you don't store all draws), but unless memory is a major issue or autocorrelation is very high, it is not generally advantageous to thin the chain.

Burn-in

- Because convergence often occurs regardless of our starting point (in not-too-complex problems at least), we can usually pick any reasonable values in the parameter space as a starting point.
- The time it takes for the chain to converge may vary depending on how close the starting values are to a high probability region of the posterior.
- Generally, we throw out a certain number of the first draws, known as the **burn-in**, as an attempt to make our draws closer to the stationary distribution and less dependent on any single set of starting values.
- However, we don't know exactly when convergence occurs, so it is not always clear how much burn-in we would need.

Example - trace plot with bad mixing

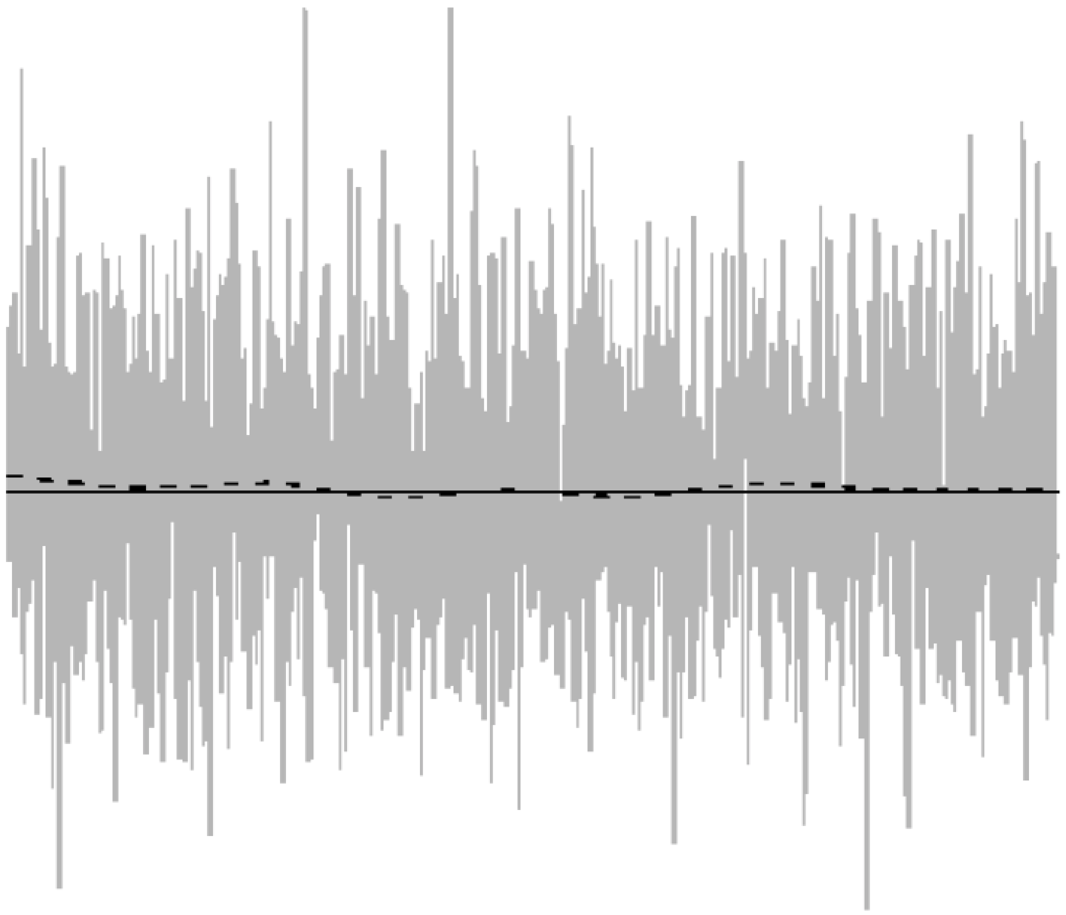
- Trace plot: plot of sampled values of a parameter vs iterations.



Poor mixing

- Exhibits "snaking" behavior in trace plot with cyclic local trends in the mean.
- Poor mixing in the Gibbs sampler caused by high posterior correlation in the parameters.
- Decreases efficiency & many more samples need to be collected to maintain low Monte Carlo error in posterior summaries.
- For very poor mixing chain, may even need millions of iterations.
- Routinely examine trace plots!

Example - trace plot with good mixing



Convergence diagnostics

- Diagnostics available to help decide on number of burn-in & collected samples.
- **Note**: no definitive tests of convergence & you should check convergence of all parameters.
- With "experience", visual inspection of trace plots perhaps most useful approach.
- There are a number of useful automated tests in R.

Diagnostics in R

- The most popular package for MCMC diagnostics in R is `coda`.
- `coda` uses a special MCMC format so you must always convert your posterior matrix into an MCMC object.
- Continuing with the posterior samples for the Pygmalion study, we have the following in R.

```r
#library(coda)
phi.mcmc <- mcmc(PHI,start=1) #no burn-in (simple problem!)
```

Diagnostics in R

```
summary(phi.mcmc)
```

```
##
## Iterations = 1:10000
## Thinning interval = 1
## Number of chains = 1
## 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
## mu     13.98961  2.94748 0.0294748      0.0341435
## tau     0.02839  0.01646 0.0001646      0.0001855
## sigma2 53.34388 53.27616 0.5327616      0.6502608
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%     97.5%
## mu      7.519819 12.36326 14.21682 15.84203  19.27701
## tau     0.005744  0.01626  0.02526  0.03726   0.06886
## sigma2 14.522591 26.83933 39.59569 61.49382 174.10833
```

The naive SE is the **standard error of the mean**, which captures simulation error of the mean rather than the posterior uncertainty.

The time-series SE adjusts the naive SE for **autocorrelation**.

Effective sample size

- The effective sample size translates the number of MCMC samples S into an equivalent number of independent samples.
- It is defined as

$$\text{ESS} = \frac{S}{1 + 2\sum_k \rho_k},$$

where S is the sample size and ρ_k is the lag k autocorrelation.

- For our data, we have

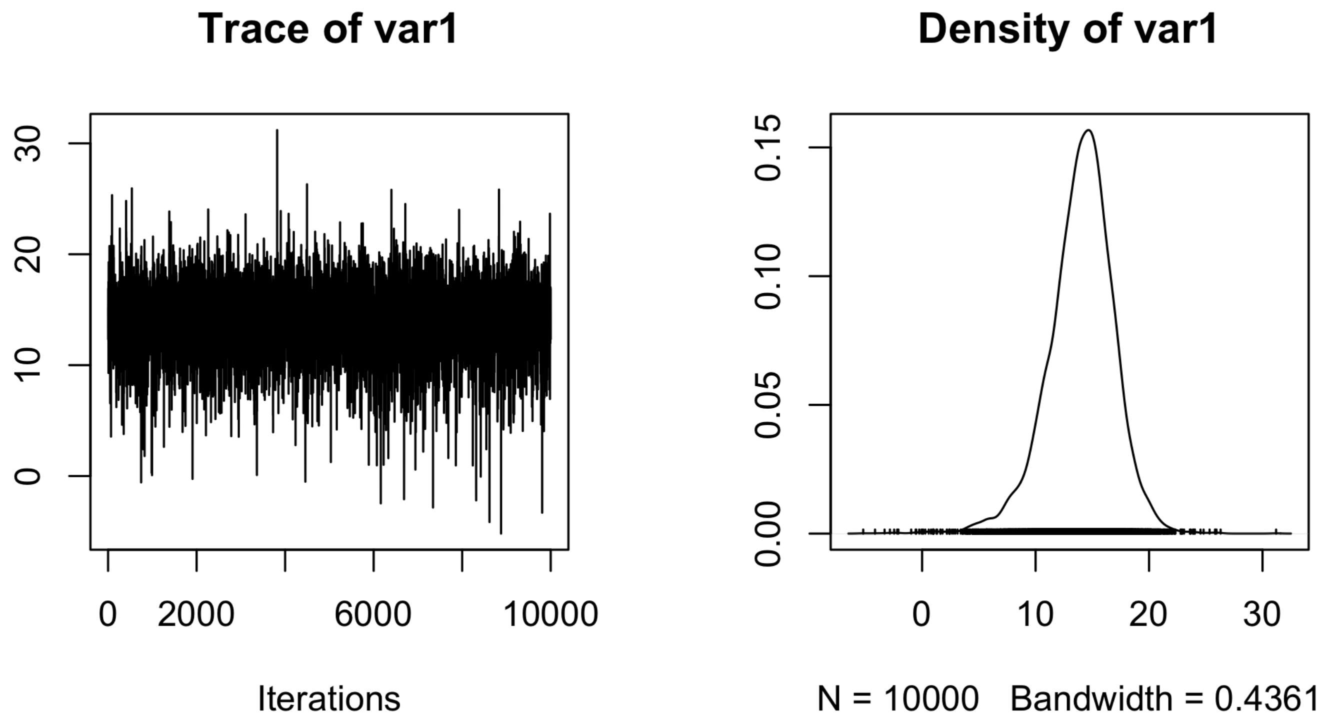
```
effectiveSize(phi.mcmc)
```

```
##       mu      tau   sigma2
## 7452.197 7877.721 6712.600
```

- So our 10,000 samples are equivalent to 7452 independent samples for μ, 7878 independent samples for τ, and 6713 independent samples for σ^2.

Trace plot for mean

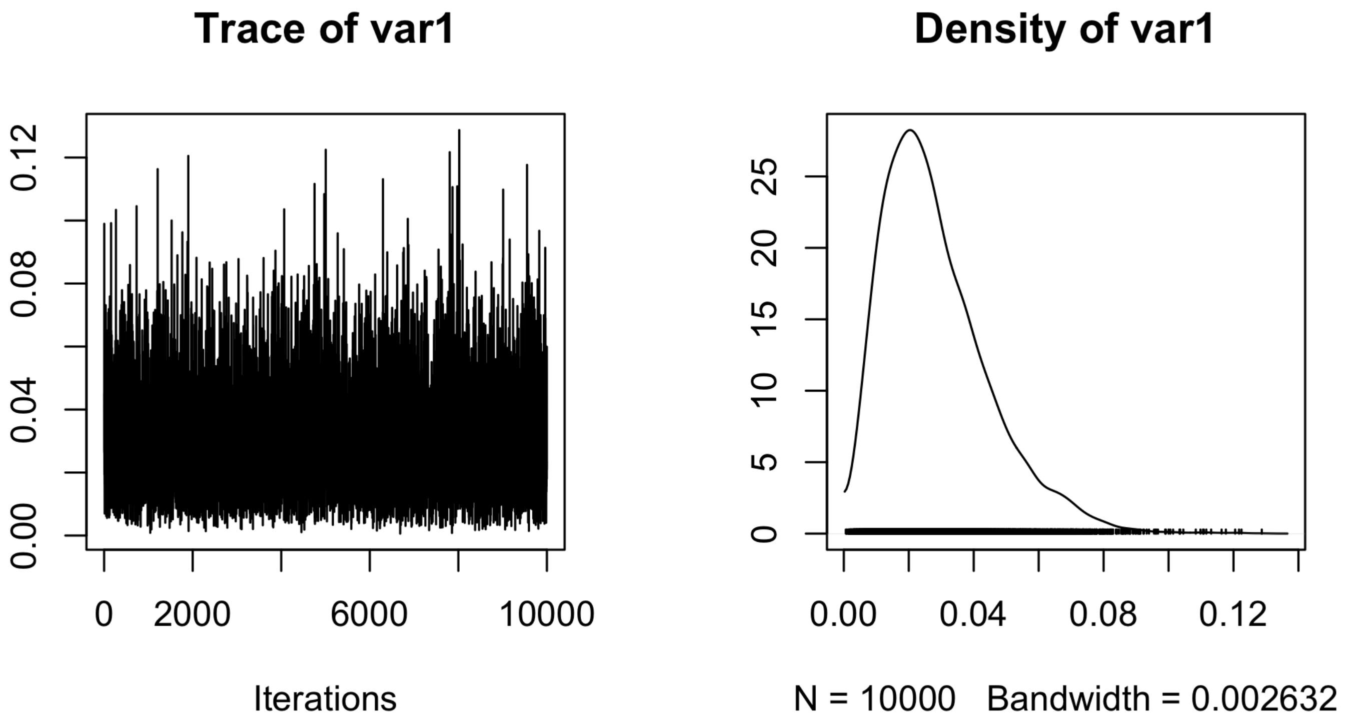
```
plot(phi.mcmc[,"mu"])
```



Looks great!

Trace plot for precision

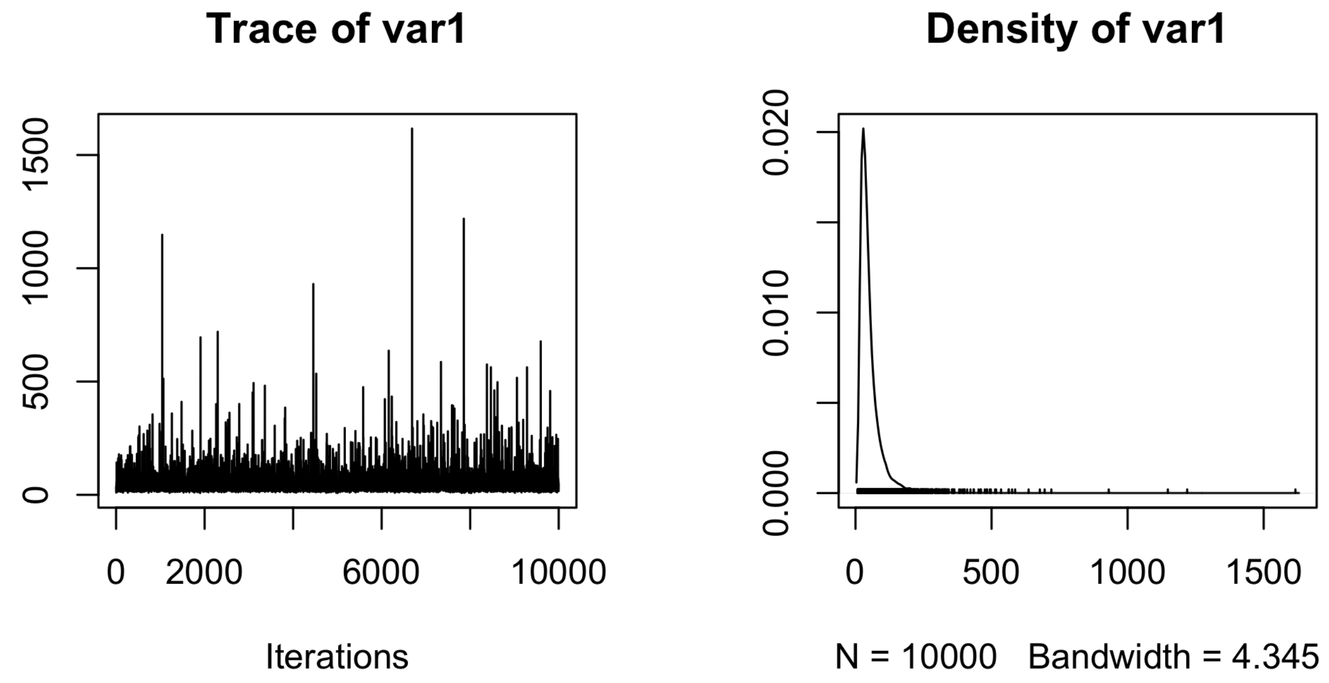
```
plot(phi.mcmc[,"tau"])
```



Looks great!

Trace plot for variance

```
plot(phi.mcmc[,"sigma2"])
```



We do see a few wacky samples that we did not see with τ, due to the scale. Generally, still looks great!

Autocorrelation

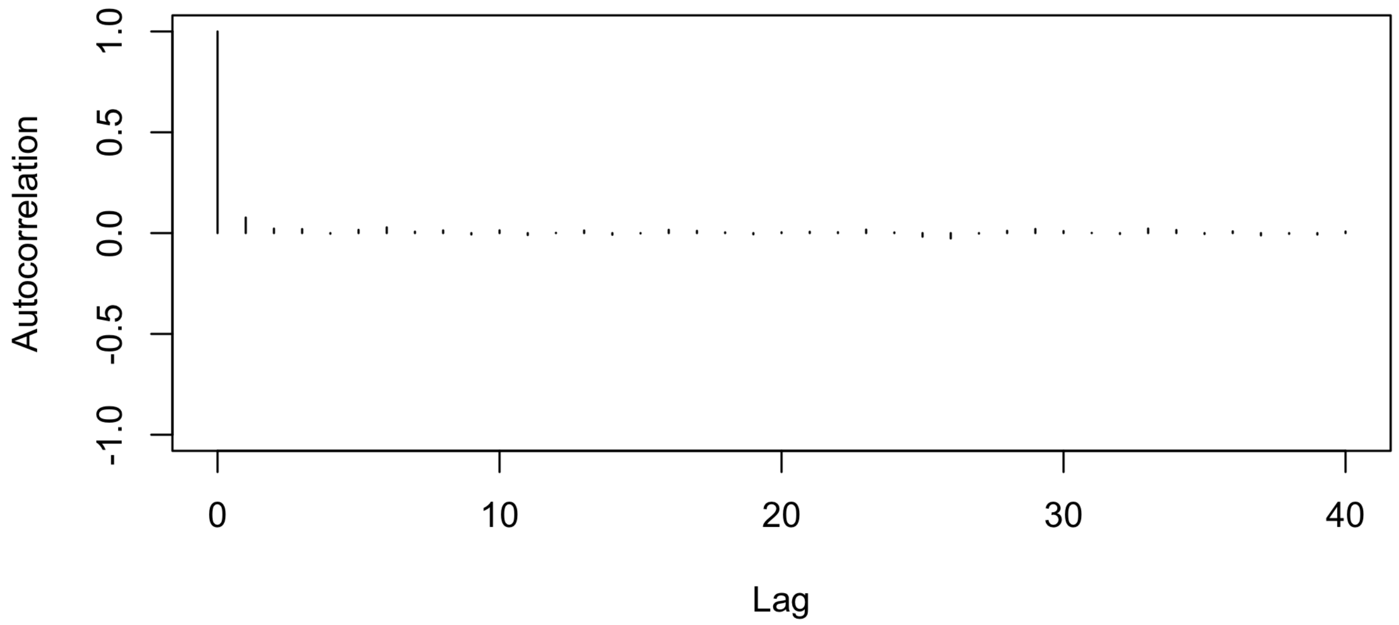
- Another way to evaluate convergence is to look at the autocorrelation between draws of our Markov chain.
- The lag k autocorrelation, ρ_k, is the correlation between each draw and its kth lag, defined as

$$\rho_k = \frac{\sum_{s=1}^{S-k}(\theta_s - \bar{\theta})(\theta_{s+k} - \bar{\theta})}{\sum_{s=1}^{S-k}(\theta_s - \bar{\theta})^2}.$$

- We expect the autocorrelation to decrease as k increases.
- If autocorrelation remains high as k increases, we have slow mixing due to the inability of the sampler to move around the space well.

Autocorrelation for mean

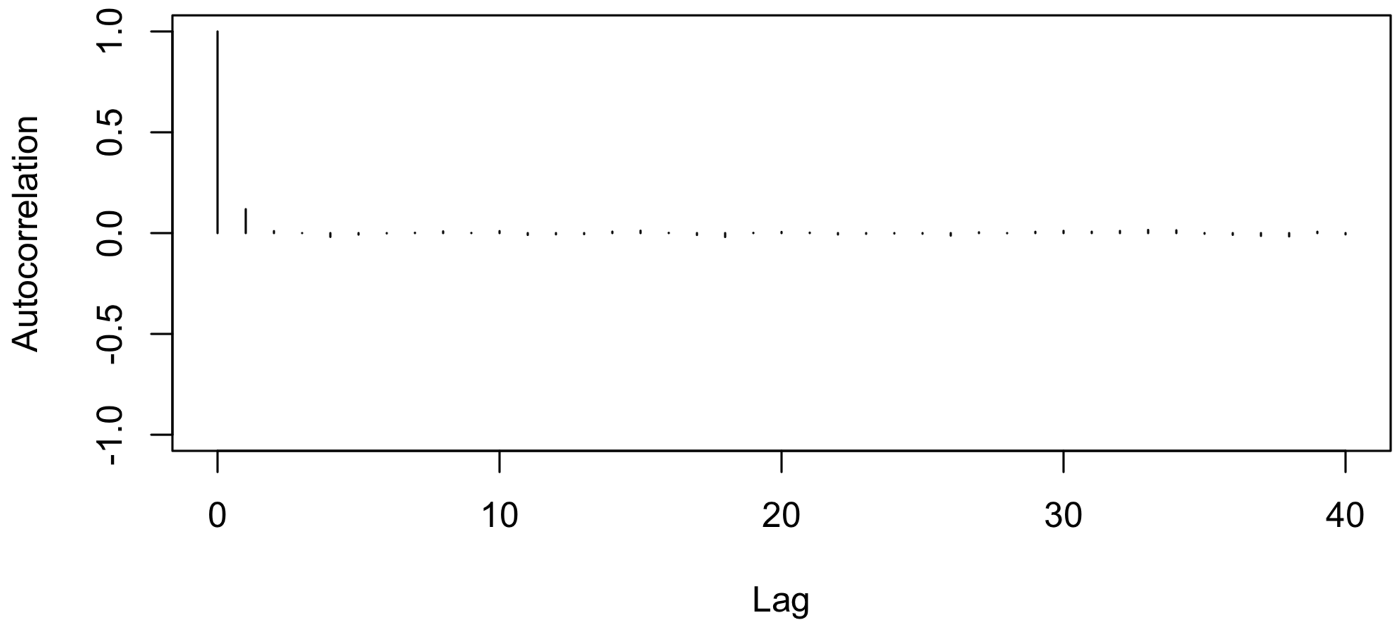
```r
autocorr.plot(phi.mcmc[,"mu"])
```



This looks great! Look how quickly autocorrelation goes to 0.

Autocorrelation for precision

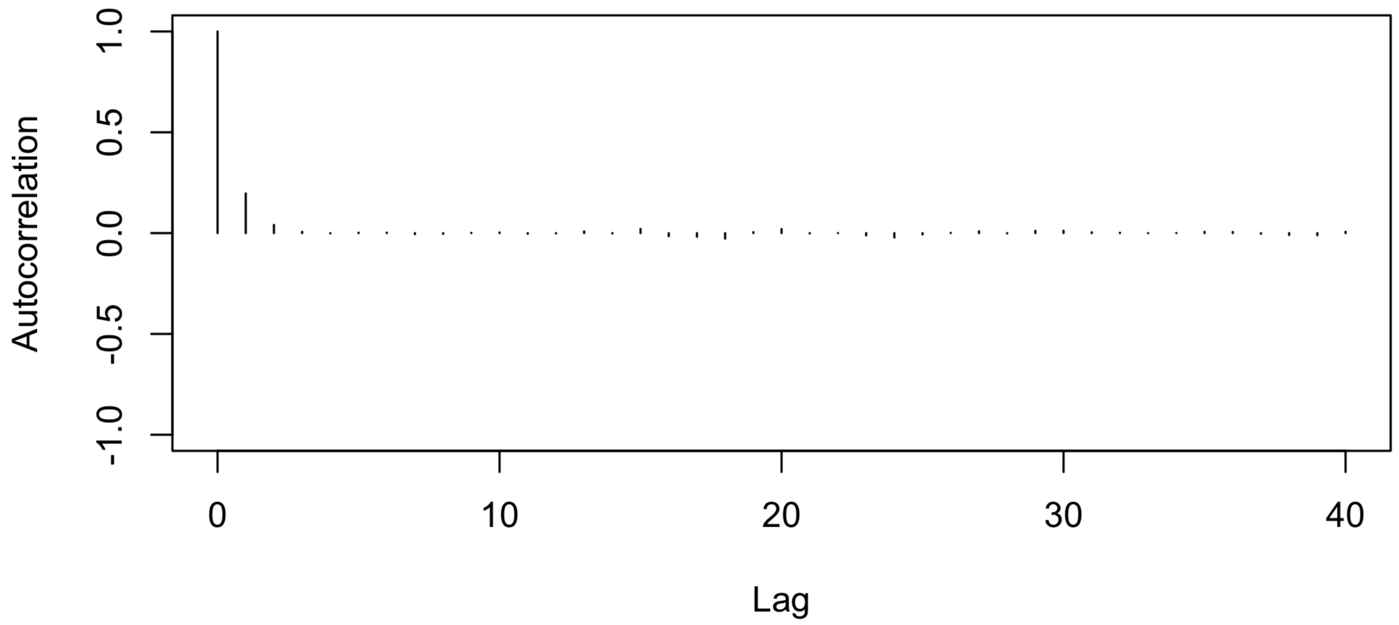
```
autocorr.plot(phi.mcmc[,"tau"])
```



Also great!

Autocorrelation for variance

```
autocorr.plot(phi.mcmc[,"sigma2"])
```



Also great!

Gelman and Rubin Statistic

- Andrew Gelman and Don Rubin suggested a diagnostic statistic based on taking separate sets of Gibbs samples (multiple chains) with dispersed initial values to test convergence.
- The algorithm proceeds as follows.
  - Run m > 2 chains of length 2S from overdispersed starting values.
  - Discard the first S draws in each chain.
  - Calculate the within-chain and between-chain variance.
  - Calculate the estimated variance of the parameter as a weighted sum of the within-chain and between-chain variance.
  - Calculate the potential scale reduction factor

$$\hat{R} = \sqrt{\frac{\hat{\text{Var}}(\theta)}{W}},$$

where $\hat{\text{Var}}(\theta)$ is the weighted sum of the within-chain and between-chain variance and W is the mean of the variances of each chain (average within-chain variance).

Geweke statistic

- Geweke proposed taking two non-overlapping parts of a single Markov chain (usually the first 10% and the last 50%) and comparing the mean of both parts, using a difference of means test.
- The null hypothesis would be that the two parts of the chain are from the same distribution.
- The test statistic is a z-score with standard errors adjusted for autocorrelation, and if the p-value is significant for a variable, you need more draws.
- The output is the z-score itself (not the p-value).

```
geweke.diag(phi.mcmc)
```

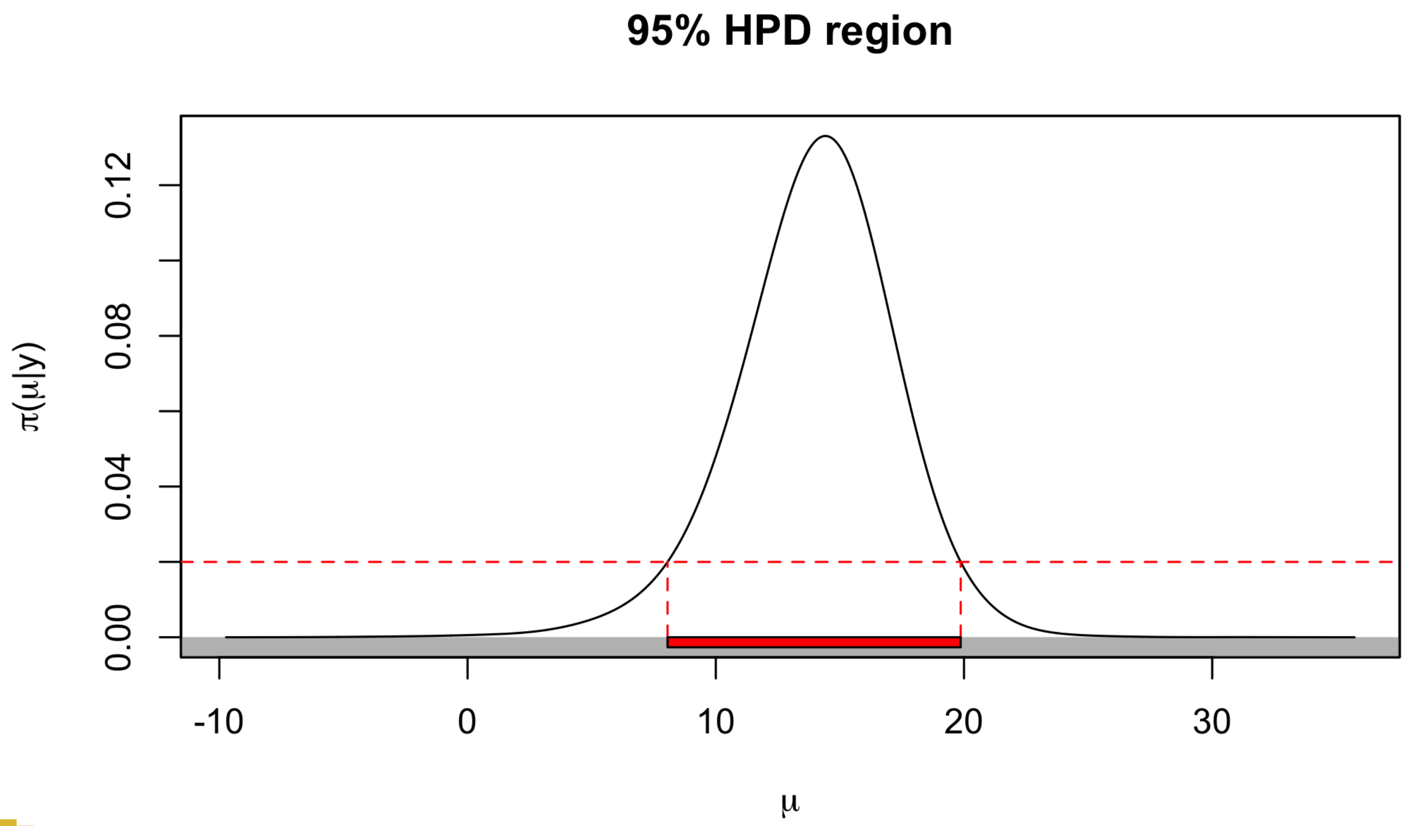
```
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##      mu     tau  sigma2
##  0.9521  2.0088 -1.9533
```

Practical advice on diagnostics

- There are more tests we can use: Raftery and Lewis diagnostic, Heidelberger and Welch, etc.
- The Gelman-Rubin approach is quite appealing in using multiple chains
- Geweke (and Heidelberger and Welch) sometimes reject even when the trace plots look good.
- Overly sensitive to minor departures from stationarity that do not impact inferences.
- Sometimes this can be solved with more iterations. Otherwise, you may want to try multiple chains.
- Most common method of assessing convergence is visual examination of trace plots.
- **CAUTION**: diagnostics cannot guarantee that a chain has converged, but they can indicate it has not converged.

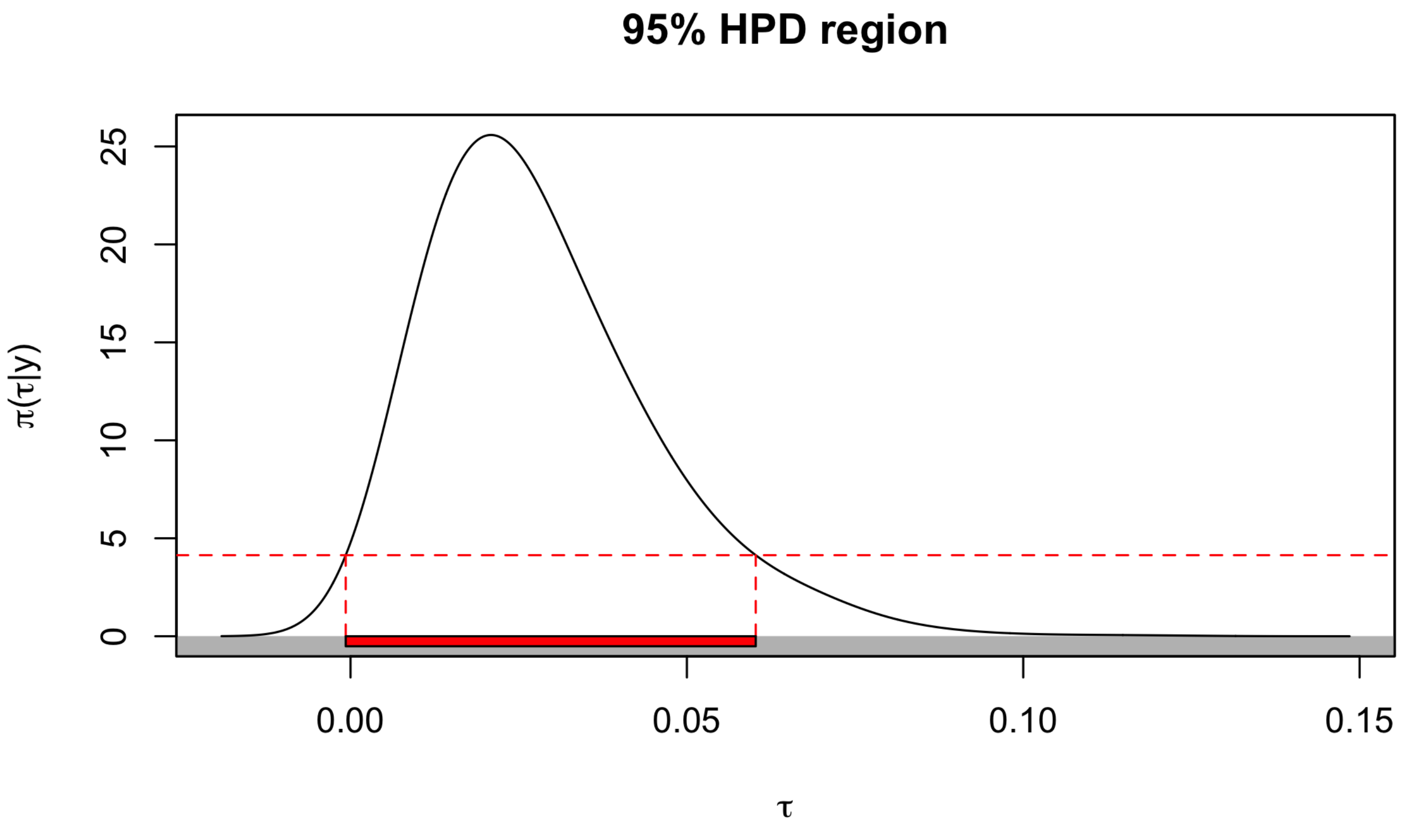
HPD interval for Pygmalion data

```r
#library(hdrcde)
hdr.den(PHI[,1],prob=95,main="95% HPD region", xlab=expression(mu),
        ylab=expression(paste(pi,"(", mu, "|y)")))
```



HPD INTERVAL FOR PYGMALION DATA

```
hdr.den(PHI[,2],prob=95,main="95% HPD region", xlab=expression(tau),
        ylab=expression(paste(pi,"(", tau, "|y)")))
```



HPD interval for Pygmalion data

```
hdr(PHI[,1],prob=95)$hdr
```

```
##           [,1]     [,2]
## 95% 8.080022 19.87699
```

```
hdr(PHI[,2],prob=95)$hdr
```

```
##                [,1]       [,2]
## 95% -0.0006954123 0.06023567
```

We can compare the HPD intervals to the equal tailed credible intervals.

```
quantile(PHI[,1],c(0.025,0.975))
```

```
##      2.5%     97.5%
##  7.519819 19.277013
```

```
quantile(PHI[,2],c(0.025,0.975))
```

```
##        2.5%       97.5%
## 0.005743552 0.068858238
```

Intervals are closer for μ (symmetric density) compared to τ (not symmetric).