Module 7, part II: Bayesian Linear Regression

BIOS 526

Concepts

- Priors for Bayesian linear regression.
- Posterior inference.
- Working with posterior samples.

Linear Regression Model

We will consider the multiple linear regression model. For $i = 1, \ldots, n$, assume

$$y_i = \boldsymbol{x}_i' \boldsymbol{\beta} + \epsilon_i, \quad \epsilon_i \stackrel{iid}{\sim} N(0, \sigma^2).$$

Here we have $(p+1)$ unknown parameters $(\beta_1, \ldots, \beta_p, \sigma^2)$.

The above model can be written in matrix form and the observed data vector **y** has distribution

$$\mathbf{y} \sim N\left(\mathbf{X}\beta,\ \sigma^2 \mathbf{I}_{n \times n}\right)$$

with likelihood

$$[\mathbf{y}|\boldsymbol{\beta}, \sigma^2] = (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left[-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\beta)'(\mathbf{y} - \mathbf{X}\beta)\right].$$

To carry out Bayesian inference, we need to assign prior distributions for $\boldsymbol{\beta}$ and σ^2.

Priors for Linear Regression Model

The standard priors for linear regression parameters are non-informative multivariate normal distribution for $\boldsymbol{\beta}$ and inverse-gamma for σ^2:

$$\boldsymbol{\beta} \sim N\left(\boldsymbol{\mu}_0,\, \tau^2 \mathbf{I}_{p\times p}\right) \qquad \text{and} \qquad \sigma^2 \sim \text{Inv-gamma}\ (c_0, d_0),$$

where τ^2 is set to be large (e.g. 10000); and c_0 and d_0 are set to be small (e.g. 0.0001). $\boldsymbol{\mu}_0$ is often set to be $\mathbf{0}$.

The above also assumes that the regression coefficients are *a priori* independent of each other. These hyper-parameter choices attempt to reflect a lack of prior information for the parameters.

Estimation is usually carried out using a Gibbs sampler that updates $\boldsymbol{\beta}$ and σ^2 iteratively. We will need to calculate the full conditional distributions:

$$[\,\boldsymbol{\beta} \,|\, \sigma^2, \mathbf{y}\,] \qquad \text{and} \qquad [\,\sigma^2 \,|\, \boldsymbol{\beta}, \mathbf{y}\,].$$

It turns out that the above two distributions have closed-form solutions.

Full Conditional Distribution for Linear Regression Model

Full conditional distribution of $\boldsymbol{\beta}$

$$[\,\boldsymbol{\beta}\,|\,\sigma^2, \mathbf{y}\,] \sim N\,(\tilde{\mu}, \tilde{\mathbf{V}}\,)$$

where $$\tilde{\mu} = \left[\sigma^{-2}(\mathbf{X}'\mathbf{X}) + \tau^{-2}\mathbf{I}_{p\times p}\right]^{-1}\left[\sigma^{-2}(\mathbf{X}'\mathbf{y}) + \tau^{-2}\boldsymbol{\mu}_0\right]$$

$$\tilde{\mathbf{V}} = \left[\sigma^{-2}(\mathbf{X}'\mathbf{X}) + \tau^{-2}\mathbf{I}_{p\times p}\right]^{-1}$$

Full conditional distribution of σ^2

$$[\,\sigma^2\,|\,\boldsymbol{\beta}, \mathbf{y}\,] \sim \text{Inv-gamma}\ (\tilde{c}, \tilde{d})$$

where $\tilde{c} = \frac{n}{2} + c_0$ and $\tilde{d} = \frac{\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2}{2} + d_0$

Note the similarities between the above two expressions and what we saw for the univariate Normal mean and variance in M7, part I.

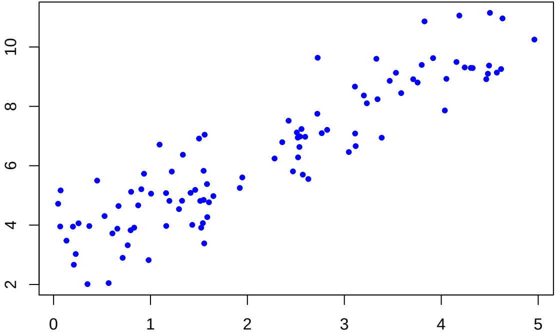
Linear Regression Example

Consider the following simple linear regression analysis. We wish to fit the model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2).$$

Simulate some data:

```
beta0 = 3
beta1 = 1.5
sigmasq = 1
Y = beta0+beta1*X+rnorm(n)
```



The MCMCpack Library

R has a package that provides functions to fit regression models (e.g. linear, logistic, Poisson). *MCMCregress* fits linear regression models.

```
> library (MCMCpack)
> MCMCregress
function (formula, data = NULL, burnin = 1000, mcmc = 10000,
    thin = 1, verbose = 0, seed = NA, beta.start = NA, b0 = 0,
    B0 = 0, c0 = 0.001, d0 = 0.001, ...)
```

- formula: like lm()
- burnin: number of initial pre-convergence samples to discard.
- mcmc: number of MCMC samples for inference.
- thin: k: keep the k^{th} MCMC sample only. Because each MCMC sample depends on the previous value, this helps reduce auto-correlation in the samples. It also saves memory storage.
- seed: seed for the random number generator. Same seed $=$ same posterior samples.
- beta.start: initial values for $\boldsymbol{\beta}$. If not provided, use the OLS estimates.

The MCMCpack Library

```
> library (MCMCpack)
> MCMCregress
function (formula, data = NULL, burnin = 1000, mcmc = 10000,
    thin = 1, verbose = 0, seed = NA, beta.start = NA, b0 = 0,
    B0 = 0, c0 = 0.001, d0 = 0.001, ...)
```

- b0 $=$ vector of prior means for regression coefficients (i.e. $\boldsymbol{\mu}_0$).
- B0 $=$ inverse of the prior covariance matrix for $\boldsymbol{\beta}$. This is also known as the precision matrix. For a scalar value, this corresponds to a covariance matrix $(1/B0) \times \mathbf{I}_{p\times p}$.
- c0 $=$ shape parameter/2 for the inverse-gamma prior for σ^2.
- d0 $=$ scale parameter/2 for the inverse-gamma prior for σ^2.

So for

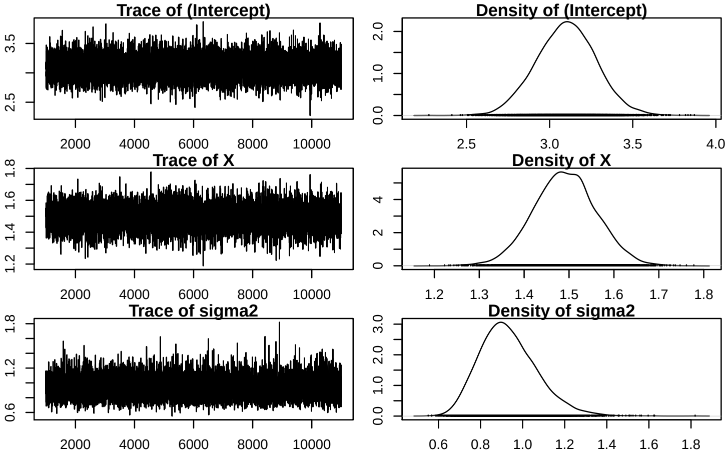
$$\boldsymbol{\beta} \sim N\left(\mathbf{0},\, 1000^2\mathbf{I}_{2\times 2}\right) \qquad \text{and} \qquad \sigma^2 \sim \text{Inv-gamma}\ (0.01, 0.01),$$

we have

```
b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01
```

Example: Trace Plots and Marginal Posterior Densities

```
> fit = MCMCregress (Y~X, b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01)
> plot(fit)
```



Example: Posterior Summary Statistics

```
> class(fit)
[1] "mcmc"
> summary(fit)

Iterations = 1001:11000
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
(Intercept) 3.1002 0.18018 0.0018018      0.0018018
X           1.4908 0.07007 0.0007007      0.0007007
sigma2      0.9328 0.13705 0.0013705      0.0014094

2. Quantiles for each variable:

              2.5%    25%    50%   75% 97.5%
(Intercept) 2.7493 2.9802 3.1021 3.220 3.452
X           1.3542 1.4444 1.4904 1.536 1.629
sigma2      0.7026 0.8365 0.9193 1.016 1.233
```

Example: Posterior Summary Statistics

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

              Mean      SD  Naive SE Time-series SE
(Intercept) 3.1002 0.18018 0.0018018      0.0018018
X           1.4908 0.07007 0.0007007      0.0007007
sigma2      0.9328 0.13705 0.0013705      0.0014094

2. Quantiles for each variable:

              2.5%    25%    50%   75% 97.5%
(Intercept) 2.7493 2.9802 3.1021 3.220 3.452
X           1.3542 1.4444 1.4904 1.536 1.629
sigma2      0.7026 0.8365 0.9193 1.016 1.233
```

- Mean: mean of the posterior distribution (point estimate).
- SD: standard deviation of the posterior distribution (uncertainty measure). Note: similar to SE of the mean in frequentist.
- Naive SE: $\mathrm{SD}/\sqrt{\mathrm{iterations}}$. This is a measure of Monte Carlo error for the point estimate.
- Time-series SE: same as *Naive SE* but includes a correction for dependent samples.
- Quantiles: quantile values of the posterior distribution (interval estimate).

Example: Posterior Summary Statistics

```
              Mean      SD  Naive SE Time-series SE
(Intercept) 3.1002 0.18018 0.0018018      0.0018018
X           1.4908 0.07007 0.0007007      0.0007007
sigma2      0.9328 0.13705 0.0013705      0.0014094

2. Quantiles for each variable:

              2.5%    25%    50%   75% 97.5%
(Intercept) 2.7493 2.9802 3.1021 3.220 3.452
X           1.3542 1.4444 1.4904 1.536 1.629
sigma2      0.7026 0.8365 0.9193 1.016 1.233
```

- The point estimate of β_1 is 1.491.
- Here, posterior medians of regression coefficients are close to posterior means. Posterior distributions appear to be symmetric.
- The 95% posterior interval of σ^2 is $(0.7026, 1.233)$. Its distribution is a little skewed to the right, so it differs a little bit from assuming normality: $0.9328 \pm 1.96 \times 0.137 = (0.66, 1.20)$ (sometimes it makes a big difference).

Compared to Frequentist Approach

```
> fit.lm = lm(Y~X)
> summary(fit.lm)

Call:
lm(formula = Y ~ X)

Residuals:
     Min       1Q   Median       3Q      Max
-2.03362 -0.66078 -0.08602  0.57926  2.47401

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0992     0.1784   17.37   <2e-16 ***
X             1.4911     0.0689   21.64   <2e-16 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 0.9553 on 98 degrees of freedom
Multiple R-squared:  0.827,Adjusted R-squared:  0.8252
F-statistic: 468.4 on 1 and 98 DF,  p-value: < 2.2e-16
```

Regression coefficients are nearly identical to the Bayesian analysis. $\hat{\sigma}^2 = 0.9553^2 = 0.9126$ is close to before. This is because we used uninformative priors.

Extract Posterior Samples

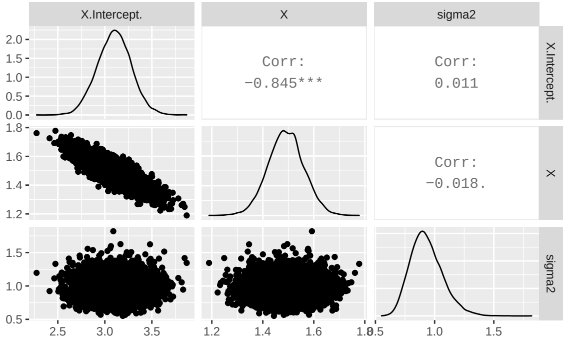
Often we want to work with the posterior samples directly. We can easily extract them into a matrix.

```
> post.samp = as.matrix(fit)
> dim(post.samp)
[1] 10000     3
> post.samp[1:3,]
     (Intercept)        X    sigma2
[1,]    3.187779 1.498849 1.0819978
[2,]    3.062054 1.551665 0.8508068
[3,]    3.177514 1.427804 1.1606806

> cor(post.samp)
            (Intercept)           X      sigma2
(Intercept)  1.00000000 -0.84513530  0.01109144
X           -0.84513530  1.00000000 -0.01760324
sigma2       0.01109144 -0.01760324  1.00000000
```

Pairwise Joint Posterior Distributions

```
### Plot pairwise-scatter plots
> pairs (pos
```



We note that $[\beta_0, \beta_1 \mid \mathbf{y}]$ is highly negatively correlated.

Bayesian Posterior Inference

Given,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2).$$

How do we answer the following questions:

- Predictions:
  1. Point estimate and 95% interval of $\beta_0 + 2\beta_1$?
  2. Point estimate and 95% interval of $\beta_0 + 2\beta_1 + \epsilon_i$?
- Functions of parameters:
  1. Point estimate and 95% interval of $\exp(\beta_1) - \beta_1$?
  2. Probability that $\beta_1/\beta_0 > 0.5$?

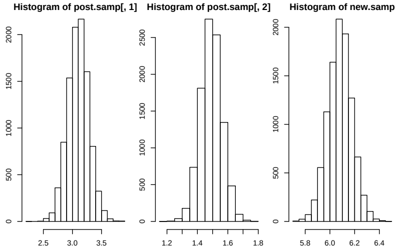
Another advantage of Bayesian inference is that we can easily estimate and quantify the uncertainties associated any statistic using posterior samples. Let $\boldsymbol{\theta}^{(k)}$ be the k^{th} posterior sample. Note that

$$E[g(\boldsymbol{\theta})] \approx \frac{1}{K} \sum_{k=1}^{K} g(\boldsymbol{\theta}^{(k)}).$$

Estimate and 95% Interval of $\beta_0 + 2\beta_1$?

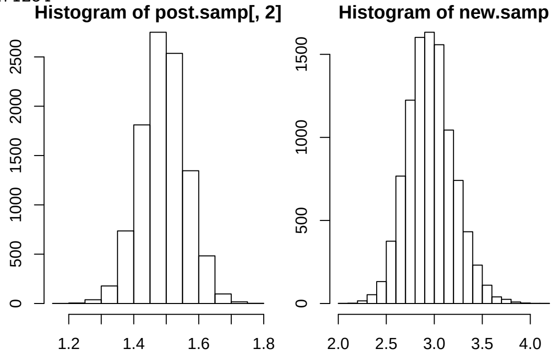
We have samples of $\beta_0^{(k)}$ and $\beta_1^{(k)}$. Calculate $\beta_0^{(k)} + 2\beta_1^{(k)}$ for each k:

```
> new.samp = post.samp[,1] + 2*post.samp[,2]
> mean(new.samp) ###Point estimate
[1] 6.081734
> quantile(new.samp, c(.025, .975)) ### 95% posterior interval
    2.5%    97.5%
5.888798 6.276088
```



Point estimate and 95% interval of $\exp(\beta_1) - \beta_1$?

```
> new.samp = exp(post.samp[,2]) - post.samp[,2]
> mean (new.samp)
[1] 2.960662
> quantile (new.samp, c(0.025, .975)) # correct way to calcualte central 95% cred int
    2.5%    97.5%
2.519534 3.471284
```

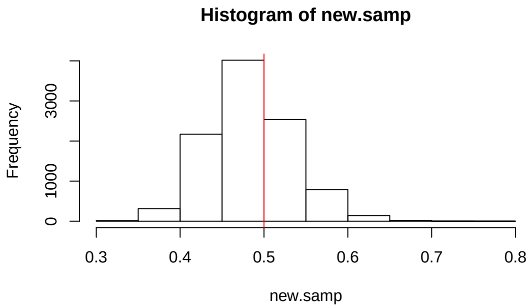


More on transformations

- Note that $\exp(\beta_1) - \beta_1$ is not monotonic, i.e., decreasing from $(-\infty, 0)$, increasing $(0, \infty)$
- By working with the empirical quantiles of the transformed posterior samples, we can capture the skewness in the uncertainty.
- For frequentist: for logistic and Poisson, we back-transformed parameter estimates and their confidence intervals because logit and log are strictly monotonic. Back-transforming quantiles for strictly monotonic functions is valid. This is not okay when the functions are not strictly monotonic.
- For a differentiable transformation (not restricted to monotonic), a standard Frequentist approach utilizes the Delta method and asymptotic normality.
- In Bayesian, we have a sample of the posterior distribution and hence can transform the sample, and then calculate any desired statistics.

Probability that $\beta_1/\beta_0 > 0.5$?

```
> # Test whether the ratio of the slope to intercept is greater than 1/2:
> new.samp = post.samp[,2]/post.samp[,1]
> # Hypothesis test:
> # ratio of beta1/beta0 is greater than 0.5
> mean ( new.samp > 0.5)
[1] 0.3484
```



Bayesian inference uses probability to assess statistical significance of a hypothesis test, instead of a p-value. Here $P(\beta_1/\beta_0 > 0.5)$ is only about 0.35, indicating weak evidence (strong evidence would be close to 1).

Estimate and 95% Interval of $\beta_0 + 2\beta_1 + \epsilon_i$?

For this problem, we wish to obtain a prediction estimate $\hat{y}$ and its prediction interval. Let $x_i = [1, 2]$. Recall from standard regression,

$$\hat{y} \sim N\left(\hat{\beta}_0 + 2\hat{\beta}_1,\ [1,2]\text{Cov}(\hat{\beta}_0, \hat{\beta}_1) \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \hat{\sigma}^2 \right).$$

If we treat $\hat{y}$ as another unknown parameter, the corresponding Bayesian posterior predictive distribution is:

$$\begin{aligned}
[\hat{y}|\mathbf{y}] &= \int [\hat{y}, \beta_0, \beta_1, \sigma^2 \,|\, \mathbf{y}]\ d\beta_0 d\beta_1 d\sigma^2 \\
&= \int [\hat{y} \,|\, \beta_0, \beta_1, \sigma^2, \mathbf{y}][\beta_0, \beta_1, \sigma^2 \,|\, \mathbf{y}]\ d\beta_0 d\beta_1 d\sigma^2 \\
&= \int [\hat{y} \,|\, \beta_0, \beta_1, \sigma^2][\beta_0, \beta_1, \sigma^2 \,|\, \mathbf{y}]\ d\beta_0 d\beta_1 d\sigma^2. \quad (\hat{y} \text{ cond indep of } \mathbf{y})
\end{aligned}$$

Bayesian posterior predictive distribution

So to predict $\hat{y}$, we want to incorporate the uncertainties in $\beta_0, \beta_1, \sigma^2$ by averaging/integrating over the regression parameters.

$$[\hat{y}|\mathbf{y}] = \int [\hat{y} \,|\, \beta_0, \beta_1, \sigma^2][\beta_0, \beta_1, \sigma^2 \,|\, \mathbf{y}] \; d\beta_0 d\beta_1 d\sigma^2.$$

The above can be estimated by Monte Carlo simulations.

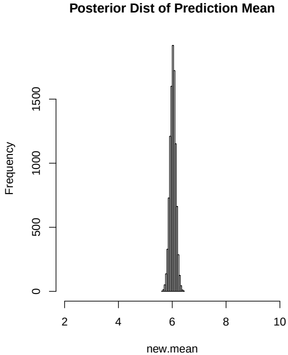
1. Generate $\beta_0^{(k)}$, $\beta_1^{(k)}$, and $\sigma^{2\;(k)}$ from $[\beta_0, \beta_1, \sigma^2 \,|\, \mathbf{y}]$.
2. Generate
$$\hat{y}^{(k)} \sim N(\,\beta_0^{(k)} + 2\beta_1^{(k)},\, \sigma^{2\;(k)}).$$

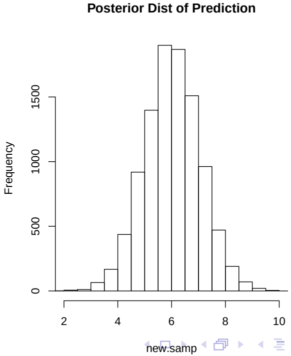
Because $\beta_0^{(k)}$, $\beta_1^{(k)}$, and $\sigma^{2\;(k)}$ are sampled from the joint posterior, they reflect uncertainties in model parameters, which will be propagated in the samples of $\hat{y}^{(k)}$.

This is different from the standard regression analysis where we assume $\hat{\sigma}^2$ is fixed and known.

Estimate and 95% Interval of $\beta_0 + 2\beta_1 + \epsilon_i$?

```
> # Example: consider a subject with x = 2:
> ### Posterior samples of the predictive mean
> new.mean = post.samp[,1] + 2*post.samp[,2]
> ### Generate normal random variables with different standard deviation.
> # NOTE: Here a different sigma is used for every realization:
> new.samp = rnorm (length(new.mean), new.mean, sqrt(post.samp[,3]))
> mean(new.samp)
[1] 6.088105
> quantile(new.samp, c(.025, .975))
    2.5%    97.5%
4.223426 7.994790
```





Effects of More MCMC Samples?

```
> fit = MCMCregress (Y~X, b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01, mcmc = 100000)
> summary(fit)
              Mean      SD  Naive SE Time-series SE
(Intercept) 3.129 0.20833 0.0006588      0.0007221
X           1.447 0.07611 0.0002407      0.0002524
sigma2      1.074 0.15753 0.0004982      0.0005303

              2.5%    25%   50%   75% 97.5%
(Intercept) 2.7203 2.9896 3.129 3.267 3.539
X           1.2965 1.3959 1.447 1.497 1.597
sigma2      0.8092 0.9628 1.060 1.169 1.423
```

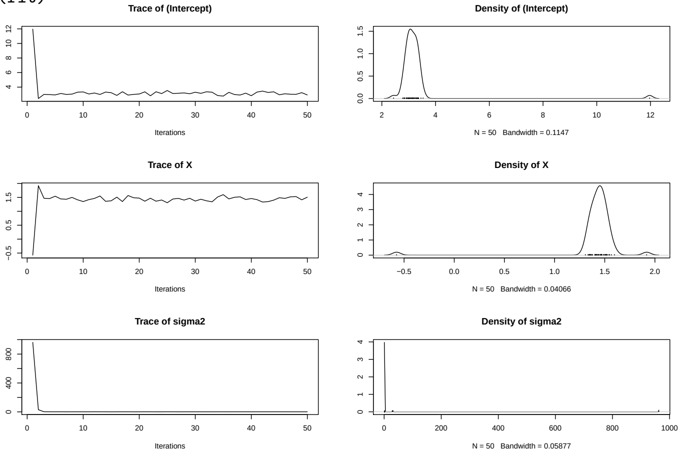
There is error in our estimate of posterior distribution due to approximation using MCMC = Monte Carlo error.

Change the number of MCMC iterations to 100,000

- All point estimates, quantiles, and SD's are close.
- Significantly reduces Naive SE and Time-series SE (reducing Monte Carlo error).
- More samples is always better. But it costs computation time and storage.

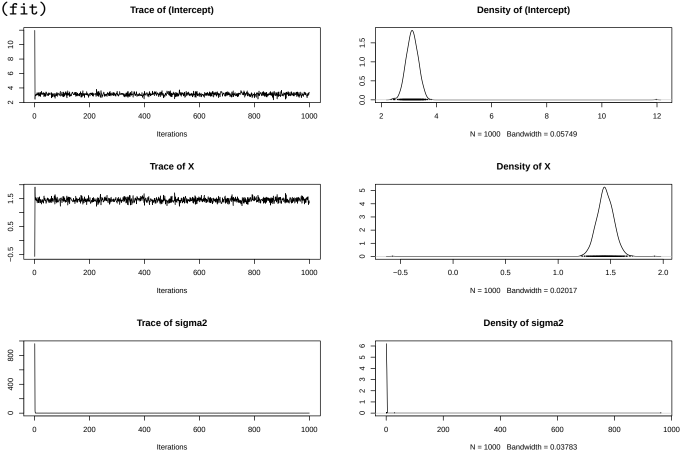
MCMC Chain Convergence: Iterations Too Small

```
### Beta starting values at (10,10)
### Run for 50 iterations with no burn-in
> fit = MCMCregress (Y~X, b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01,
    burnin = 0, mcmc = 50, beta.start = c(10,10) )
> plot (fit)
```



MCMC Chain Convergence: Longer Iterations

```
### Beta starting values at (10,10)
### Run for 1000 iterations with no burn-in
### Non-convergence appears for the first few samples.
> fit = MCMCregress (Y~X, b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01,
    burnin = 0, mcmc = 1000, beta.start = c(10,10) )
> plot (fit)
```



MCMC Chain Convergence: Longer MCMC with Burn-in

```
### Beta starting values at (10,10)
### Run for 1000 iterations with 100 burn-in samples
### Convergence Okay!
> fit = MCMCregress (Y~X, b0 = c(0,0), B0 = 0.001^2, c0=0.01, d0=0.01,
    burnin = 100, mcmc = 1000, beta.start = c(10,10) )
> plot (fit)
```