A Guide to Bayesian Modeling

Author: Belias Michael

Student in MSc in Biostatistics

Athens, 20 April 2016

Bayesian Analysis

Michael Belias

April 20, 2017

Contents

1 Introduction

According to the Oxford dictionary, statistics is a branch of mathematics dealing with the collection, analysis, interpretation, presentation, and organization of data. Data may be of any applied science field such as medical, finance, social, physics etc and they can be separated into 2 types quantitative and qualitative.

The typical steps of a statistical analysis are seven in general :

1) Define the problem
2) Data collection and manipulation
3) Explore the data
4) Using the above three decide the model that will be used
5) Fit the model
6) Check the model and develop if necessary
7) Make the final model and infere

The above step are not distinct and in some cases there are overlaps and more steps nested. The same principles can be applied in the Bayesian Framework too.

In this tutorial we will learn:

- The bayesian intuition
- Fit the bayesian methods in simple popular statistical approaches such as:
  - (M)ANOVA
  - Linear Regression
  - Poisson regression
- Multi-lelel modeling
  - Hierarchical models
  - Meta analysis

2 Common probability distributions

2.1 Discrete distributions

2.1.1 Uniform

The uniform distribution is the simplest discrete probability distribution. It assigns equal probability to N different outcomes, usually represented with numbers 1,2,. . . .,N .

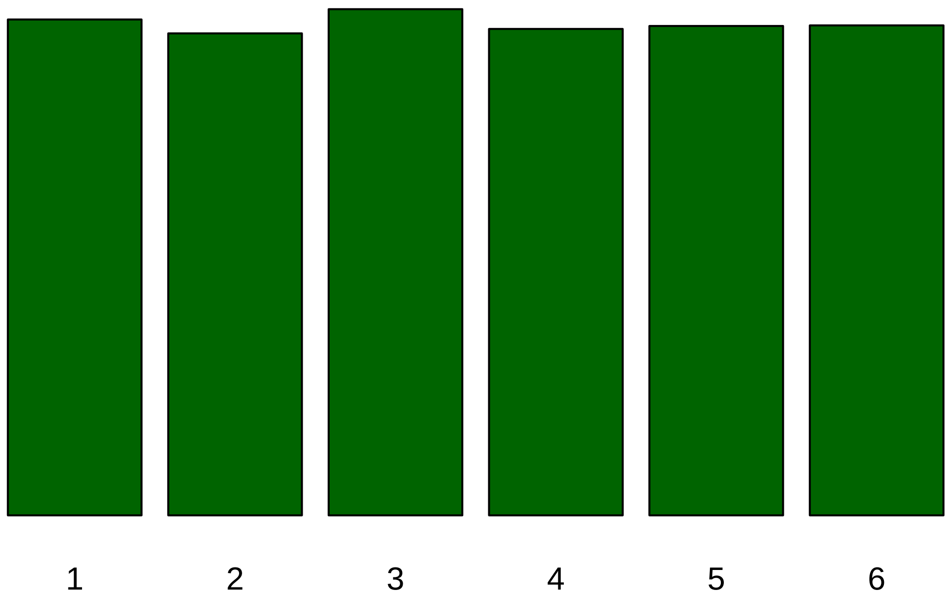
X ~ Uniform(N)

$P(X = x|N) = 1/N$ for x = 1,2,. . . N

$E[X] = \frac{N+1}{2}$

$Var[X] = \frac{N^2+1}{12}$

One common example is the outcome of throwing a fair six-sided die where N=6.



2.1.2 Bernoulli

The Bernoulli distribution is used for binary outcomes, such as 0 and 1. It has one parameter p, which is the probability of "success" frequently geting 1 (or any value we set). X ~ Bern(p)

$P(X = x|p) = p^x(1-p)^{1-x}$ for x = 0, 1

E[X] = p

Var[X] = p(1-p)

One common example is the outcome of flipping a fair coin (p = 0.5)

2.1.3 Binomial

The binomial distribution counts the number of “successes” in n independent Bernoulli trials (each with the same probability of success). Thus if $X_1, X_2, ..., X_n$ are independent Bernoulli(p) random variables, then Y = $\sum_{i=1}^{n} X_i$ is binomial distributed.

Y ~Binom(n, p)

P(Y= y|n,p) = $\binom{n}{y} p^y (1-p)^{(n-y)}$, for y = 0,1, . . . , n

E[Y] =np

Var[Y]= np(1-p)

where $\binom{n}{y} = \frac{n!}{y!(n-y)!}$.

2.1.4 Poisson

The Poisson distribution is used for counts, and arises in a variety of situations. The parameter $\lambda > 0$ is the rate at which we expect to observe the thing we are counting.

X~Pois(λ)

$P(X = x|\lambda) = \frac{\lambda^x exp(-\lambda)}{x!}$

E[X] = λ

Var[X] = λ

A Poisson process is a process wherein events occur on average at rate λ, events occur one at a time, and events occur independently of each other.

Example:

Significant earthquakes occur in the Western United States approximately following a Poisson process with rate of two earthquakes per week. What is the probability there will be at least 3 earthquakes in the next two weeks? Answer: the rate per two weeks is 2*2 = 4, so let X ~Pois(4) and we want to know $P(X \geq 3) = 1 - (X \leq 2) = 1 - P(X = 0) - P(X = 1) - P(X = 2) = 1 - e_4^{-4} e^{-4} - \frac{4^2 e^{-4}}{2} = 1 - 13e^{-4} = 0.762$

Note that 0! = 1 by definition.

2.1.5 Geometric

The geometric distribution is the number of failures before obtaining the first success, i.e., the number of Bernoulli failures until a success is observed, such as the first head when flipping a coin. It takes values on the positive integers starting with 0 (alternatively, we could count total trials until first success, in which case we would start with 1).

X~Geo(p)

$P(X = x|p) = p(1 - p)^x$, for x=1,2,...

$E[X] = \frac{1-p}{p}$

If the probability of getting a success is p, then the expected number of trials until the first success is 1=p and the expected number of failures until the first success is (1 - p)=p.

2.1.6 Negative Binomial

The negative binomial distribution extends the geometric distribution to model the number of failures before achieving the rth success. It takes values on the positive integers starting with 0.

Y~NegBinom(r,p)

P(Y= y|n,p) = $\binom{r+y-1}{y} p^r (1-p)^{(y)}$ for y=1,2,...

E[Y] = $\frac{r(1-p)}{p}$

Var[Y] = $\frac{r(1-p)}{p^2}$

Note that the geometric distribution is a special case of the negative binomial distribution where r = 1. Because 0 < p < 1, we have E[Y] < Var[Y]. This makes the negative binomial a popular alternative to the Poisson when modeling counts with high variance (recall, that the mean equals the variance for Poisson distributed variables).

2.1.7 Multinomial

Another generalization of the Bernoulli and the binomial is the multinomial distribution, which is like a binomial when there are more than two possible outcomes. Suppose we have n trials and there are k different possible outcomes which occur with probabilities $p_1, p_2, ...p_k$. For example, we are rolling a six-sided die that might be loaded so that the sides are not equally likely, then n is the total number of rolls, k = 6, p1 is the probability of rolling a one, and we denote by $x_1, x_2, ..., x_6$ a possible outcome for the number of times we observe rolls of each of one through six, where $\sum_{i=1}^{6} x_i = n$ and $\sum_{i=1}^{6} p_i = 1$

3 Bayesian Theory

3.1 History

Bayesian statistics are based on the homonymous Bayes' theorem or rule, invented by Thomas Bayes, which was a british reverend the 1740s . His primary field of studying was theology but Bayes was also "amateur" mathematician. He was influenced by David Hume a philosopher teacher while his studies in Edinburgh proposing that we can only rely on what we learn from experience. The probabilities as a mathematical field these days where just emerging being able to solve simple problems like *what is the probability of observing an effect given a cause?* but not the inverse P(cause | effect). Bayes gave a simple example of tossing balls on a table and recording where they stop (to the left or to the right side of the table), noting that the more balls are thrown, the better we may infere if the ball-tosing is bias to a side. This is nowadays called a learning process and although it was a remarkable finding Bayes forgot it in a drawer (!) until his death.Richard Price found it and after studing his papers for 2 years and making some corrections he finally published **An Essay toward solving a Problem in the Doctrine of Chances". 1763**.

Still the theorem was just an example not having the final form of today and even after this publication no-one really continued the development except of Laplace, who was trying to solve an astronomical problem , studied Price's paper developed a first version of what we now call Bayes theorem. The reception of Laplace's proposal was slightly hostile due to the inherent challenges such as the equal prior probabilities, being subjective and the serious technical computational problems in practice, which is still a great issue .

3.2 Bayes theorem

Bayes theorem is calculating the probability event given prior knowledge of conditions that might be related to the event. Bayes' theorem is stated mathematically as the following equation (Efron, 2013) :

$P(A \mid B) = \frac{P(B|A)\, P(A)}{P(B)}$

$P(A \mid B) = \frac{P(B|A)\, P(A)}{P(B)}$

where A and B are events and P(B) $\neq$ 0.

- P(A) and P(B) are the probabilities of observing A and B without regard to each other.
- P(A ∣ B), a conditional probability, is the probability of observing event A given that B is true.
- P(B ∣ A) is the probability of observing event B given that A is true.

This is the basis of Bayesian inference which is a particular approach to statistical inference, with it's own interpretation and When applied, the probabilities involved in Bayes' theorem may have different probability interpretations. With the Bayesian probability interpretation the theorem expresses how a subjective degree of belief should rationally change to account for availability of related evidence. Bayesian inference is fundamental to Bayesian statistics.

4 Monte Carlo estimation

4.1 Simulation and CLT

Before we learn how to simulate from complicated posterior distributions, let's review some of the basics of Monte Carlo estimation. Monte Carlo estimation refers to simulating hypothetical draws from a probability distribution in order to calculate important quantities. These quantities might include the mean, the variance, the probability of some event, or quantiles of the distribution. All of these calculations involve integration, which except for the simplest distributions, can be very difficult or impossible.

Suppose we have a random variable $\hat{\theta}$ that follows a Gamma(a,b). Let's say a=2 and b=1/3 , where b is the rate parameter. To calculate the mean of this distribution, we would need to compute the following integral

$$\mathrm{E}(\theta) = \int_0^\infty \theta f(\theta) d\theta = \int_0^\infty \theta \frac{b^a}{\Gamma(a)} \theta^{a-1} e^{-b\theta} d\theta$$

It is possible to compute this integral, and the answer is a/b (6 in this case). However, we could verify this answer through Monte Carlo estimation. To do so, we would simulate a large number of draws (call them θ_i^* for i = 1,2,. . . ,m) from this gamma distribution and calculate their sample mean. Why can we do this? Recall from the previous course that if we have a random sample from a distribution, the average of those samples converges in probability to the true mean of that distribution by the Law of Large Numbers. Furthermore, by the Central Limit Theorem (CLT), this sample mean $\bar{\theta}^* = \frac{1}{m} \sum_{i=1}^m \theta_i^*$ approximately follows a normal distribution with mean E(θ) and variance Var(θ)/m. The theoretical variance of θ is the following integral:

$$\mathrm{Var}(\theta) = \int_0^\infty (\theta - E(\theta))^2 f(\theta) d\theta$$

Just like we did with the mean, we could approximate this variance with the sample variance $\frac{1}{m} \sum_{i=1}^m (\theta_i^* - \bar{\theta}^*)^2$

4.2 Calculating probabilities

This method can be used to calculate many different integrals. Say h(θ) is any function and we want to calculate $\int h(\theta)p(\theta)d\theta$ This integral is precisely what is meant by E(h(θ)), so we can conveniently approximate it by taking the sample mean of h(θ_i^*). That is, we apply the function hh to each simulated sample from the distribution, and take the average of all the results.

One extremely useful example of an hh function is is the indicator $I_A(\theta)$ where A is some logical condition about the value of θ. To demonstrate, suppose $h(\theta) = I_{[\theta<5]}(\theta)$, which will give a 1 if $\theta < 5$ and 0 otherwise.

The E(h(θ)) = $\int_0^\infty I_{[\theta<5]}(\theta)p(\theta)d\theta = \int_0^5 1 \cdot p(\theta)d\theta + \int_5^\infty 0 \cdot p(\theta)d\theta = P(\theta < 5)$. This means we can approximate the probability that $\theta < 5$ by drawing many samples θ_i^* , and approximating this integral with $\frac{1}{m}\sum_{i=1}^m I_{\theta^*<5}(\theta_i^*)$. This expression is simply counting how many of those samples come out to be less than 55, and dividing by the total number of simulated samples. So simple!

Likewise, we can approximate quantiles of a distribution. If we are looking for the value ζ such that P(θ < z) = 0.9 , we simply arrange the samples θ_i^* in ascending order and find the smallest drawn value that is greater than 90% of the others.

4.3 Monte Carlo error

How good is an approximation by Monte Carlo sampling? Again we can turn to the CLT, which tells us that the variance of our estimate is controlled in part by m . For a better estimate, we want larger m .

For example, if we seek E(θ), then the sample mean $\bar{\theta}^*$ approximately follows a normal distribution with mean E(θ) and variance Var(θ)/m. The variance tells us how far our estimate might be from the true value. One way to approximate Var(θ) is to replace it with the sample variance. The standard deviation of our Monte Carlo estimate is the square root of that, or the sample standard deviation divided by $\sqrt{m}$.

If m is large, it is reasonable to assume that the true value will likely be within about two standard deviations of your Monte Carlo estimate.

4.4 Marginalization

We can also obtain Monte Carlo samples from hierarchical models. As a simple example, let's consider a binomial random variable where $y \mid \phi \sim \text{Bin}(10, \phi)$, and further suppose ϕ is random (as if it had a prior) and is distributed beta $\phi\ Beta(2, 2)$. Given any hierarchical model, we can always write the joint distribution of y and ϕ as $p(y, \phi) = p(y \mid \phi)p(\phi)$ using the chain rule of probability. To simulate from this joint distribution, repeat these steps for a large number m :

- Simulate ϕ_i^* rom its Beta(2,2)distribution
- Given the drawn ϕ_i^* ,simulate y_i^* from $\text{Bin}(10, \phi_i^*)$

This will produce m independent pairs of $(y^*, \phi^*)_i$ drawn from their joint distribution. One major advantage of Monte Carlo simulation is that marginalizing is easy. Calculating the marginal distribution of y, $p(y) = \int_0^1 p(y, \phi)d\phi$ might be challenging. But if we have draws from the joint distribution, we can just discard the ϕ_i^* raws and use the y_i^*as samples from their marginal distribution. This is also called the prior predictive distribution introduced in the previous course.

In the next segment, we will demonstrate some of these principles. Remember, we do not yet know how to sample from the complicated posterior distributions introduced in the previous lesson. But once we learn that, we will be able to use the principles from this lesson to make approximate inferences from those posterior distributions.

5 Markov chains

Definition If we have a sequence of random variables $X_1, X_2, ..., X_n$ where the indices *1,2,…,n* represent successive points in time, we can use the chain rule of probability to calculate the probability of the entire sequence:

$p(X_{t+1}|X_t, X_{t-1}, \ldots, X_2, X_1) = p(X_{t+1}|X_t)$

Markov chains simplify this expression by using the *Markov assumption*. The assumption is that given the entire past history, the probability distribution for the random variable at the next time step only depends on the current variable. Mathematically, the assumption is written like this:

$p(X_{t+1}|X_t, X_{t-1}, \ldots, X_2, X_1) = p(X_{t+1}|X_t)$

for all *t=2,…,n*. Under this assumption, we can write the first expression as $p(X_1, X_2, \ldots X_n) = p(X_1) \cdot p(X_2|X_1) \cdot p(X_3|X_2) \cdot p(X_4|X_3) \cdot \ldots \cdot p(X_n|X_{n-1})$,

which is much simpler than the original. It consists of an initial distribution for the first variable, $P(X_1)$, and **n - 1** transition probabilities. We usually make one more assumption: that the transition probabilities do not change with time. Hence, the transition from time tt to time **t+1** depends only on the value of X_t.

5.1 Examples of Markov chains

5.1.1 Discrete Markov chain

Suppose you have a secret number (make it an integer) between 1 and 5. We will call it your initial number at *step 1*. Now for each time step, your secret number will change according to the following rules:

1. Flip a coin.
2.

- If the coin turns up heads, then increase your secret number by one (5 increases to 1).
- If the coin turns up tails, then decrease your secret number by one (1 decreases to 5).

3. Repeat *n** times, and record the evolving history of your secret number.

Before the experiment, we can think of the sequence of secret numbers as a sequence of random variables, each taking on a value in **{1,2,3,4,5}**. Assume that the coin is fair, so that with each flip, the probability of heads and tails are both 0.5.

Does this game qualify as a true Markov chain? Suppose your secret number is currently 4 and that the history of your secret numbers is **(2,1,2,3)**. What is the probability that on the next step, your secret number will be 5? What about the other four possibilities? Because of the rules of this game, the probability of the next transition will depend only on the fact that your current number is 4. The numbers further back in your history are irrelevant, so this is a Markov chain.

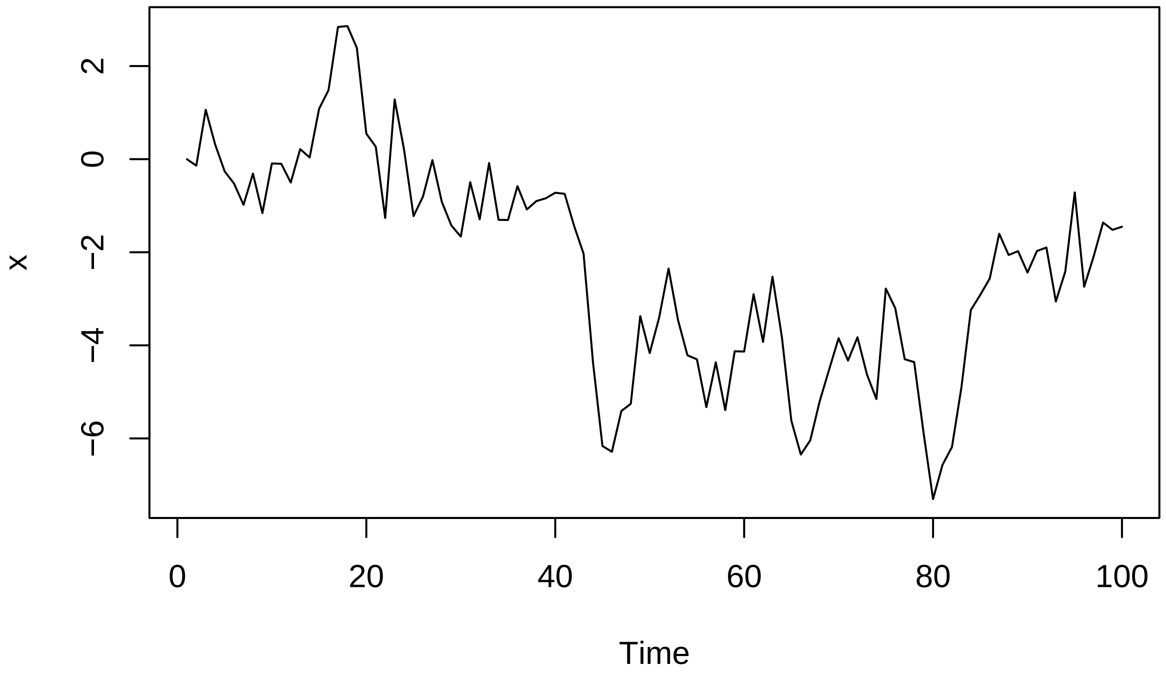
This is an example of a discrete Markov chain, where the possible values of the random variables come from a discrete set. Those possible values (secret numbers in this example) are called states of the chain. The states are usually numbers, as in this example, but they can represent anything. In one common example, the states describe the weather on a particular day, which could be labeled as 1-fair, 2-poor.

5.1.2 Random walk (continuous)

Now let's look at a continuous example of a Markov chain. Say Xt=0Xt=0 and we have the following transition model: $p(X_{t+1}|X_t = x_t) = \mathrm{N}(x_t, 1),$. That is, the probability distribution for the next state is Normal with variance 1 and mean equal to the current state. This is often referred to as a "random walk." Clearly, it is a Markov chain because the transition to the next state X_{t+1}only depends on the current state X_t.

R-code

```
set.seed(34)

n = 100
x = numeric(n)

for (i in 2:n) {
  x[i] = rnorm(1, mean=x[i-1], sd=1.0)
}

plot.ts(x)
```



5.1.3 Transition matrix

Let's return to our example of the discrete Markov chain. If we assume that transition probabilities do not change with time, then there are a total of $5^2 = 25$ potential transition probabilities. Potential transition probabilities would be from *State 1* to *State 2*, *State 1* to *State 3*, and so forth. These transition probabilities can be arranged into a matrix Q:

$$Q = \begin{pmatrix} 0 & .5 & 0 & 0 & .5 \\ .5 & 0 & .5 & 0 & 0 \\ 0 & .5 & 0 & .5 & 0 \\ 0 & 0 & .5 & 0 & .5 \\ .5 & 0 & 0 & .5 & 0 \end{pmatrix}$$

where the transitions from *State 1*are in the first row, the transitions from *State 2* are in the second row, etc. For example, the probability $p(X_{t+1} = 5 \mid X_t = 4)$ can be found in the fourth row, fifth column.

The transition matrix is especially useful if we want to find the probabilities associated with multiple steps of the chain. For example, we might want to know $p(X_{t+2} = 3 \mid X_t = 1)$, the probability of your secret number being 3 two steps from now, given that your number is currently 1. We can calculate this as $\sum_{k=1}^{5} p(X_{t+2} = 3 \mid X_{t+1} = k) \cdot p(X_{t+1} = k \mid X_t = 1)$, which conveniently is found in the first row and third column of Q^2.

R-code

```
Q = matrix(c(0.0, 0.5, 0.0, 0.0, 0.5,
                0.5, 0.0, 0.5, 0.0, 0.0,
                0.0, 0.5, 0.0, 0.5, 0.0,
                0.0, 0.0, 0.5, 0.0, 0.5,
                0.5, 0.0, 0.0, 0.5, 0.0),
             nrow=5, byrow=TRUE)


Q %*% Q # Matrix multiplication in R. This is Q^2.
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.50 0.00 0.25 0.25 0.00
## [2,] 0.00 0.50 0.00 0.25 0.25
## [3,] 0.25 0.00 0.50 0.00 0.25
## [4,] 0.25 0.25 0.00 0.50 0.00
## [5,] 0.00 0.25 0.25 0.00 0.50
```

5.1.4 Stationary distribution

Suppose we want to know the probability distribution of the your secret number in the distant future, say $p(X_{t+h}|X_t)$ where h is a large number. Let's calculate this for a few different values of h.

```
Q5 = Q %*% Q %*% Q %*% Q %*% Q # h=5 steps in the future
round(Q5, 3)
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0.062 0.312 0.156 0.156 0.312
## [2,] 0.312 0.062 0.312 0.156 0.156
## [3,] 0.156 0.312 0.062 0.312 0.156
## [4,] 0.156 0.156 0.312 0.062 0.312
## [5,] 0.312 0.156 0.156 0.312 0.062
```

```
Q10 = Q %*% Q %*% Q %*% Q %*% Q %*% Q %*% Q %*% Q %*% Q %*% Q # h=10 steps
round(Q10, 3)
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0.248 0.161 0.215 0.215 0.161
## [2,] 0.161 0.248 0.161 0.215 0.215
## [3,] 0.215 0.161 0.248 0.161 0.215
## [4,] 0.215 0.215 0.161 0.248 0.161
## [5,] 0.161 0.215 0.215 0.161 0.248
```

```
Q30 = Q
for (i in 2:30) {
  Q30 = Q30 %*% Q
}
round(Q30, 3) # h=30 steps in the future
```

```
##        [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 0.201 0.199 0.200 0.200 0.199
## [2,] 0.199 0.201 0.199 0.200 0.200
## [3,] 0.200 0.199 0.201 0.199 0.200
## [4,] 0.200 0.200 0.199 0.201 0.199
## [5,] 0.199 0.200 0.200 0.199 0.201
```

Notice that as the future horizon gets more distant, the transition distributions appear to converge. The state you are currently in becomes less important in determining the more distant future. If we let hh get really large, and take it to the limit, all the rows of the long-range transition matrix will become equal to *(.2,.2,.2,.2,.2)*. That is, if you run the Markov chain for a very long time, the probability that you will end up in any particular state is *1/5=.2* for each of the five states. These long-range probabilities are equal to what is called the stationary distribution of the Markov chain.

The stationary distribution of a chain is the initial state distribution for which performing a transition will not change the probability of ending up in any given state. That is,

```
c(0.2, 0.2, 0.2, 0.2, 0.2) %*% Q
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  0.2  0.2  0.2  0.2  0.2
```

One consequence of this property is that once a chain reaches its stationary distribution, the stationary distribution will remain the distribution of the states thereafter.

We can also demonstrate the stationary distribution by simulating a long chain from this example.

```
n = 5000
```

```
x = numeric(n)
x[1] = 1 # fix the state as 1 for time 1
for (i in 2:n) {
  x[i] = sample.int(5, size=1, prob=Q[x[i-1],]) # draw the next state from
}
```

Now that we have simulated the chain, let's look at the distribution of visits to the five states.

```
table(x) / n
```

```
## x
##      1      2      3      4      5
## 0.1996 0.2020 0.1980 0.1994 0.2010
```

The overall distribution of the visits to the states is approximately equal to the stationary distribution.

As we have just seen, if you simulate a Markov chain for many iterations, the samples can be used as a Monte Carlo sample from the stationary distribution. This is exactly how we are going to use Markov chains for Bayesian inference. In order to simulate from a complicated posterior distribution, we will set up and run a Markov chain whose stationary distribution is the posterior distribution.

It is important to note that the stationary distribution doesn't always exist for any given Markov chain. The Markov chain must have certain properties, which we won't discuss here. However, the Markov chain algorithms we'll use in future lessons for Monte Carlo estimation are guaranteed to produce stationary distributions.

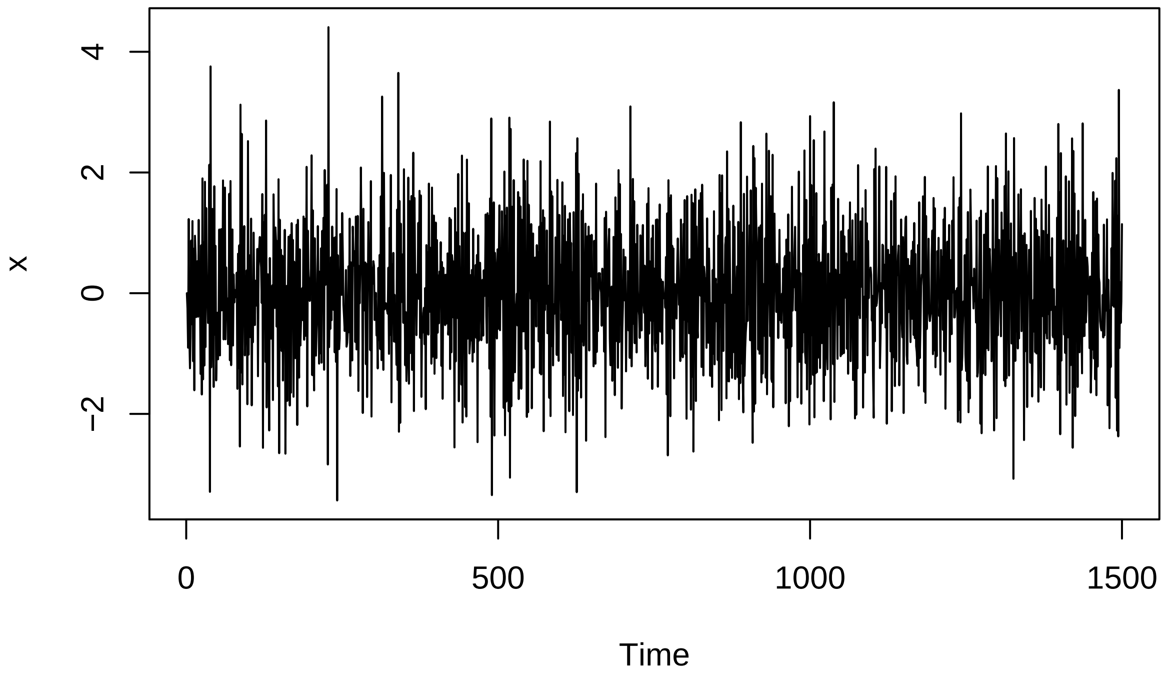
5.1.5 Continuous example

The continuous random walk example we gave earlier does not have a stationary distribution. However, we can modify it so that it does have a stationary distribution.

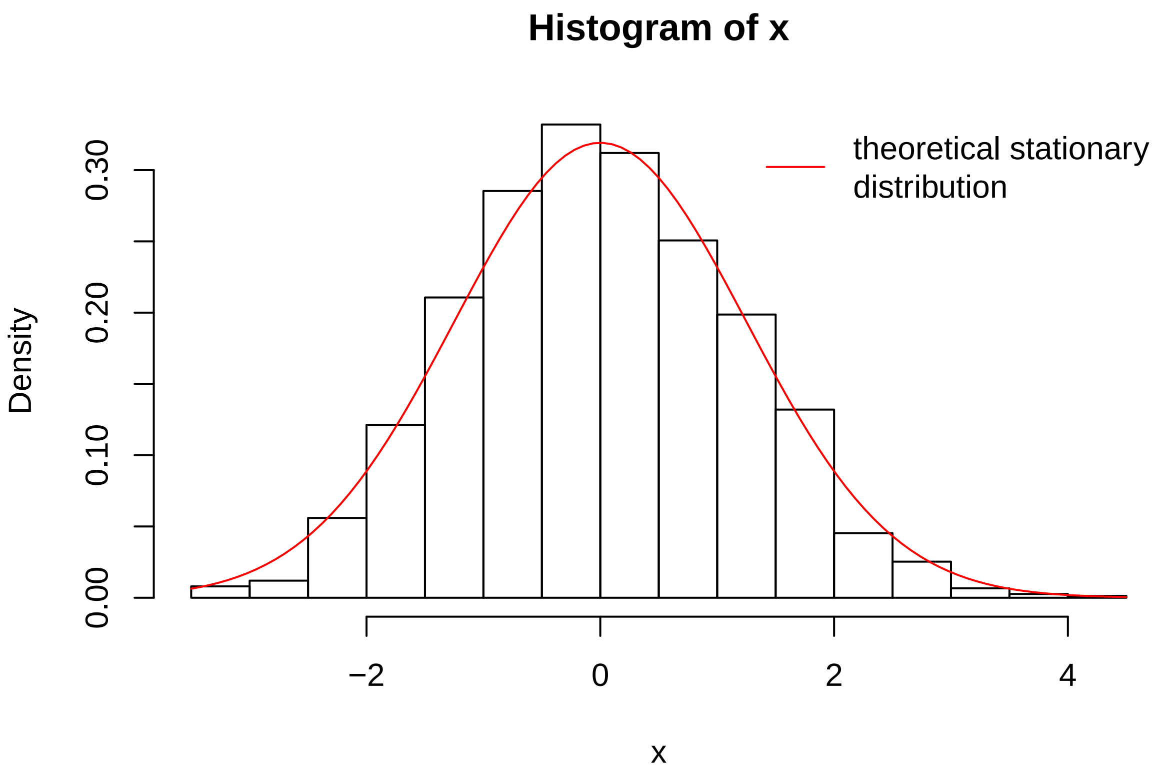
Let the transition distribution be $p(X_{t+1}|X_t = x_t) = \mathrm{N}(\phi x_t, 1)$ where *-1<ϕ<1* .That is, the probability distribution for the next state is Normal with variance 1 and mean equal to ϕ times the current state. As long as ϕ is between *-1* and* 1*, then the stationary distribution will exist for this model.

Let's simulate this chain for $\phi = -0.6$.

```
set.seed(38)

n = 1500
x = numeric(n)
phi = -0.6

for (i in 2:n) {
  x[i] = rnorm(1, mean=phi*x[i-1], sd=1.0)
}

plot.ts(x)
```



The theoretical stationary distribution for this chain is normal with mean 0 and variance 1/(1-ϕ^2) which in our example approximately equals *1.562*. Let's look at a histogram of our chain and compare that with the theoretical stationary distribution.



It appears that the chain has reached the stationary distribution. Therefore, we could treat this simulation from the chain like a Monte Carlo sample from the stationary distribution, a normal with mean *0* and variance *1.562*.

Because most posterior distributions we will look at are continuous, our Monte Carlo simulations with Markov chains will be similar to this example.

5.2 Monte Carlo Example

6 Metropolis-Hastings

Metropolis-Hastings is an algorithm that allows us to sample from a generic probability distribution (which we will call the target distribution), even if we do not know the normalizing constant. To do this, we construct and sample from a Markov chain whose stationary distribution is the target distribution. It consists of picking an arbitrary starting value, and iteratively accepting or rejecting candidate samples drawn from another distribution, one that is easy to sample.

Let's say we wish to produce samples from a target distribution $p(\theta) \propto g(\theta)$ where we don't know the normalizing constant (since $\int g(\theta) d\theta$ is hard or impossible to compute), so we only have $g(\theta)$ to work with. The Metropolis-Hastings algorithm proceeds as follows.

1. Select an initial value θ_0 .
2. For i=1,. . . ,m repeat the following steps:

- Draw a candidate sample θ^* from a proposal distribution $q(\theta^* \mid \theta_{i-1})$ (more on this later). *Compute the ratio $\alpha = \frac{g(\theta^*)/q(\theta^*|\theta_{i-1})}{g(\theta_{i-1})/q(\theta_{i-1}|\theta^*)} = \frac{g(\theta^*)q(\theta_{i-1}|\theta^*)}{g(\theta_{i-1})q(\theta^*|\theta_{i-1})}$.

If $\alpha \geq 1$, then set $\theta_i = \theta^*$ If $\alpha < 1$, then set $\theta_i = \theta^*$ with probability α, or $\theta_i = \theta_{i-1}$ with probability 1-α.

Steps 2b and 2c act as a correction since the proposal distribution is not the target distribution. At each step in the chain, we draw a candidate and decide whether to "move" the chain there or remain where we are. If the proposed move to the candidate is "advantageous," ($\alpha \geq 1$) we "move" there and if it is not "advantageous," we still might move there, but only with probability α. Since our decision to "move" to the candidate only depends on where the chain currently is, this is a Markov chain.

6.1 Proposal distribution

One careful choice we must make is the candidate generating distribution$q(\theta^* \mid \theta_{i-1})$ It may or may not depend on the previous iteration's value of θ. One example where it doesn't depend on the previous value would be if $q(\theta)$is always the same distribution. If we use this option, $q(\theta)$ should be as similar as possible to $p(\theta)$.

Another popular option, one that does depend on the previous iteration, is Random-Walk Metropolis-Hastings. Here, the proposal distribution is centered on θ_{i-1}. For instance, it might be a normal distribution with mean θ_{i-1}. Because the normal distribution is symmetric, this example comes with another advantage: $q(\theta^* \mid \theta_{i-1}) = q(\theta_{i-1} \mid \theta^*)$, causing it to cancel out when we calculate α. Thus, in Random-Walk Metropolis-Hastings where the candidate is drawn from a normal with mean θ_{i-1} and constant variance, the acceptance ratio is $\alpha = \frac{g(\theta^*)}{g(\theta_{i-1})}$.

6.2 Acceptance rate

Clearly, not all candidate draws are accepted, so our Markov chain sometimes "stays" where it is, possibly for many iterations. How often you want the chain to accept candidates depends on the type of algorithm you use. If you approximate $p(\theta)$ with $q(\theta^*)$and always draw candidates from that, accepting candidates often is good; it means $q(\theta^*)$is approximating $p(\theta)$ well. However, you still may want qq to have a larger variance than pp and see some rejection of candidates as an assurance that qq is covering the space well.

As we will see in coming examples, a high acceptance rate for the Random-Walk Metropolis-Hastings sampler is not a good thing. If the random walk is taking too small of steps, it will accept often, but will take a very long time to fully explore the posterior. If the random walk is taking too large of steps, many of its proposals will have low probability and the acceptance rate will be low, wasting many draws. Ideally, a random walk sampler should accept somewhere between 23% and 50% of the candidates proposed.

In the next segment, we will see a demonstration of this algorithm used in a discrete case, where we can show mathematically that the Markov chain converges to the target distribution. In the following segment, we will demonstrate coding a Random-Walk Metropolis-Hastings algorithm in R to solve one of the problems from the end of Lesson 2.

6.3 Random walk with normal likelihood, t prior

Recall the model from the last segment of Lesson 2 where the data are the percent change in total personnel from last year to this year for *n=10* companies. We used a normal likelihood with known variance and *t* distribution for the prior on the unknown mean. Suppose the values are *y=(1.2,1.4,-0.5,0.3,0.9,2.3,1.0,0.1,1.3,1.9)* . Because this model is not conjugate, the posterior distribution is not in a standard form that we can easily sample. To obtain posterior samples, we will set up a Markov chain whose stationary distribution is this posterior distribution.

Recall that the posterior distribution is:

$$p(\mu \mid y_1, \ldots, y_n) \propto \frac{\exp[n(\bar{y}\mu - \mu^2/2)]}{1+\mu^2}$$

The posterior distribution on the left is our target distribution and the expression on the right is our g(μ).

The first thing we can do in R is write a function to evaluate g(μ). Because posterior distributions include likelihoods (the product of many numbers that are potentially small), g(μ) might evaluate to such a small number that to the computer, it effectively zero. This will cause a problem when we evaluate the acceptance ratio α. To avoid this problem, we can work on the log scale, which will be more numerically stable. Thus, we will write a function to evaluate

$$\log(g(\mu)) = n(\bar{y}\mu - \mu^2/2) - \log(1+\mu^2)$$

This function will require three arguments, μ , $\bar{y}$ and n.

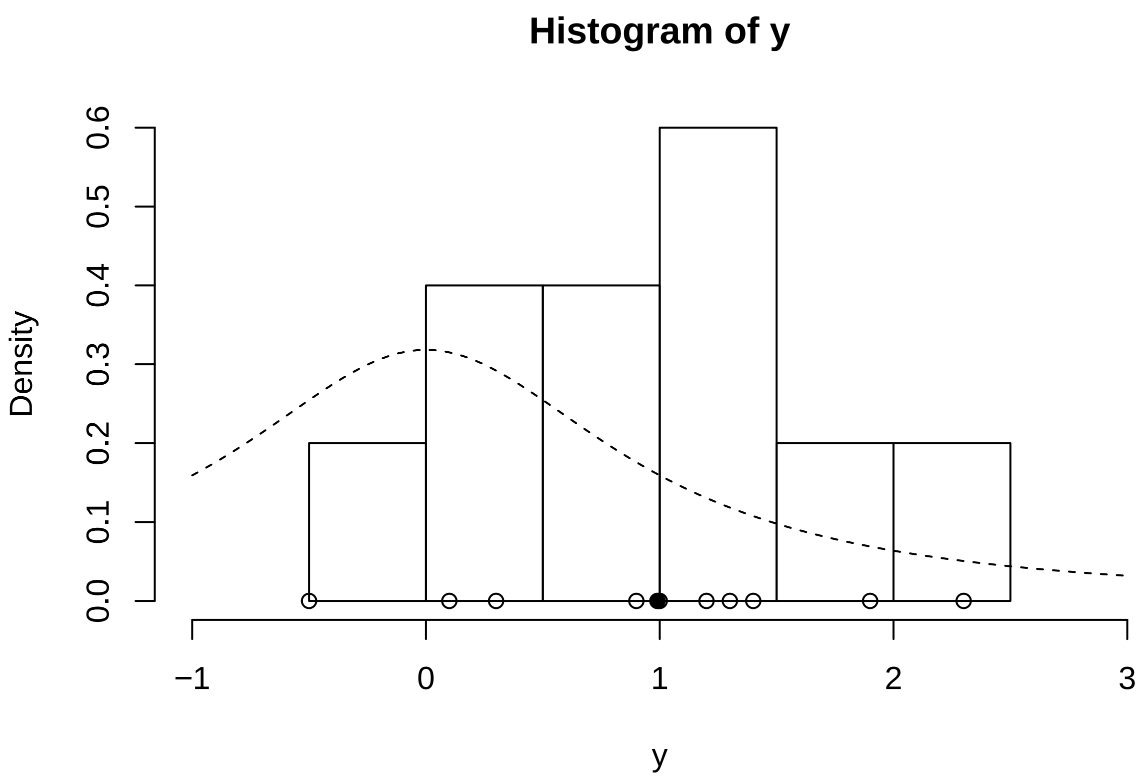
```
lg = function(mu, n, ybar) {
  mu2 = mu^2
  n * (ybar * mu - mu2 / 2.0) - log(1 + mu2)
}
```

Next, let's write a function to execute the Random-Walk Metropolis-Hastings sampler with normal proposals.

```
mh = function(n, ybar, n_iter, mu_init, cand_sd) {
  ## Random-Walk Metropolis-Hastings algorithm

  ## step 1, initialize
  mu_out = numeric(n_iter)
  accpt = 0
  mu_now = mu_init
  lg_now = lg(mu=mu_now, n=n, ybar=ybar)

  ## step 2, iterate
  for (i in 1:n_iter) {
    ## step 2a
    mu_cand = rnorm(n=1, mean=mu_now, sd=cand_sd) # draw a candidate

    ## step 2b
    lg_cand = lg(mu=mu_cand, n=n, ybar=ybar) # evaluate log of g with the c
    lalpha = lg_cand - lg_now # log of acceptance ratio
    alpha = exp(lalpha)
    ## step 2c
    u = runif(1) # draw a uniform variable which will be less than alpha u
    if (u < alpha) { # then accept the candidate
      mu_now = mu_cand
      accpt = accpt + 1 # to keep track of acceptance
      lg_now = lg_cand
    }

    ## collect results
    mu_out[i] = mu_now # save this iteration's value of mu
  }

  ## return a list of output
  list(mu=mu_out, accpt=accpt/n_iter)
}
```

Now, let's set up the problem.

```
y = c(1.2, 1.4, -0.5, 0.3, 0.9, 2.3, 1.0, 0.1, 1.3, 1.9)
ybar = mean(y)
n = length(y)
hist(y, freq=FALSE, xlim=c(-1.0, 3.0)) # histogram of the data
curve(dt(x=x, df=1), lty=2, add=TRUE) # prior for mu
points(y, rep(0,n), pch=1) # individual data points
points(ybar, 0, pch=19) # sample mean
```

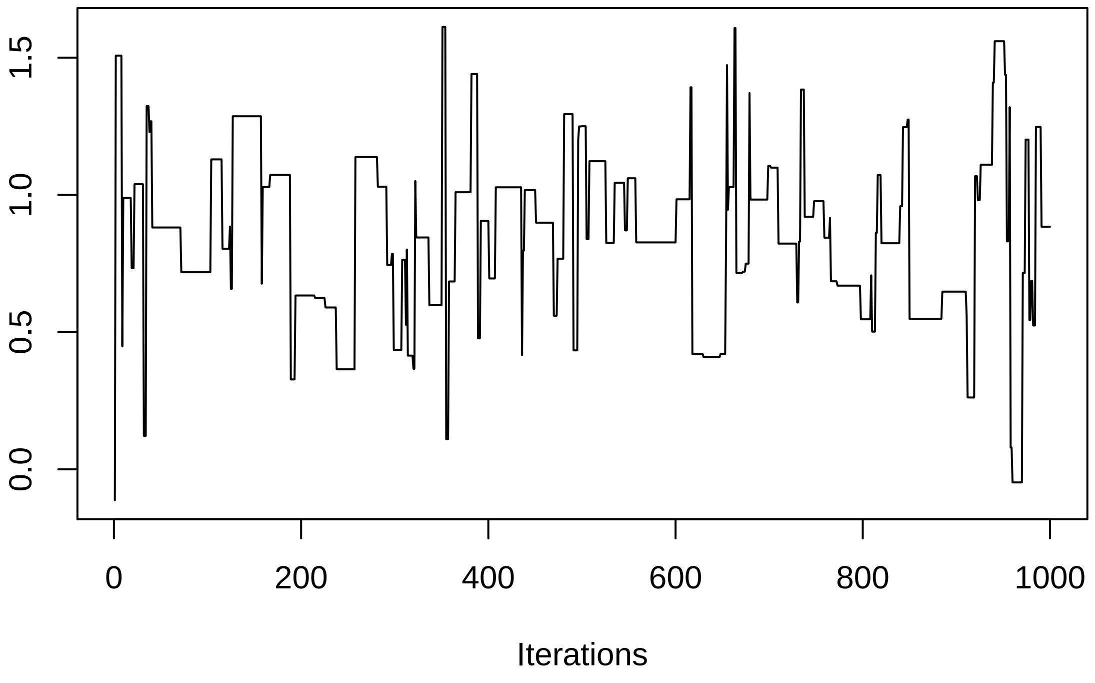


Finally, we're ready to run the sampler! Let's use *m=1000* iterations and proposal standard deviation (which controls the proposal step size) 3.0, and initial value at the prior median 0

```
set.seed(43) # set the random seed for reproducibility
post = mh(n=n, ybar=ybar, n_iter=1e3, mu_init=0.0, cand_sd=3.0)
str(post)
```

```
List of 2
 $ mu   : num [1:1000] -0.113 1.507 1.507 1.507 1.507 ...
 $ accpt: num 0.122
```

```
library("coda")
traceplot(as.mcmc(post$mu))
```



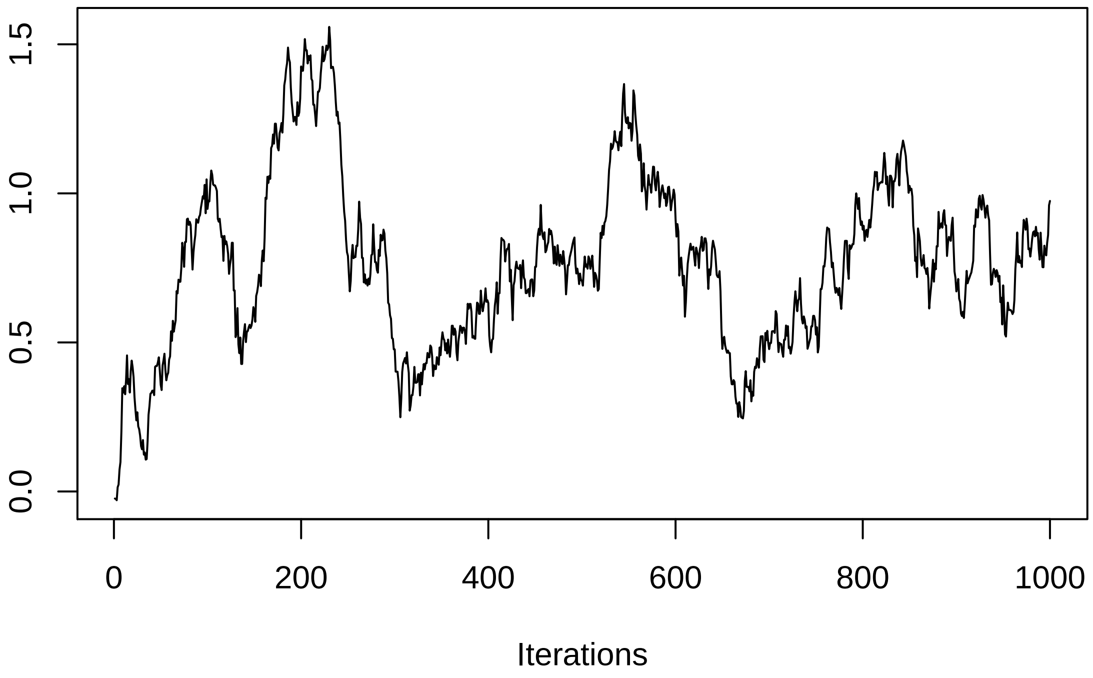
This last plot is called a trace plot. It shows the history of the chain and provides basic feedback about whether the chain has reached its stationary distribution.

It appears our proposal step size was too large (acceptance rate below 23%). Let's try another.

```
post = mh(n=n, ybar=ybar, n_iter=1e3, mu_init=0.0, cand_sd=0.05)
post$accpt
```

```
## [1] 0.946
```

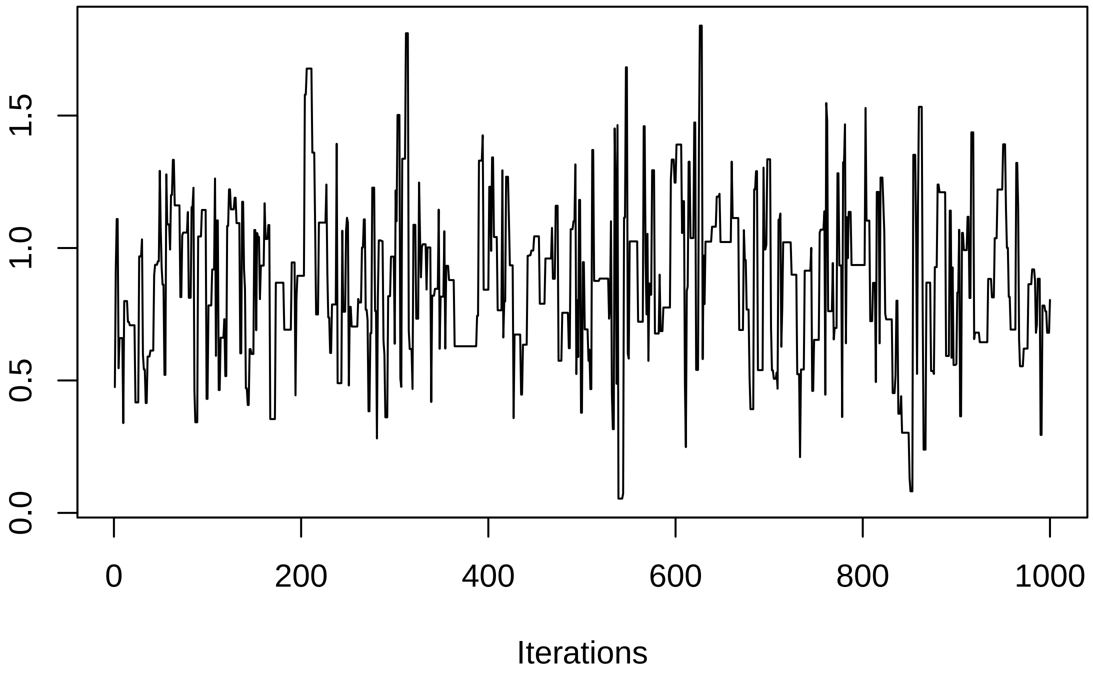
```
traceplot(as.mcmc(post$mu))
```



```
post = mh(n=n, ybar=ybar, n_iter=1e3, mu_init=0.0, cand_sd=0.9)
post$accpt
```

```
## [1] 0.38
```

```
traceplot(as.mcmc(post$mu))
```

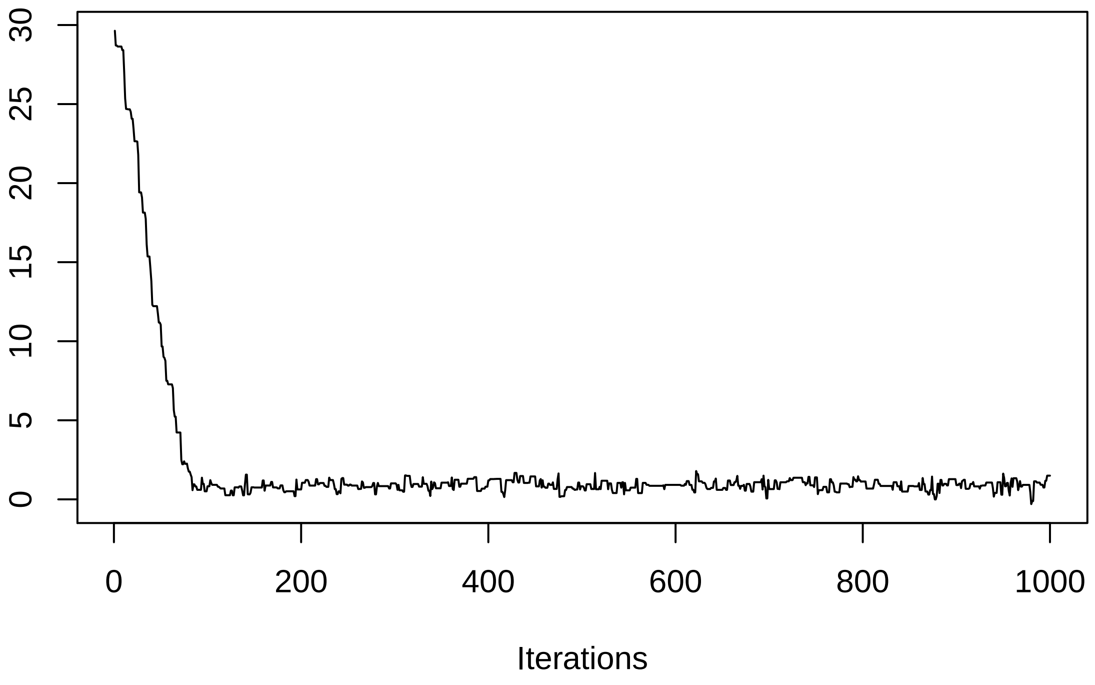


Hey, that looks pretty good. Just for fun, let's see what happens if we initialize the chain at some far-off value.

```
post = mh(n=n, ybar=ybar, n_iter=1e3, mu_init=30.0, cand_sd=0.9)
post$accpt
```

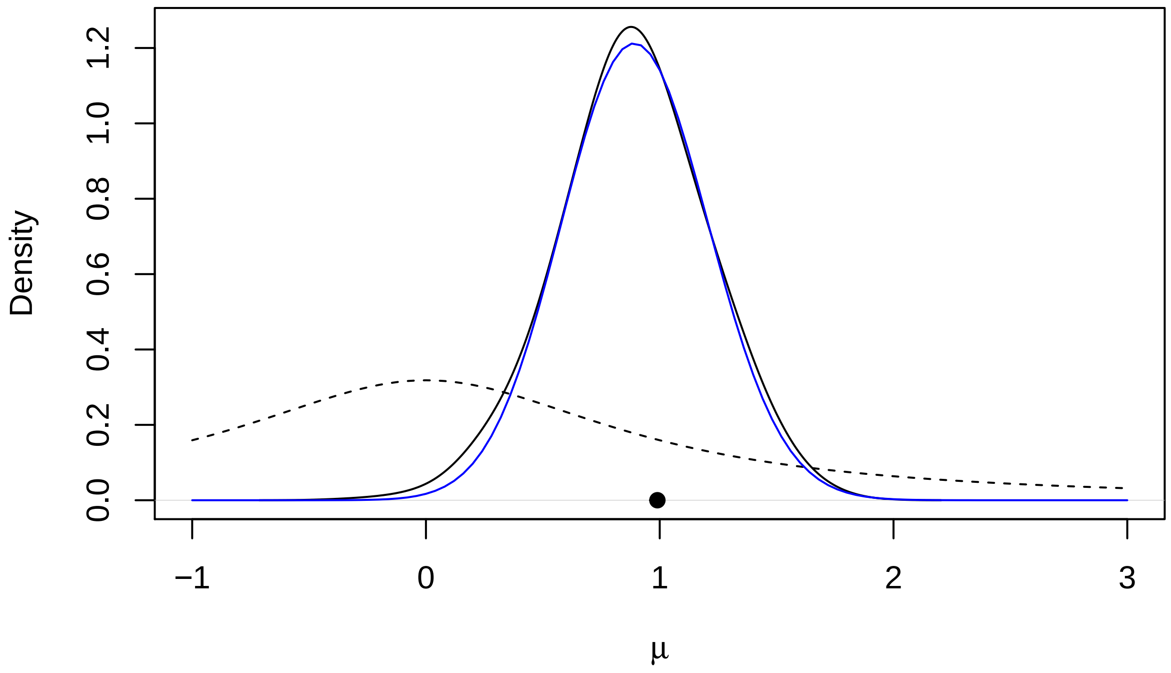
```
## [1] 0.387
```

```
traceplot(as.mcmc(post$mu))
```



It took awhile to find the stationary distribution, but it looks like we succeeded! If we discard the first 100 or so values, it appears like the rest of the samples come from the stationary distribution, our posterior distribution! Let's plot the posterior density against the prior to see how the data updated our belief about μ.

```
post$mu_keep = post$mu[-c(1:100)] # discard the first 200 samples
plot(density(post$mu_keep, adjust=2.0), main="", xlim=c(-1.0, 3.0), xlab=ex
curve(dt(x=x, df=1), lty=2, add=TRUE) # prior for mu
points(ybar, 0, pch=19) # sample mean

curve(0.017*exp(lg(mu=x, n=n, ybar=ybar)), from=-1.0, to=3.0, add=TRUE, col
```



These results are encouraging, but they are preliminary. We still need to investigate more formally whether our Markov chain has converged to the stationary distribution. We will explore this in a future lesson.

Obtaining posterior samples using the Metropolis-Hastings algorithm can be time-consuming and require some fine-tuning, as we've just seen. The good news is that we can rely on software to do most of the work for us. In the next couple of videos, we'll introduce a program that will make posterior sampling easy.

7 Gibbs sampling

So far, we have demonstrated MCMC for a single parameter. What if we seek the posterior distribution of multiple parameters, and that posterior distribution does not have a standard form? One option is to perform Metropolis-Hastings (M-H) by sampling candidates for all parameters at once, and accepting or rejecting all of those candidates together. While this is possible, it can get complicated. Another (simpler) option is to sample the parameters one at a time.

As a simple example, suppose we have a joint posterior distribution for two parameters θ and ϕ, written $P(\theta, \phi|y) \propto g(\theta, \phi)$. If we knew the value of ϕ, then we would just draw a candidate for θ and use $g(\theta, \phi)$ to compute our Metropolis-Hastings ratio, and possibly accept the candidate. Before moving on to the next iteration, if we don't know ϕ, then we can perform a similar update for it. Draw a candidate for ϕ using some proposal distribution and again use $g(\theta, \phi)$ to compute our Metropolis-Hastings ratio. Here we pretend we know the value of θ by substituting its current iteration from the Markov chain. Once we've drawn for both θ and ϕ, that completes one iteration and we begin the next iteration by drawing a new θ. In other words, we're just going back and forth, updating the parameters one at a time, plugging the current value of the other parameter into $g(\theta, \phi)$.

This idea of one-at-a-time updates is used in what we call Gibbs sampling, which also produces a stationary Markov chain (whose stationary distribution is the posterior).

7.1 Full conditional distributions

Before describing the full Gibbs sampling algorithm, there's one more thing we can do. Using the chain rule of probability, we have $p(\theta, \phi \mid y) = p(\theta \mid \phi, y) \cdot p(\phi \mid y)$. notice that the only difference between $p(\theta, \phi \mid y)$ and $p(\theta \mid \phi, y)$ is multiplication by a factor that doesn't involve θ . Since the $g(\theta, \phi)$ function above, when viewed as a function of θ is proportional to both these expressions, we might as well have replaced it with $p(\theta \mid \phi, y)$ in our update for θ.This distribution $p(\theta \mid \phi, y)$ is called the full conditional distribution for θ. Why use it instead of $g(\theta, \phi)$? In some cases, the full conditional distribution is a standard distribution we know how to sample. If that happens, we no longer need to draw a candidate and decide whether to accept it. In fact, if we treat the full conditional distribution as a candidate proposal distribution, the resulting Metropolis-Hastings acceptance probability becomes exactly 1.

Gibbs samplers require a little more work up front because you need to find the full conditional distribution for each parameter. The good news is that all full conditional distributions have the same starting point: the full joint posterior distribution. Using the example above, we have $p(\theta \mid \phi, y) \propto p(\theta, \phi|y)$ where we simply now treat ϕ as a known number. Likewise, the other full conditional is $p(\phi \mid \theta, y) \propto p(\theta, \phi \mid y)$ where here, we consider θ to be a known number. We always start with the full posterior distribution. Thus, the process of finding full conditional distributions is the same as finding the posterior distribution of each parameter, pretending that all other parameters are known.

7.2 Gibbs sampler

The idea of Gibbs sampling is that we can update multiple parameters by sampling just one parameter at a time, cycling through all parameters and repeating. To perform the update for one particular parameter, we substitute in the current values of all other parameters.

Here is the algorithm. Suppose we have a joint posterior distribution for two parameters θ and ϕ, written $P(\theta, \phi \mid y)$. If we can find the distribution of each parameter at a time, i.e., $P(\theta \mid \phi, y)$ and $P(\phi \mid \theta, y)$, then we can take turns sampling these distributions like so:

1.Using ϕ_{i-1}draw θ_i from $P(\theta|\phi = \phi_{i-1}, y)$.

2.Using θ_i, draw ϕ_i from $P(\phi|\theta = \theta_i, y)$.

Together, steps 1 and 2 complete one cycle of the Gibbs sampler and produce the draw for (θ_i, ϕ_i) in one iteration of a MCMC sampler. If there are more than two parameters, we can handle that also. One Gibbs cycle would include an update for each of the parameters.

7.3 Example

7.3.1 Normal likelihood, unknown mean and variance

Let's make an example, where we have normal likelihood with unknown mean and unknown variance. The model is :

$$\begin{aligned} y_i \mid \mu, \sigma^2 &\overset{\text{iid}}{\sim} \mathrm{N}(\mu, \sigma^2), \quad i = 1, \ldots, n \\ \mu &\sim \mathrm{N}(\mu_0, \sigma_0^2) \\ \sigma^2 &\sim \mathrm{IG}(\nu_0, \beta_0) \end{aligned}$$

.

We chose a normal prior for μ because, in the case where σ^2 is known, the normal is the conjugate prior for μ. Likewise, in the case where μ is known, the inverse-gamma is the conjugate prior for σ^2. This will give us convenient full conditional distributions in a Gibbs sampler.

Let's first work out the form of the full posterior distribution. When we begin analyzing data, the JAGS software will complete this step for us. However, it is extremely valuable to see and understand how this works.

$$p(\mu, \sigma^2 \mid y_1, y_2, .$$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y_i-\mu)^2}{2\sigma^2}\right] \times \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right] \times \frac{\beta_0^{\nu_0}}{\Gamma(\nu_0)}(\sigma^2)^{-(\nu_0+1)} \exp\left[-\frac{\beta_0}{\sigma^2}\right] I_{\sigma^2}$$

$$\propto (\sigma^2)^{-n/2} \exp\left[-\frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2\sigma^2}\right] \exp\left[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right] (\sigma^2)^{-(\nu_0+1)} \exp\left[-\frac{\beta_0}{\sigma^2}\right] I_{\sigma^2}$$

From here, it is easy to continue on to find the two full conditional distributions we need. First let's look at μ, assuming σ^2 is known (in which case it becomes a constant and is absorbed into the normalizing constant):

$$\begin{aligned}
p(\mu \mid \sigma^2, y_1, \dots, y_n) &\propto p(\mu, \sigma^2 \mid y_1, \dots, y_n) \\
&\propto \exp\left[-\frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2\sigma^2}\right] \exp\left[-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right] \\
&\propto \exp\left[-\frac{1}{2}\left(\frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2\sigma^2} + \frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right)\right] \\
&\propto \text{N}\left(\mu \mid \frac{n\bar{y}/\sigma^2 + \mu_0/\sigma_0^2}{n/\sigma^2 + 1/\sigma_0^2}, \frac{1}{n/\sigma^2 + 1/\sigma_0^2}\right),
\end{aligned}$$

which we derived in the supplementary material of the last course. So, given the data and σ^2, μ follows this normal distribution.

Now let's look at σ^2, assuming μ is known:

$$\begin{aligned}
p(\sigma^2 \mid \mu, y_1, \dots, y_n) &\propto p(\mu, \sigma^2 \mid y_1, \dots, y_n) \\
&\propto (\sigma^2)^{-n/2} \exp\left[-\frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2\sigma^2}\right] (\sigma^2)^{-(\nu_0+1)} \exp\left[-\frac{\beta_0}{\sigma^2}\right] I_{\sigma^2>0}(\sigma^2) \\
&\propto (\sigma^2)^{-(\nu_0+n/2+1)} \exp\left[-\frac{1}{\sigma^2}\left(\beta_0 + \frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2}\right)\right] I_{\sigma^2>0}(\sigma^2) \\
&\propto \text{IG}\left(\sigma^2 \mid \nu_0 + \frac{n}{2}, \beta_0 + \frac{\sum_{i=1}^{n}(y_i-\mu)^2}{2}\right).
\end{aligned}$$

These two distributions provide the basis of a Gibbs sampler to simulate from a Markov chain whose stationary distribution is the full posterior of both μ and σ^2. We simply alternate draws between these two parameters, using the most recent draw of one parameter to update the other.

We will do this in R in the next segment.

7.3.1.1 Gibbs sampler in R

To implement the Gibbs sampler we just described, let's return to our running example where the data are the percent change in total personnel from last year to this year for *n=10* companies. We'll still use a normal likelihood, but now we'll relax the assumption that we know the variance of growth between companies, σ^2, and estimate that variance. Instead of the t prior from earlier, we will use the conditionally conjugate priors, normal for μ and inverse-gamma for σ^2

The first step will be to write functions to simulate from the full conditional distributions we derived in the previous segment. The full conditional for μ, given σ^2 and data is

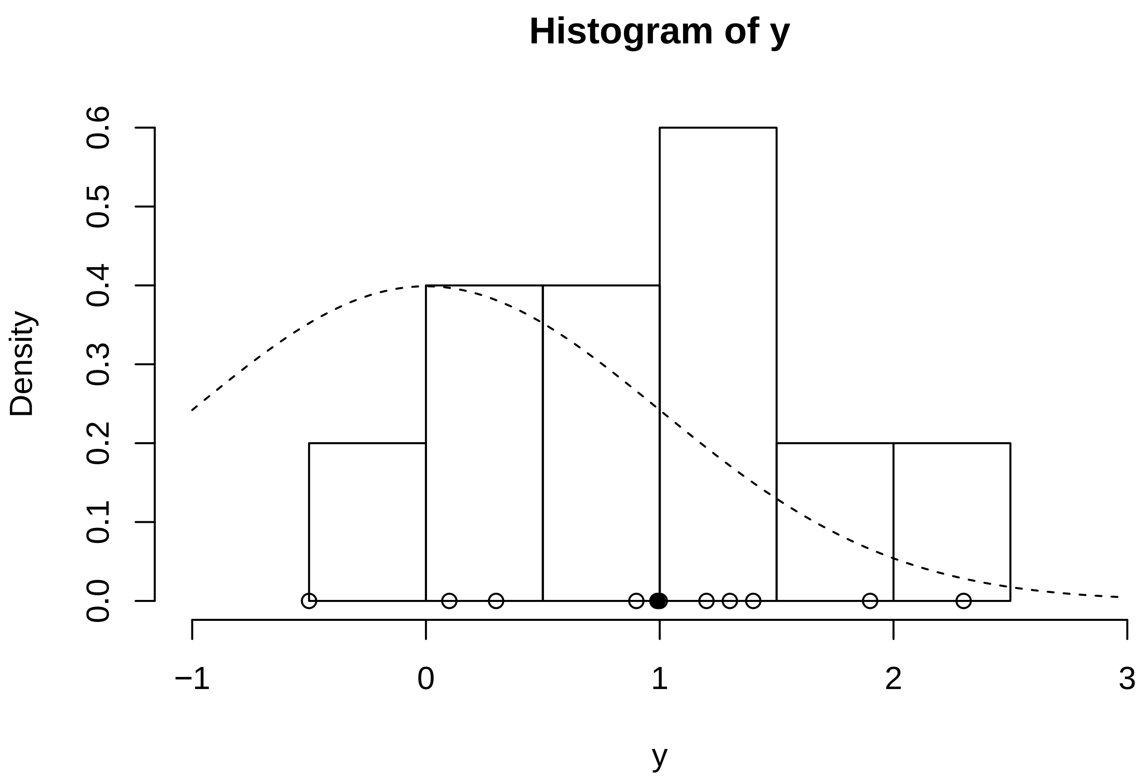
$\mathrm{N}\left(\mu \mid \frac{n\bar{y}/\sigma^2+\mu_0/\sigma_0^2}{n/\sigma^2+1/\sigma_0^2}, \frac{1}{n/\sigma^2+1/\sigma_0^2}\right)$

The full conditional for σ^2 given μ and data is:

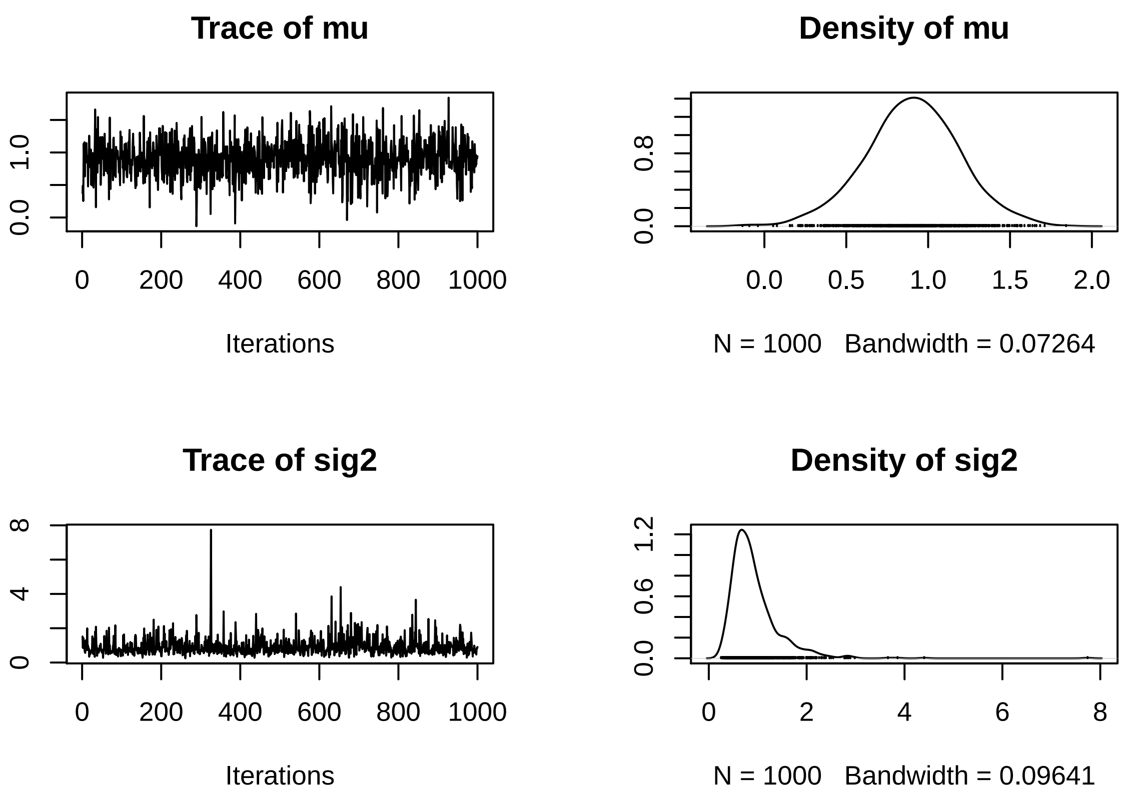
$\mathrm{IG}\left(\sigma^2 \mid \nu_0 + \frac{n}{2}, \beta_0 + \frac{\sum_{i=1}^n (y_i-\mu)^2}{2}\right)$

With functions for drawing from the full conditionals, we are ready to write a function to perform Gibbs sampling.

Now we are ready to set up the problem in R.



```
              mu      sig2
[1,] 0.3746992 1.5179144
[2,] 0.4900277 0.8532821
[3,] 0.2536817 1.4325174
[4,] 1.1378504 1.2337821
[5,] 1.0016641 0.8409815
[6,] 1.1576873 0.7926196
```



```
Iterations = 1:1000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 1000

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

       Mean     SD Naive SE Time-series SE
mu   0.9051 0.2868  0.00907        0.00907
sig2 0.9282 0.5177  0.01637        0.01810

2. Quantiles for each variable:
       2.5%    25%    50%   75% 97.5%
mu   0.3024 0.7244 0.9089 1.090 1.481
sig2 0.3577 0.6084 0.8188 1.094 2.141
```

As with the Metropolis-Hastings example, these chains appear to have converged. In the next lesson, we will discuss convergence in more detail.

8 Popular Models

8.1 Linear Regression

8.2 (M)ANOVA

8.3 Poisson regression

9 Multi-level modeling

9.1 Hierarchical models

9.1.1 Data

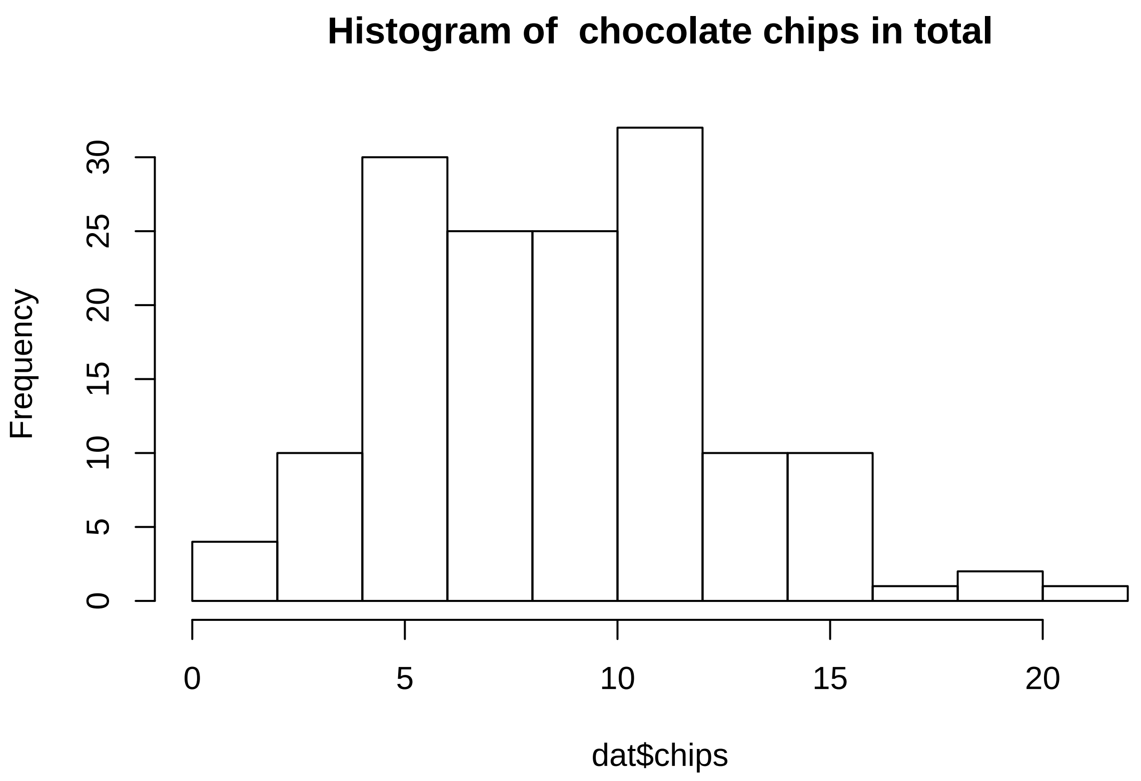
Let's fit our hierarhical model for counts of chocolate chips. The data can be found in

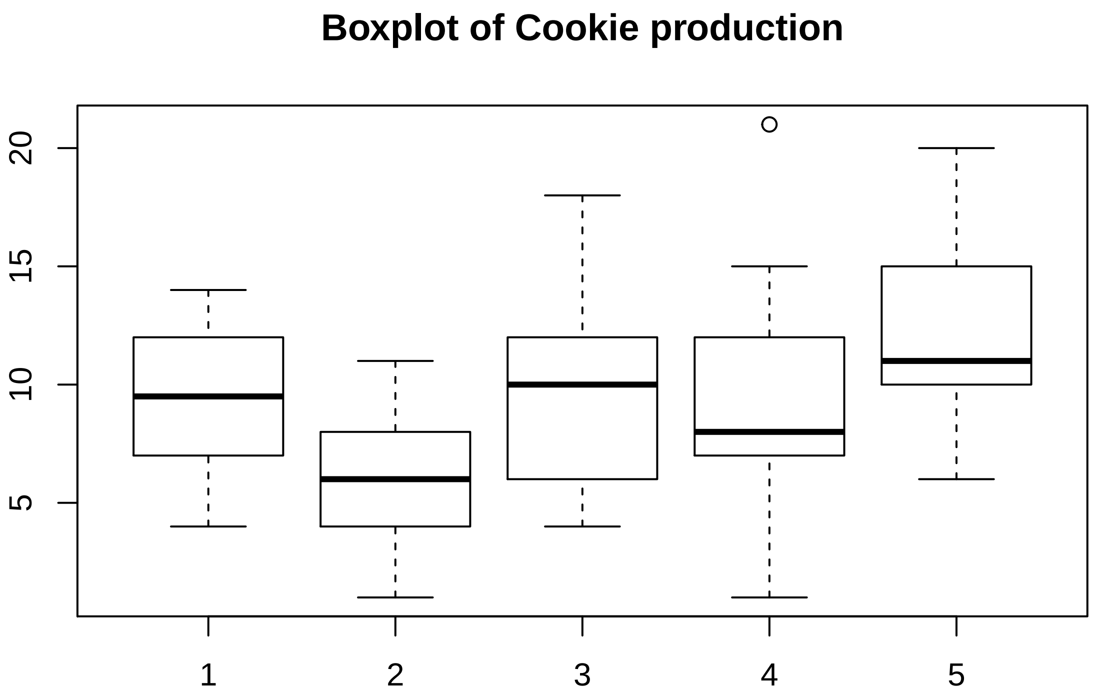
Table 1: First 10 values

chips	location
12	1
12	1
6	1
13	1
12	1
12	1
9	1
10	1
7	1
14	1

```
##
##  1  2  3  4  5
## 30 30 30 30 30
```

We can also visualize the distribution of chips by location.





9.1.2 Prior predictive checks

Before implementing the model, we need to select prior distributions for α and β, the hyperparameters governing the gamma distribution for the λ parameters. First, think about what the λ's represent. For location j, λ_j is the expected number of chocolate chips per cookie. Hence, α and β control the distribution of these means between locations. The mean of this gamma distribution will represent the overall mean of number of chips for all cookies. The variance of this gamma distribution controls the variability between locations. If this is high, the mean number of chips will vary widely from location to location. If it is small, the mean number of chips will be nearly the same from location to location.

To see the effects of different priors on the distribution of λ's, we can simulate. Suppose we try independent exponential priors for α and β.

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    0.021    2.983    9.852   61.127   29.980 4858.786
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##    0.1834    3.3663    8.5488   41.8137   22.2219 2865.6461
```

After simulating from the priors for α and β, we can use those samples to simulate further down the hierarchy:

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##     0.000     1.171     7.667    83.062    28.621 11005.331
```

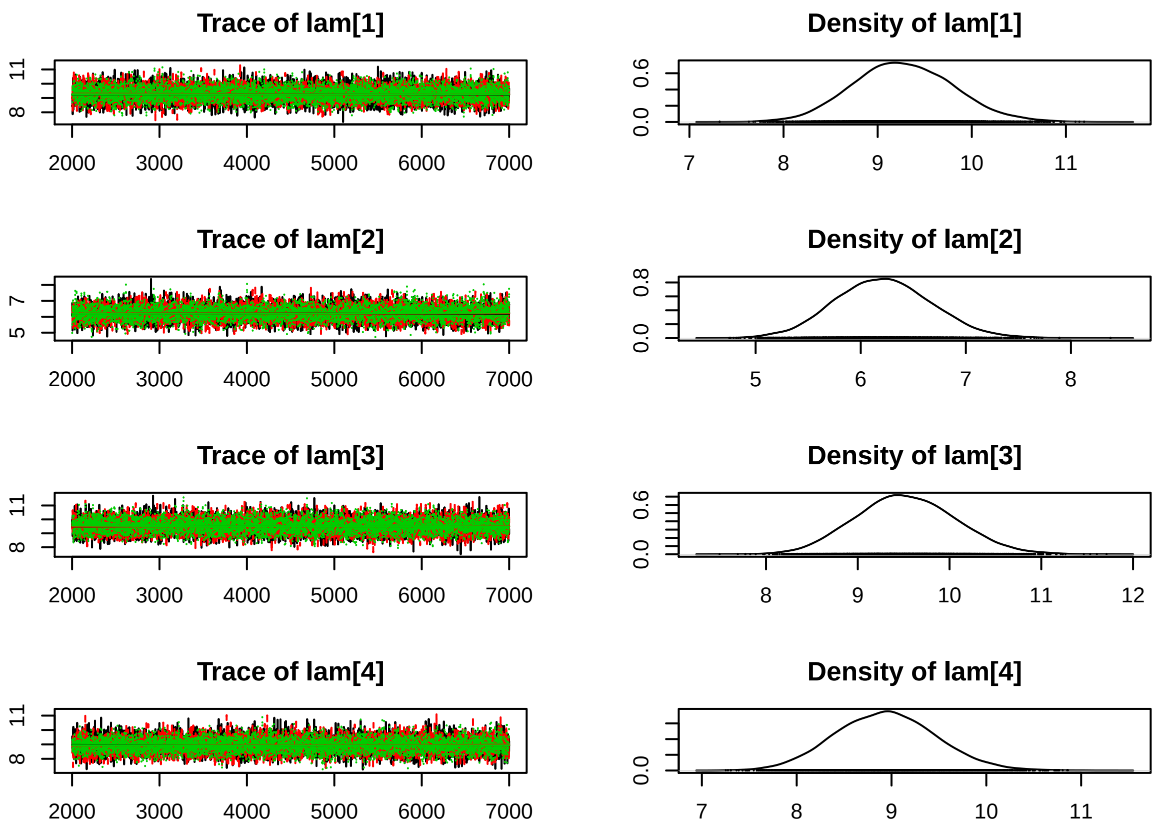
Or for a prior predictive reconstruction of the original data set:

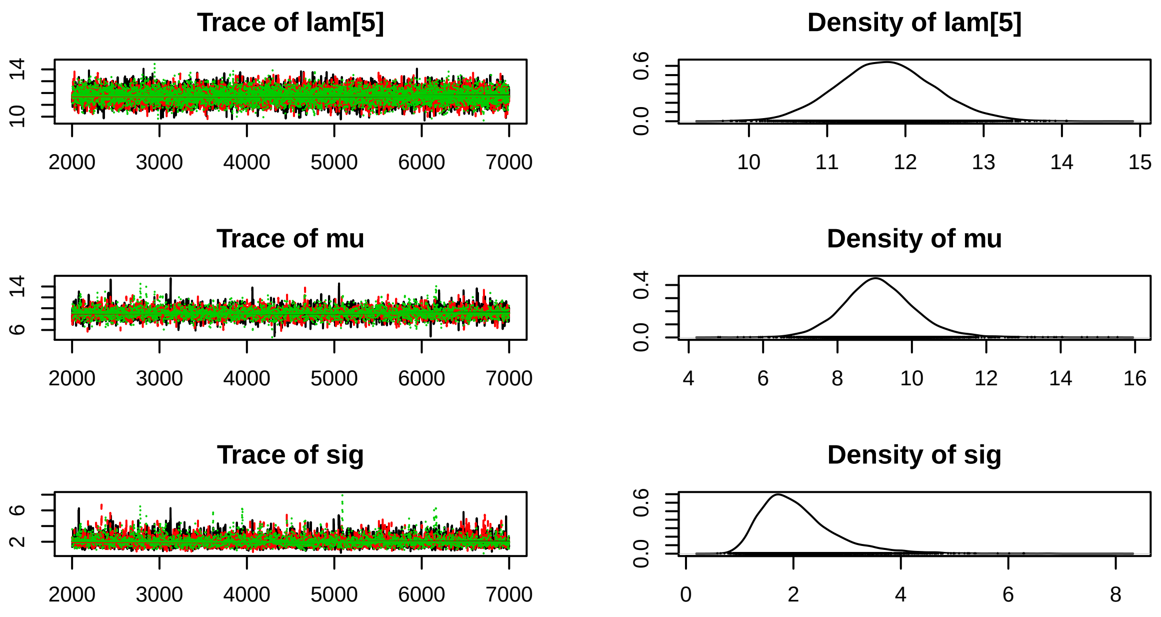
```
[1] 66.444084  9.946688  6.028319 15.922568 47.978587
```

```
  [1] 63 58 64 63 70 62 61 48 71 73 70 77 66 60 72 77 69 62 66 71 49 80 66
 [24] 75 74 55 62 90 65 57 12  9  7 10 12 10 11  7 14 13  9  6  6 13  7 10
 [47] 12  9  9 10  7  8  6  9  7 10 13 13  8 12  6 10  3  6  7  4  6  7  5
 [70]  5  4  3  6  2  8  4  8  4  5  7  1  4  5  3  8  8  3  1  7  3 16 14
 [93] 13 17 17 12 13 13 16 16 15 14 11 10 13 17 16 19 16 17 15 16  7 17 21
[116] 16 12 15 14 13 52 44 51 46 39 40 40 44 46 59 45 49 58 42 31 52 43 47
[139] 53 41 48 57 35 60 51 58 36 34 41 59
```

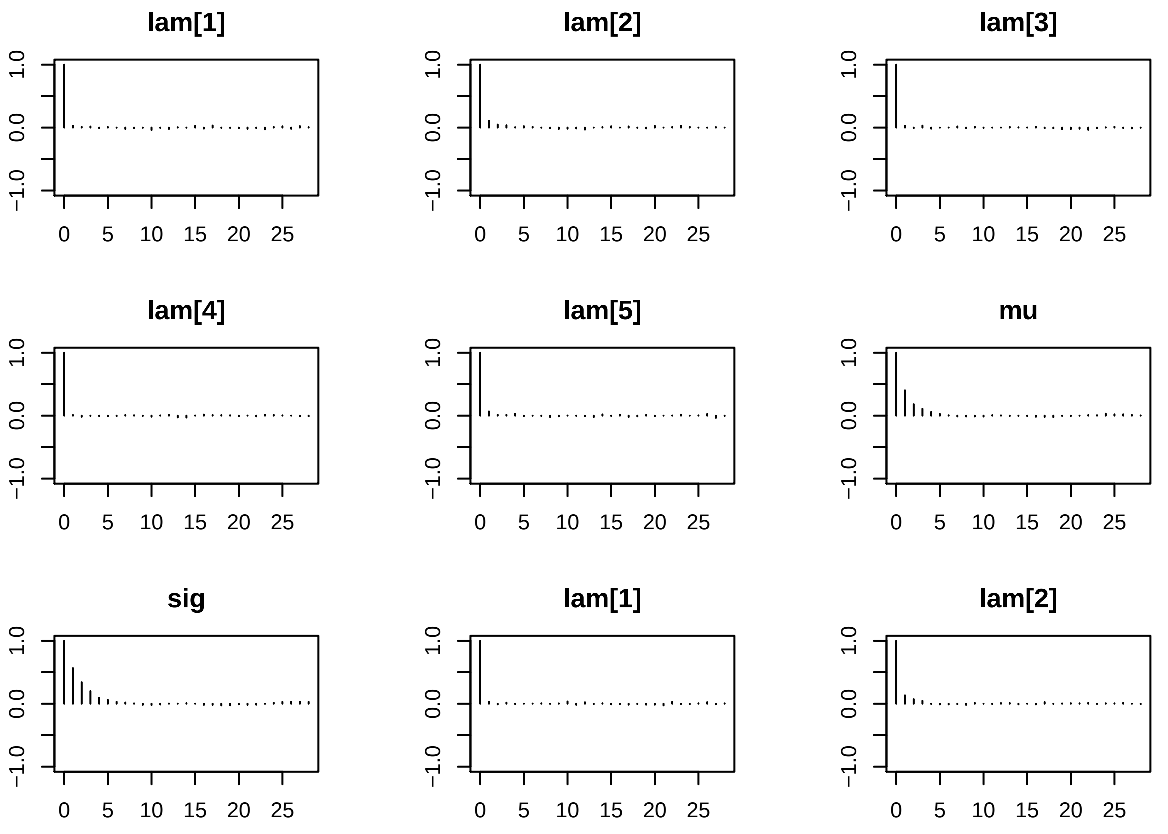
Because these priors have high variance and are somewhat noninformative, they produce unrealistic predictive distributions. Still, enough data would overwhelm the prior, resulting in useful posterior distributions. Alternatively, we could tweak and simulate from these prior distributions until they adequately represent our prior beliefs. Yet another approach would be to re-parameterize the gamma prior, which we'll demonstrate as we fit the model.

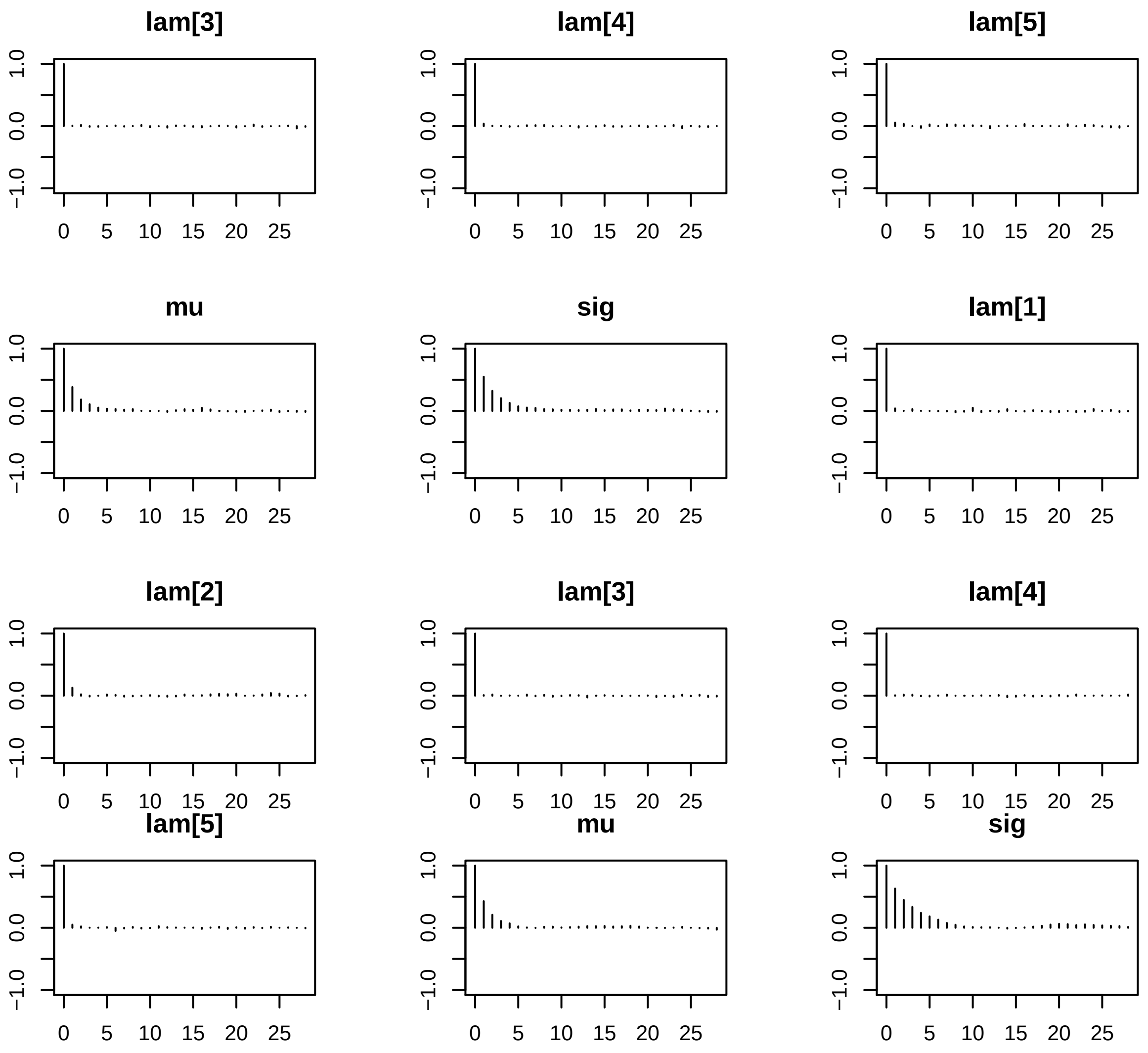
```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 150
   Unobserved stochastic nodes: 7
   Total graph size: 319


Initializing model
```





```
Potential scale reduction factors:

       Point est. Upper C.I.
lam[1]          1          1
lam[2]          1          1
lam[3]          1          1
lam[4]          1          1
lam[5]          1          1
mu              1          1
sig             1          1

Multivariate psrf

1

             lam[1]      lam[2]       lam[3]       lam[4]      lam[5]
Lag 0   1.000000000 1.000000000  1.000000000  1.000000000 1.000000000
Lag 1   0.032270952 0.122734353  0.014562573  0.018027027 0.057291103
Lag 5   0.002379583 0.009699435 -0.000275629 -0.008141504 0.009812255
Lag 10  0.015549353 -0.003138788 -0.009652451 -0.006642067  0.003040346
Lag 50 -0.019336430  0.005404646  0.006474722 -0.030316901 -0.008268752
                 mu         sig
Lag 0   1.000000000 1.000000000
Lag 1   0.404492740 0.581691539
Lag 5   0.028003801 0.105427017
Lag 10 -0.003357365 0.003504738
Lag 50  0.002102249 0.005189040
```





```
   lam[1]     lam[2]     lam[3]     lam[4]     lam[5]         mu        sig
13798.431 10623.481 14511.131 14627.212 13287.724  5715.464  3544.364
```

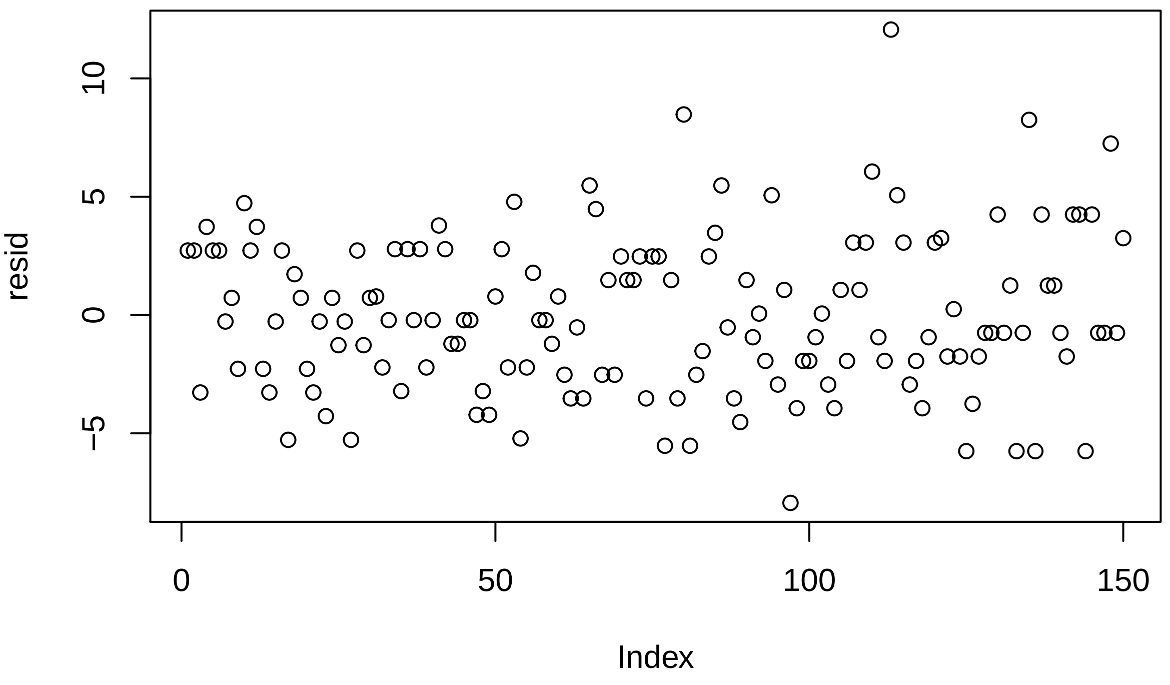
9.1.3 Model checking

After assessing convergence, we can check the fit via residuals. With a hierarhcical model, there are now two levels of residuals: the observation level and the location mean level. To simplify, we'll look at the residuals associated with the posterior means of the parameters.

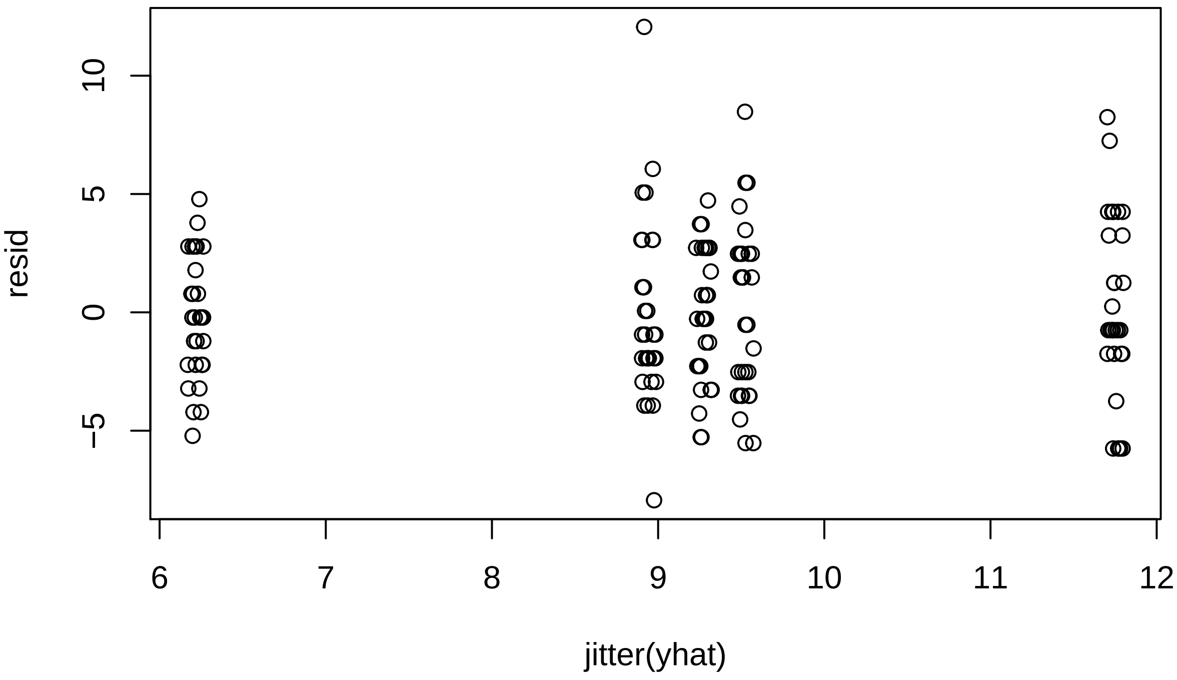
First, we have observation residuals, based on the estimates of location means.

```
## observation level residuals
```

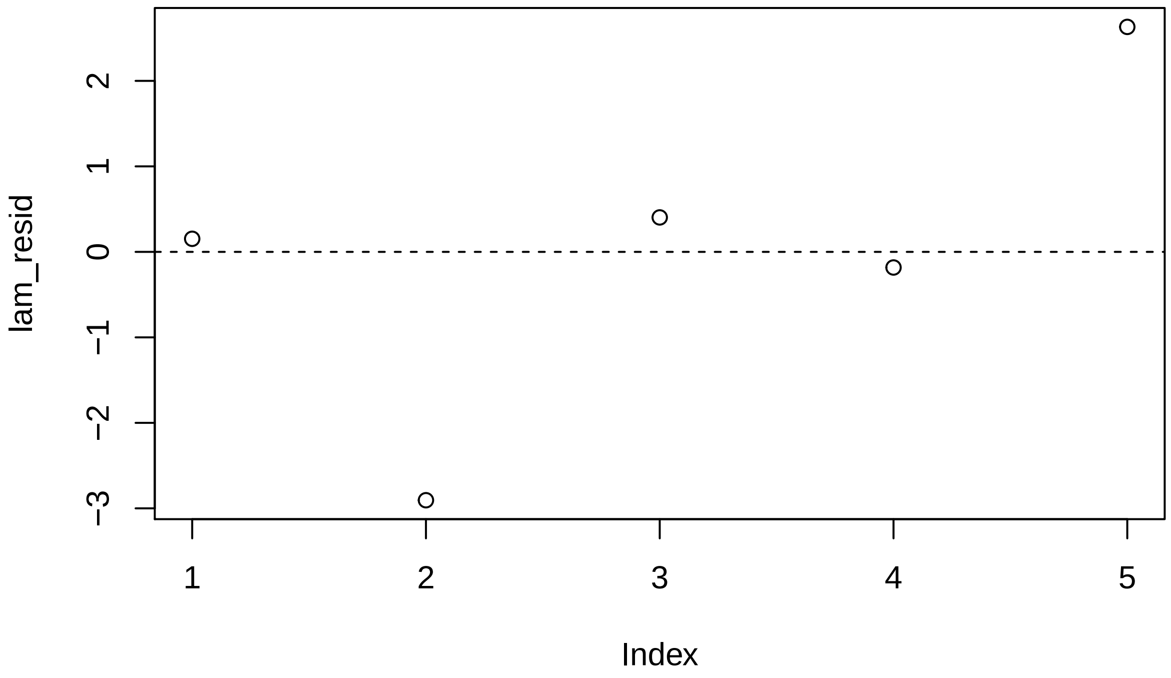
```
(pm_params = colMeans(mod_csim))

   lam[1]    lam[2]    lam[3]    lam[4]    lam[5]        mu       sig
 9.274796  6.217039  9.524885  8.939276 11.753045  9.121894  2.089785

yhat = rep(pm_params[1:5], each=30)
resid = dat$chips - yhat
plot(resid)
```



```
plot(jitter(yhat), resid)
```



```
var(resid[yhat<7])

[1] 6.447126

var(resid[yhat>11])

[1] 13.72414

## location level residuals
lam_resid = pm_params[1:5] - pm_params["mu"]
plot(lam_resid)
abline(h=0, lty=2)
```



We don't see any obvious violations of our model assumptions.

9.1.4 Results

```
summary(mod_sim)
```

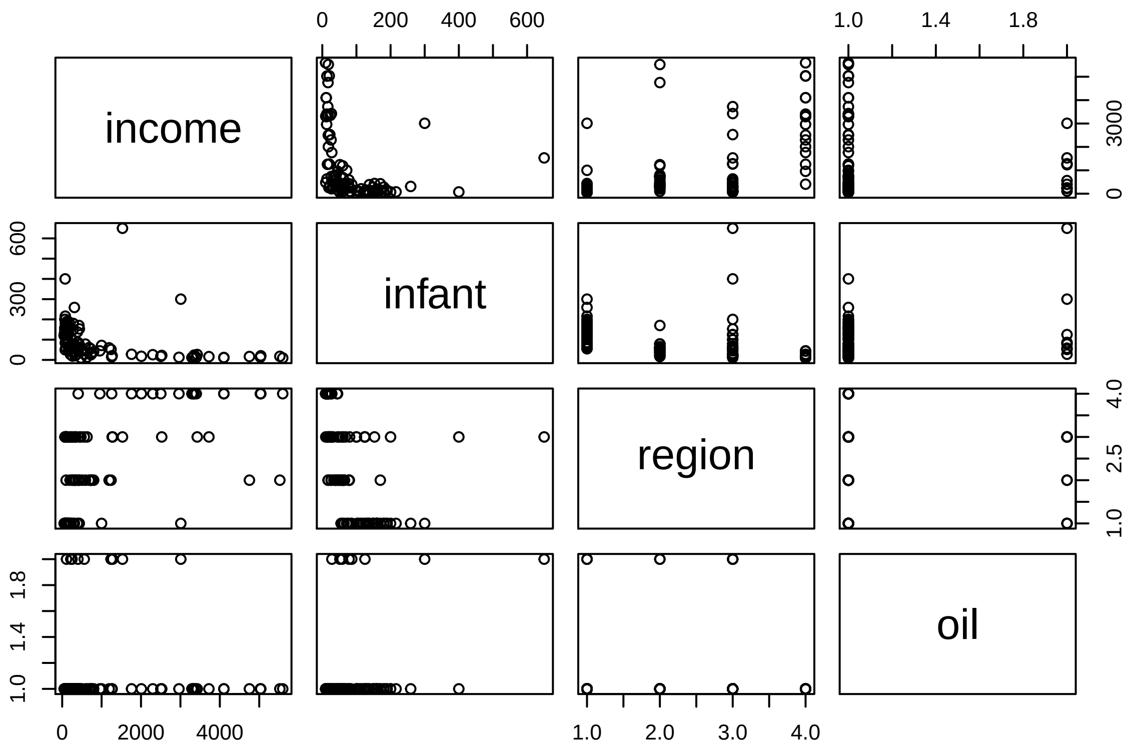
```

Iterations = 2001:7000
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]   9.275 0.5354 0.004372       0.004561
lam[2]   6.217 0.4643 0.003791       0.004516
```

```
lam[3]   9.525 0.5482 0.004476       0.004555
lam[4]   8.939 0.5304 0.004330       0.004389
lam[5] 11.753 0.6214 0.005074       0.005396
mu       9.122 0.9988 0.008155       0.013216
sig      2.090 0.7219 0.005894       0.012427

2. Quantiles for each variable:

          2.5%    25%    50%    75%  97.5%
lam[1]   8.268  8.909  9.260  9.637 10.367
lam[2]   5.331  5.899  6.210  6.521  7.151
lam[3]   8.495  9.150  9.511  9.892 10.627
lam[4]   7.929  8.572  8.935  9.291 10.005
lam[5] 10.566 11.335 11.742 12.157 13.020
mu       7.276  8.486  9.079  9.703 11.234
sig      1.104  1.590  1.953  2.424  3.914
```

9.1.5 Random intercept linear model

We can extend the linear model for the Leinhardt data on infant mortality by incorporating the region variable. We'll do this with a hierarhcical model, where each region has its own intercept.

```
'data.frame':   105 obs. of  4 variables:
 $ income: int  3426 3350 3346 4751 5029 3312 3403 5040 2009 2298 ...
 $ infant: num  26.7 23.7 17 16.8 13.5 10.1 12.9 20.4 17.8 25.7 ...
 $ region: Factor w/ 4 levels "Africa","Americas",..: 3 4 4 2 4 4 4 4 4 4 .
 $ oil   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```



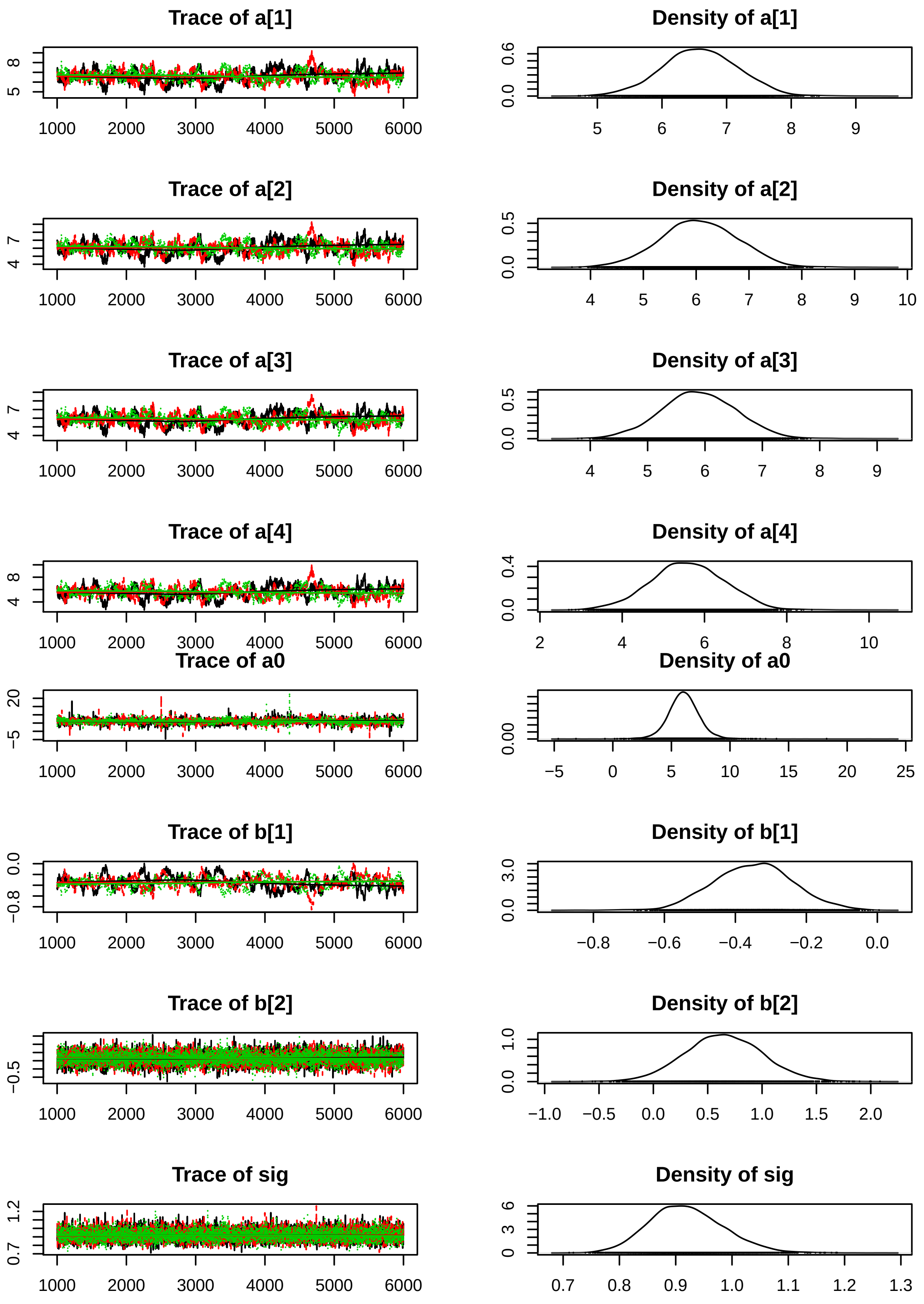
```
          income infant   region oil
Australia   3426   26.7     Asia  no
Austria     3350   23.7   Europe  no
Belgium     3346   17.0   Europe  no
Canada      4751   16.8 Americas  no
Denmark     5029   13.5   Europe  no
Finland     3312   10.1   Europe  no
```

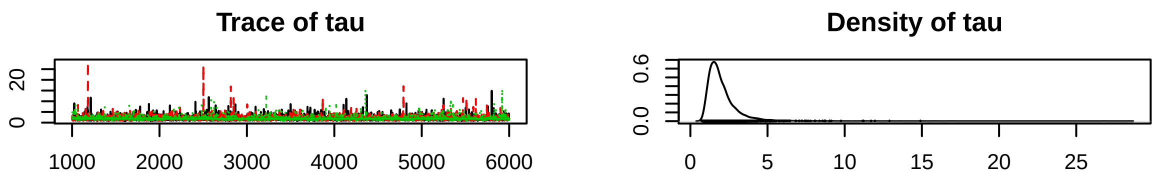
Previously, we worked with infant mortality and income on the logarithmic scale. Recall also that we had to remove some missing data.

```
'data.frame':   101 obs. of  6 variables:
 $ income   : int  3426 3350 3346 4751 5029 3312 3403 5040 2009 2298 ...
 $ infant   : num  26.7 23.7 17 16.8 13.5 10.1 12.9 20.4 17.8 25.7 ...
 $ region   : Factor w/ 4 levels "Africa","Americas",..: 3 4 4 2 4 4 4 4 4
 $ oil      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ logincome: num  8.14 8.12 8.12 8.47 8.52 ...
 $ loginfant: num  3.28 3.17 2.83 2.82 2.6 ...
 - attr(*, "na.action")=Class 'omit'  Named int [1:4] 24 83 86 91
  .. ..- attr(*, "names")= chr [1:4] "Iran" "Haiti" "Laos" "Nepal"
```

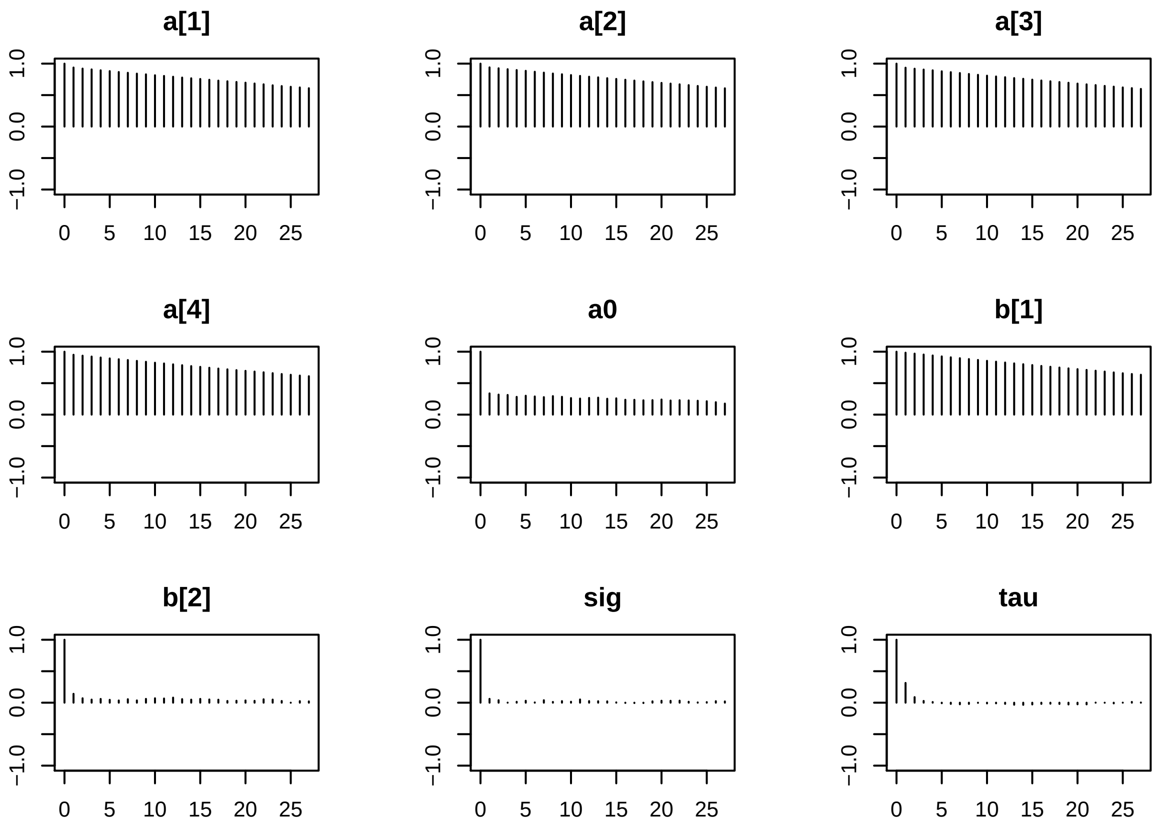
Now we can fit the proposed model:

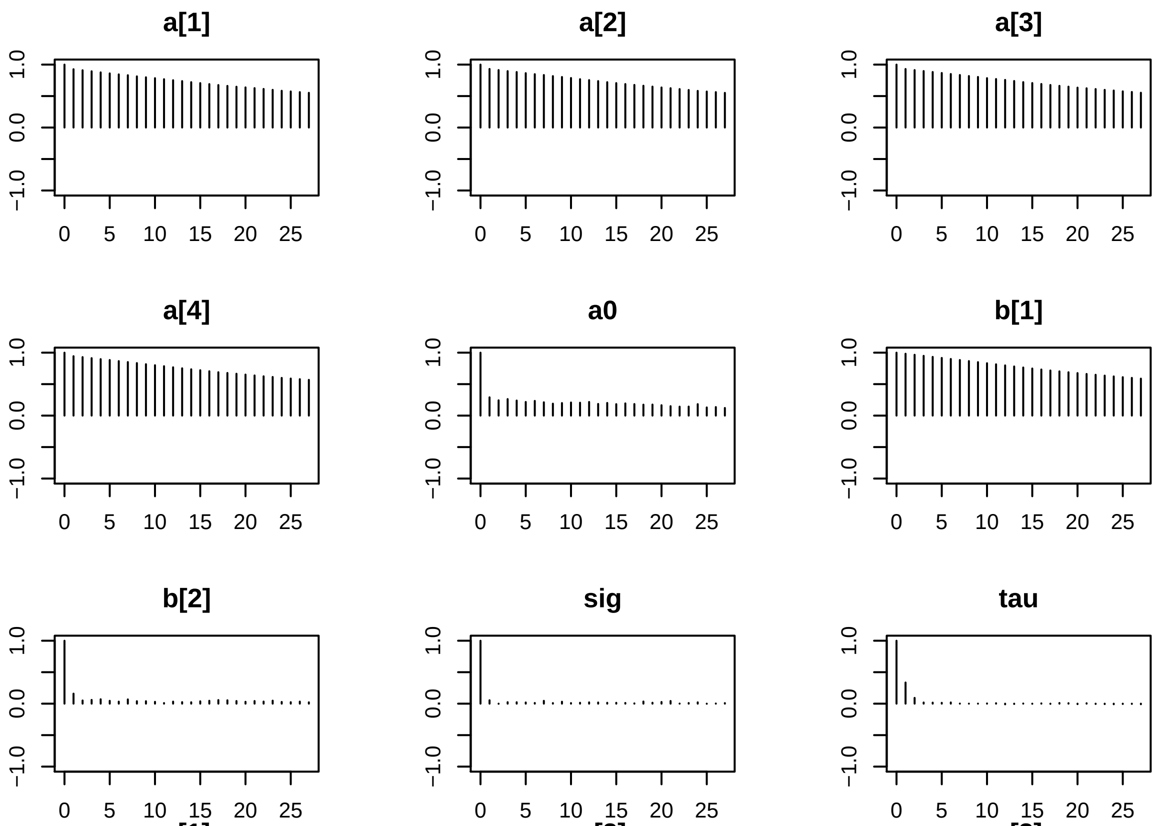
```
  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


     1  2  3  4
  0 31 20 24 18
  1  3  2  3  0
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 101
   Unobserved stochastic nodes: 9
   Total graph size: 639

Initializing model
```

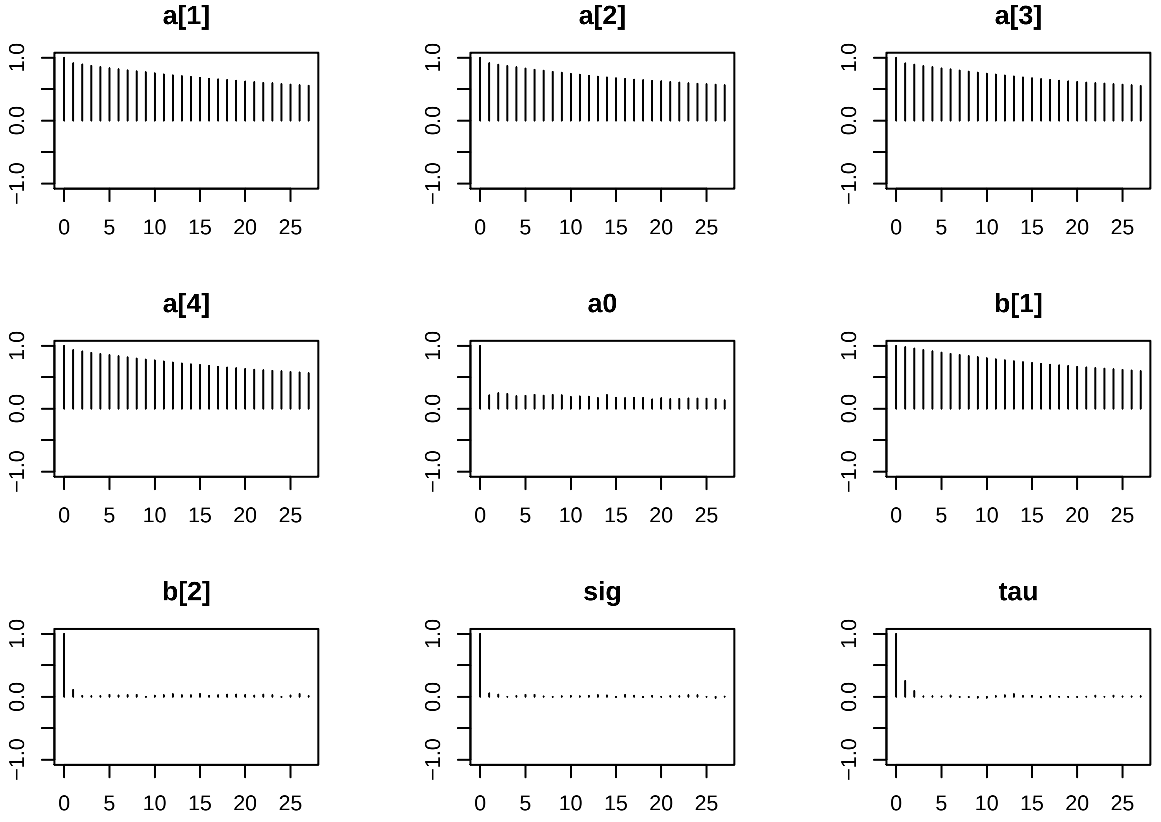

Trace of a[1]
5 8
1000 2000 3000 4000 5000 6000
Density of a[1]
0.0 0.6
5 6 7 8 9
Trace of a[2]
4 7
1000 2000 3000 4000 5000 6000
Density of a[2]
0.0 0.5
4 5 6 7 8 9 10
Trace of a[3]
4 7
1000 2000 3000 4000 5000 6000
Density of a[3]
0.0 0.5
4 5 6 7 8 9
Trace of a[4]
4 8
1000 2000 3000 4000 5000 6000
Density of a[4]
0.0 0.4
2 4 6 8 10
Trace of a0
−5 20
1000 2000 3000 4000 5000 6000
Density of a0
0.00
−5 0 5 10 15 20 25
Trace of b[1]
−0.8 0.0
1000 2000 3000 4000 5000 6000
Density of b[1]
0.0 3.0
−0.8 −0.6 −0.4 −0.2 0.0
Trace of b[2]
−0.5
1000 2000 3000 4000 5000 6000
Density of b[2]
0.0 1.0
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0
Trace of sig
0.7 1.2
1000 2000 3000 4000 5000 6000
Density of sig
0 3 6
0.7 0.8 0.9 1.0 1.1 1.2 1.3



```
Potential scale reduction factors:

     Point est. Upper C.I.
a[1]       1.02       1.08
a[2]       1.02       1.08
a[3]       1.02       1.08
a[4]       1.02       1.08
a0         1.01       1.03
b[1]       1.03       1.08
b[2]       1.00       1.01
sig        1.00       1.00
tau        1.00       1.00

Multivariate psrf

1.02
            a[1]      a[2]      a[3]      a[4]        a0      b[1]
Lag 0  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Lag 1  0.9254806 0.9278306 0.9250892 0.9420967 0.2802747 0.9817108
Lag 5  0.8582312 0.8594187 0.8584423 0.8763527 0.2409722 0.9111218
Lag 10 0.7840719 0.7840017 0.7803901 0.7976858 0.2197752 0.8295286
Lag 50 0.3852481 0.3847752 0.3824178 0.3926753 0.1036313 0.4063930
             b[2]         sig         tau
Lag 0  1.00000000 1.000000000  1.000000000
Lag 1  0.13649960 0.056498700  0.300134890
Lag 5  0.04071449 0.026846721  0.003345875
Lag 10 0.03924521 0.012637959 -0.008013147
Lag 50 0.02925338 0.006688377  0.003003689
```







```
    a[1]        a[2]        a[3]        a[4]           a0         b[1]
```

```
  152.9531   154.8190   152.7232   143.3869   698.0995   138.4393 
      b[2]        sig        tau 
 4270.7174 10034.2582  8176.1755 
```

9.2 Meta analysis

Bibliography

Efron, B., 2013. Bayes theorem in the 21st century. Science 340, 1177–1178.