Assignment 2

Raj Patel

2020-01-31

In this assignment, we look at the following degenerate regression model:

$$y_i|\beta_1, \beta_2, x_i, \sigma \sim N(\beta_1 x_i + \beta_2 x_i, \sigma^2)$$

Question 1:

Here, we want to simulate three datasets (n = 10, 100, 1000) from the model provided with $\beta_1 = 0.2$ and $\beta_2 = 0.8$.

Answer:

To simulate a dataset of n=10 for this model, we first simulate 10 data points from N(3,1) which we would be using as our covariates. Then we use that to generate 10 data points for our first dataset which follows $N(\beta_1 x_i + \beta_2 x_i, \sigma^2)$, which in our case would be, $N(0.2x_i + 0.8x_i, 2^2)$, assuming we consider our variance as 4 for our degenerate regression model.

Similarly, we generate our second and third datasets.

```
beta_1 <- 0.2
beta_2 <- 0.8

# generating covariates for first dataset
cov_1 <- rnorm(10,3,1)

# first dataset for degenerate regression model with n=10
degen_reg_1 <- rnorm(10, beta_1*cov_1 + beta_2*cov_1, 2)

# generating covariates for second dataset
cov_2 <- rnorm(100,3,1)

# second dataset for degenerate regression model with n=100
degen_reg_2 <- rnorm(100, beta_1*cov_2 + beta_2*cov_2, 2)

# generating covariates for third dataset
cov_3 <- rnorm(1000,3,1)

# third dataset for degenerate regression model with n=1000
degen_reg_3 <- rnorm(1000, beta_1*cov_3 + beta_2*cov_3, 2)
```

Question 2:

Here, we want to fit the datasets we created in Stan with $p(\beta) \propto 1$ and describe the output. Following that, we want to use the Rhat and n_eff in our description of what went wrong.

```
data {
```

```
  int<lower=0> N; // number of observations
  vector[N] x; // predictor
  vector[N] y; // response
}
parameters {
  real beta1;
  real beta2;
  real<lower=0> sigma;
}
model {
  y ~ normal(beta1 * x + beta2 * x, sigma);
}
```

```
options(mc.cores = parallel::detectCores())

N = 10;
x = cov_1;
y = degen_reg_1;
stan_data <- list(N=N, x=x, y=y)

fit <- sampling(lin_reg, data = stan_data)
print(fit)
```

```
## Inference for Stan model: 2e8a8300b271b803ca5e632f92d1293c.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean      sd     2.5%      25%     50%     75%   97.5% n_eff
## beta1  989.97  686.16 1401.43  -885.33   107.81  756.28 1404.01 4832.91     4
## beta2 -989.35  686.16 1401.44 -4832.00 -1403.71 -755.62 -107.08  885.70     4
## sigma    2.34    0.16    0.67     1.52     1.91    2.22    2.65    3.69    17
## lp__   -11.23    0.18    1.20   -14.07   -11.66  -10.90  -10.43  -10.11    43
##       Rhat
## beta1 2.27
## beta2 2.27
## sigma 1.17
## lp__  1.08
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:23:15 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
N = 100;
x = cov_2;
y = degen_reg_2;
stan_data_2 <- list(N=N, x=x, y=y)

fit2 <- sampling(lin_reg, data = stan_data_2)
print(fit2)
```

```
## Inference for Stan model: 2e8a8300b271b803ca5e632f92d1293c.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean     sd     2.5%     25%     50%     75%   97.5% n_eff
## beta1 -409.49  609.66 912.35 -2351.03 -835.76 -267.05  255.68  947.08     2
## beta2  410.46  609.66 912.35  -946.17 -254.70  268.01  836.73 2352.02     2
## sigma    2.11    0.02   0.14     1.86    2.02    2.10    2.19    2.42    40
## lp__  -125.34    0.06   0.93  -127.91 -125.70 -125.04 -124.69 -124.42   240
##       Rhat
## beta1 4.48
## beta2 4.48
## sigma 1.08
## lp__  1.02
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:23:26 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
N = 1000;
x = cov_3;
y = degen_reg_3;
stan_data_3 <- list(N=N, x=x, y=y)

fit3 <- sampling(lin_reg, data = stan_data_3)
print(fit3)
```

```
## Inference for Stan model: 2e8a8300b271b803ca5e632f92d1293c.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean     sd     2.5%      25%      50%      75%    97.5%
## beta1    46.38   83.08 137.65  -169.56   -84.82    55.27   158.49   295.29
## beta2   -45.37   83.08 137.65  -294.29  -157.47   -54.25    85.81   170.56
## sigma     2.04    0.01   0.04     1.96     2.01     2.04     2.07     2.13
## lp__  -1211.98    0.06   0.92 -1214.38 -1212.39 -1211.69 -1211.29 -1211.05
##       n_eff Rhat
## beta1     3 2.48
## beta2     3 2.48
## sigma    29 1.22
## lp__    261 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:24:41 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Based on the output for all three models (n=10,100,1000), we observe that Rhat for β_1 and β_2 is not close to 1. In fact, it is way too off from 1, as for all the cases it is more than 2 (for n=10,100,1000). As it is this big, we can say that the chains have not mixed well as Rhat statistic quantifies the consistency of an ensemble of Markov Chains. In practice, generally, we only use samples if Rhat is less than 1.05. With Rhat this big, we can not use the sample as the chains .

Moreover, besides Rhat, if we look at n_eff, which is effective sample size taking serial correlation in chains into account, for all three cases, it is quite less. So, this also suggests that the samples generated are not good enough.

Question 3:

Here, we want to fit the datasets we created in Stan with priors $\beta_j \sim N(0,1)$ for all three datasets and describe the output and difference as compared to the previous one.

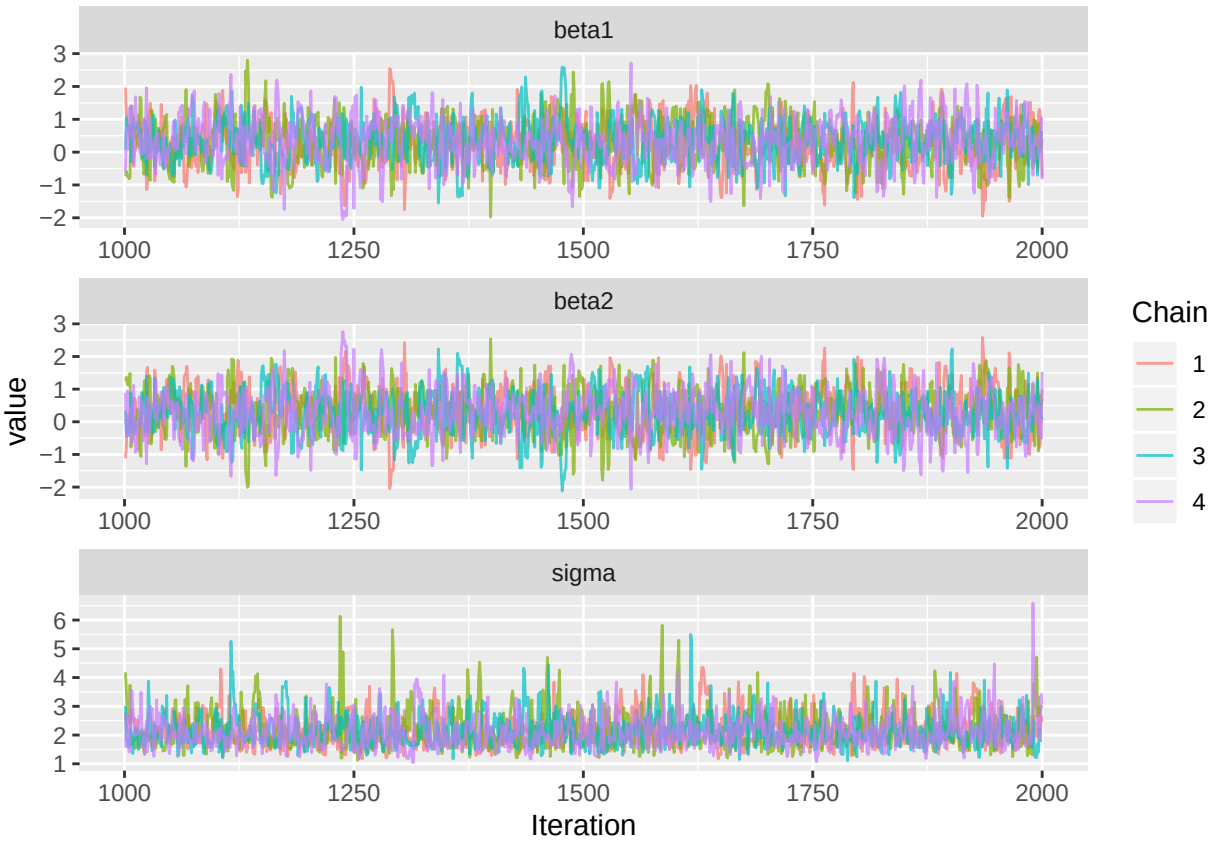
Following that, we want to see if it converges to true values for β_1 and β_2. If not, we want to see what converges to true value and what is the role of prior

```
data {
  int<lower=0> N; // number of observations
  vector[N] x; // predictor
  vector[N] y; // response
}
parameters {
  real beta1;
  real beta2;
  real<lower=0> sigma;
}
model {
  y ~ normal(beta1 * x + beta2 * x, sigma);
  beta1 ~ normal(0,1);
  beta2 ~ normal(0,1);
}
```

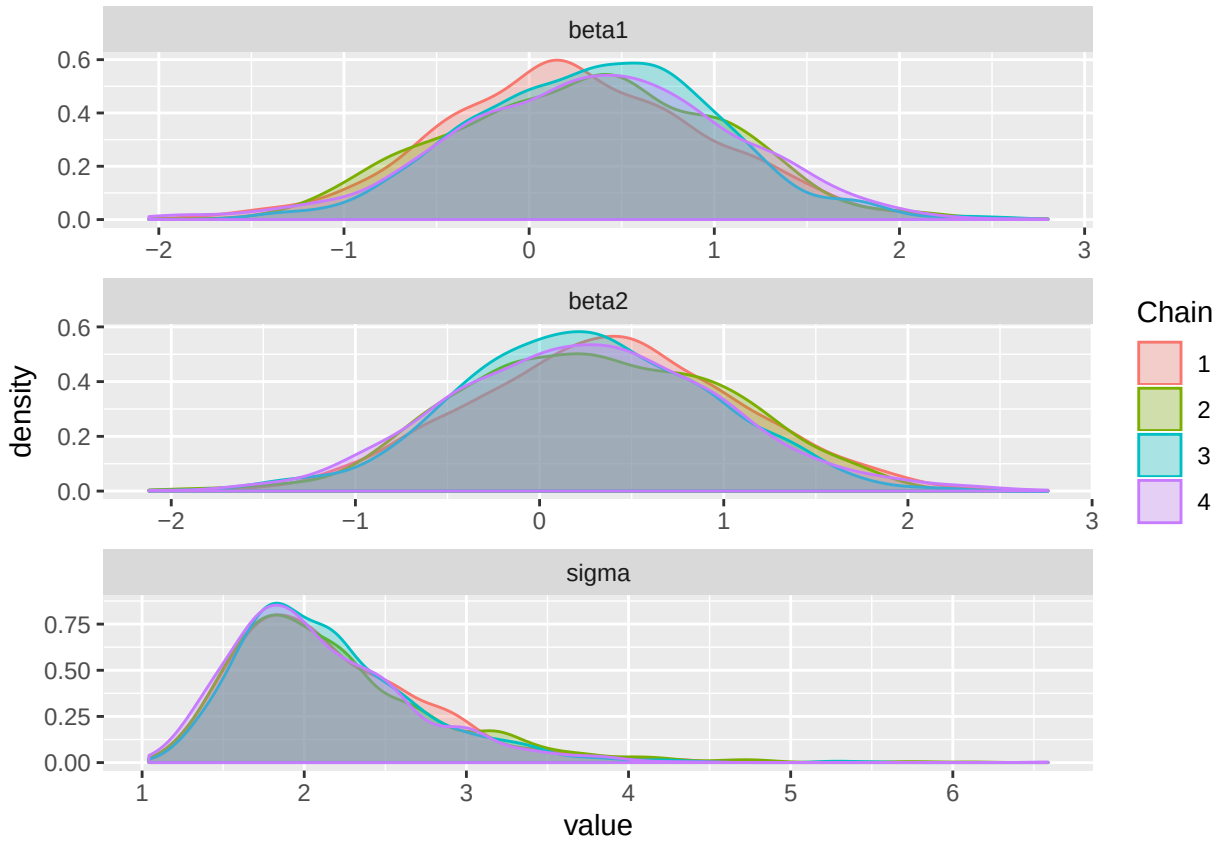
```
options(mc.cores = parallel::detectCores())

N = 10;
x = cov_1;
y = degen_reg_1;
stan_data <- list(N=N, x=x, y=y)

fit <- sampling(lin_reg2, data = stan_data)
print(fit)
```

```
## Inference for Stan model: 5b299a01656171f142b9c83df06edaaf.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta1   0.32    0.02 0.71  -1.07  -0.17   0.32   0.80   1.68  1255 1.00
## beta2   0.29    0.02 0.71  -1.05  -0.20   0.29   0.78   1.68  1279 1.00
## sigma   2.15    0.02 0.59   1.33   1.73   2.03   2.46   3.56  1474 1.01
## lp__  -11.73    0.04 1.24 -14.92 -12.33 -11.40 -10.82 -10.28  1132 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:25:19 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# plotting traceplot of all chains
ggs_traceplot(ggs(fit))
```



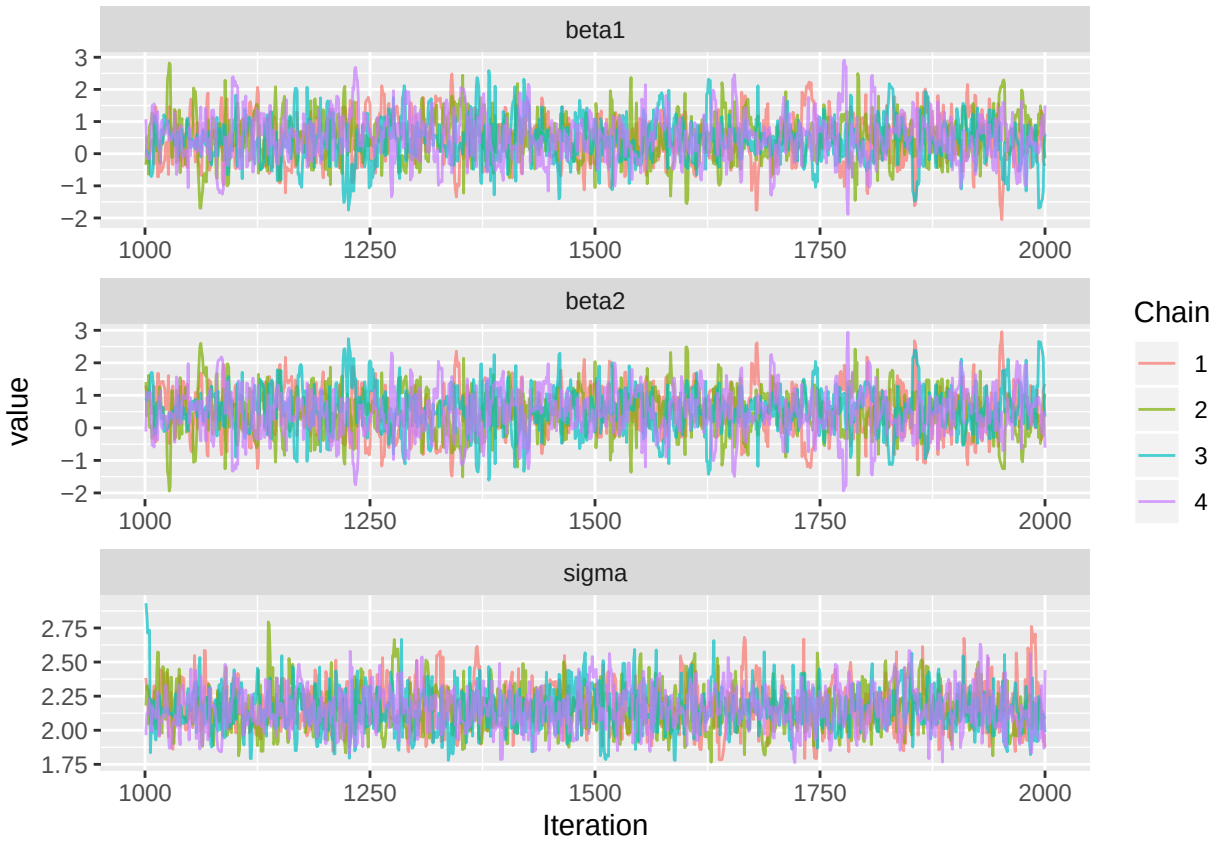

```
ggs_density(ggs(fit))
```



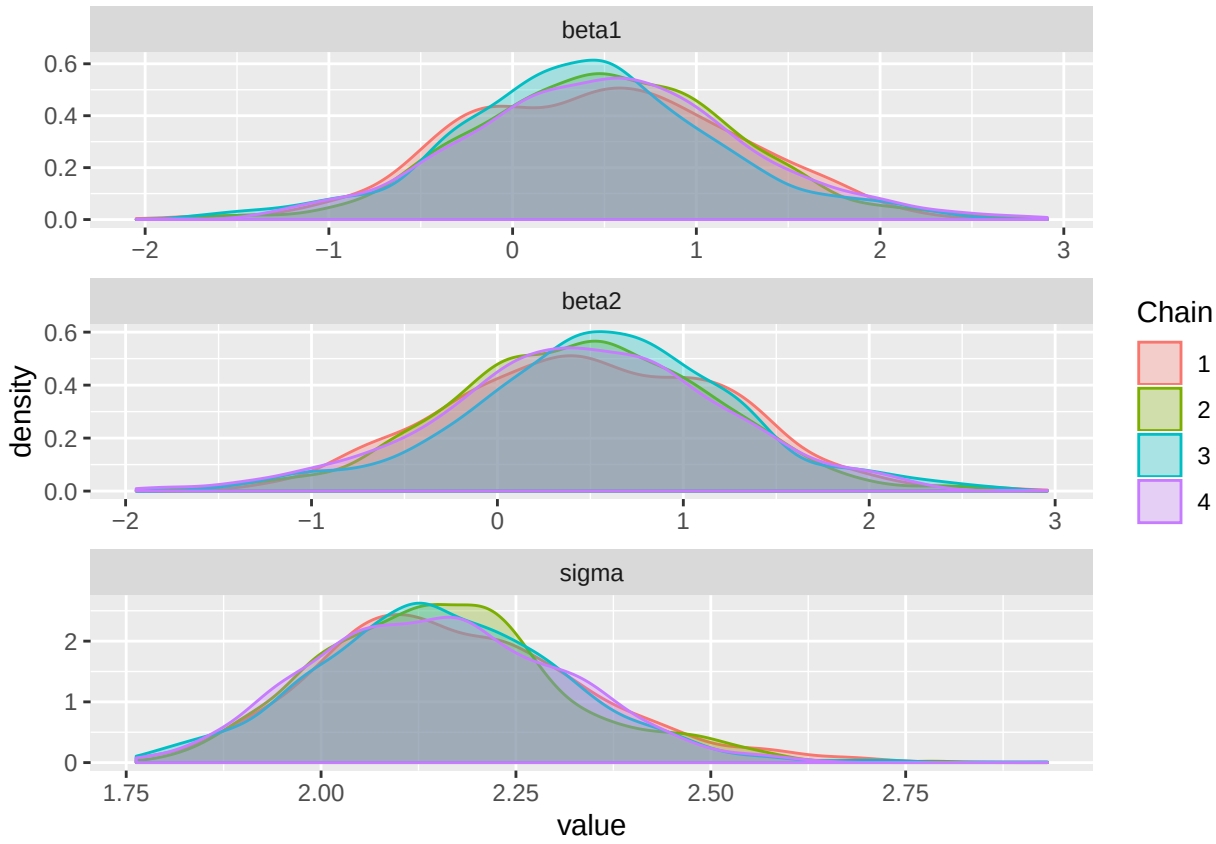

```
N = 100;
x = cov_2;
y = degen_reg_2;
stan_data_2 <- list(N=N, x=x, y=y)

fit2 <- sampling(lin_reg2, data = stan_data_2)
print(fit2)
```

```
## Inference for Stan model: 5b299a01656171f142b9c83df06edaaf.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## beta1    0.47    0.02 0.72   -0.95   -0.02    0.47    0.94    1.94  1071    1
## beta2    0.50    0.02 0.72   -0.99    0.02    0.50    0.99    1.95  1076    1
## sigma    2.16    0.00 0.16    1.87    2.05    2.15    2.26    2.50  1582    1
## lp__  -126.20    0.04 1.26 -129.47 -126.81 -125.85 -125.27 -124.74  1184    1
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:25:20 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# plotting traceplot of all chains
ggs_traceplot(ggs(fit2))
```



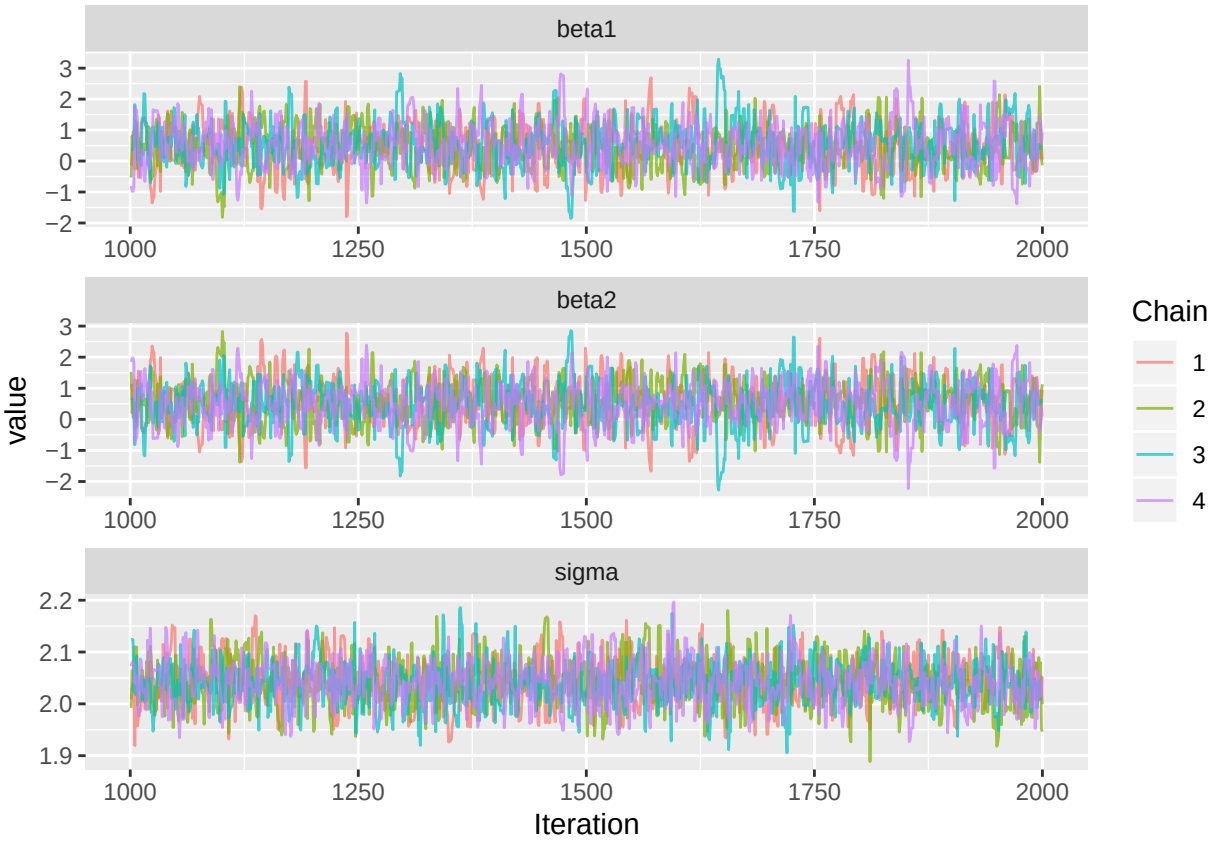

```
ggs_density(ggs(fit2))
```



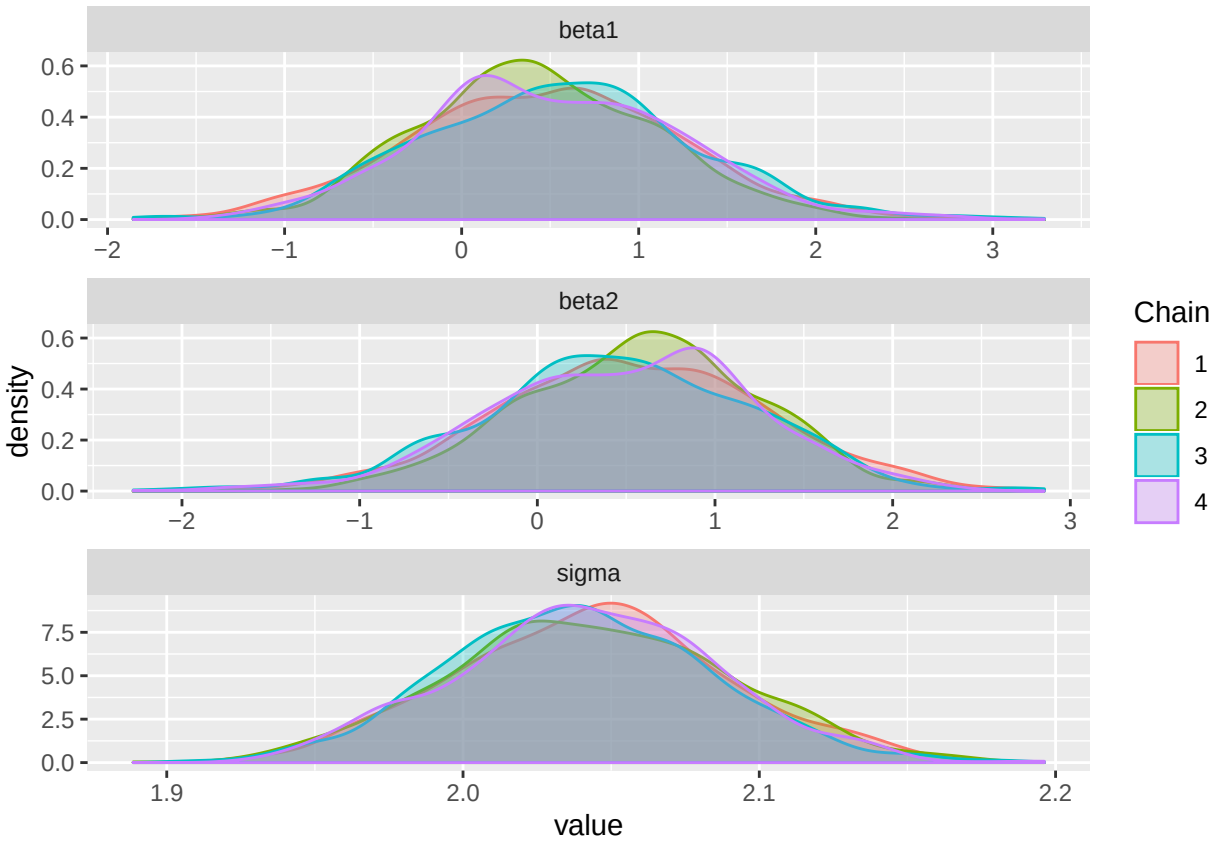

```
N = 1000;
x = cov_3;
y = degen_reg_3;
stan_data_3 <- list(N=N, x=x, y=y)

fit3 <- sampling(lin_reg2, data = stan_data_3)
print(fit3)
```

```
## Inference for Stan model: 5b299a01656171f142b9c83df06edaaf.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff
## beta1     0.49    0.02 0.73    -0.89     0.00     0.47     0.98     1.93  1189
## beta2     0.52    0.02 0.73    -0.93     0.03     0.54     1.01     1.91  1191
## sigma     2.04    0.00 0.04     1.95     2.01     2.04     2.07     2.13  1708
## lp__  -1212.79    0.04 1.24 -1216.03 -1213.35 -1212.47 -1211.90 -1211.40  1067
##       Rhat
## beta1    1
## beta2    1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Tue Feb  4 00:25:32 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
# plotting traceplot of all chains
ggs_traceplot(ggs(fit3))
```




```r
ggs_density(ggs(fit3))
```



Now, if we look at the outputs here, it has improved significantly as far as convergence and sampling is concerned. Rhat for all the three cases is now 1 for β_1 and β_2. This tells us that for all these cases, the chains have converged. Moreover, with n_eff being greater than 1000, we can say that the draws that Stan is producing are better than independent draws with those parameters.

However, we see that despite this, the mean of β_1 and β_2 still do not converge to the true value of β_1 and β_2 respectively. Though the mean does not converge, but we observe that as the sample size increases, the standard deviation converges.

Here, the role of the prior is to restrict the posterior from outputting massive values. With this prior, the posterior outputs values that are close to 0 such that $\beta_1 + \beta_2 \approx 1$