Rstan Practice

weiya

March 6, 2019

```
set.seed(1234)
N = 250
K = 3
covariates = replicate(K, rnorm(n=N))
colnames(covariates) = c('X1', 'X2', 'X3')
X = cbind(Intercept=1, covariates)

coefs = c(5, .2, -1.5, .9)
mu = X %*% coefs
sigma = 2
y = rnorm(N, mu, sigma)

modlm = lm(y~., data = data.frame(X[,-1]))
summary(modlm)
```

```
##
## Call:
## lm(formula = y ~ ., data = data.frame(X[, -1]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6160 -1.2391 -0.0206  1.2228  5.3990
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.9362     0.1236  39.940  < 2e-16 ***
## X1            0.3211     0.1214   2.645  0.00869 **
## X2           -1.2710     0.1173 -10.833  < 2e-16 ***
## X3            0.9885     0.1317   7.508  1.1e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.948 on 246 degrees of freedom
## Multiple R-squared:  0.4139, Adjusted R-squared:  0.4067
## F-statistic:  57.9 on 3 and 246 DF,  p-value: < 2.2e-16
```

Time for Stan.

```
# create the data list object for stan input
dat = list(N = N, K = ncol(X), y=y, X=X)
# create the stan model object using Stan's syntax
stanmodelcode = "
data {
  int<lower=1> N;
  int<lower=1> K;
  matrix[N, K] X;
  vector[N] y;
}
```

```
parameters {
  vector[K] beta;
  real<lower=0> sigma;
}
model {
  vector[N] mu;
  mu = X * beta;
  // priors
  beta ~ normal(0, 10);
  sigma ~ cauchy(0, 5);
  // likelihood
  y ~ normal(mu, sigma);
}
"
```

Run the stan model

```
library(rstan)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.18.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
### run the model and examine results
fit = stan(model_code = stanmodelcode,
           data = dat,
           iter = 5000,
           warmup = 2500,
           thin = 10,
           chains = 4)
```

```
##
## SAMPLING FOR MODEL '7b080505fdc1ab8b066e2ee81db2afab' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.35 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
```

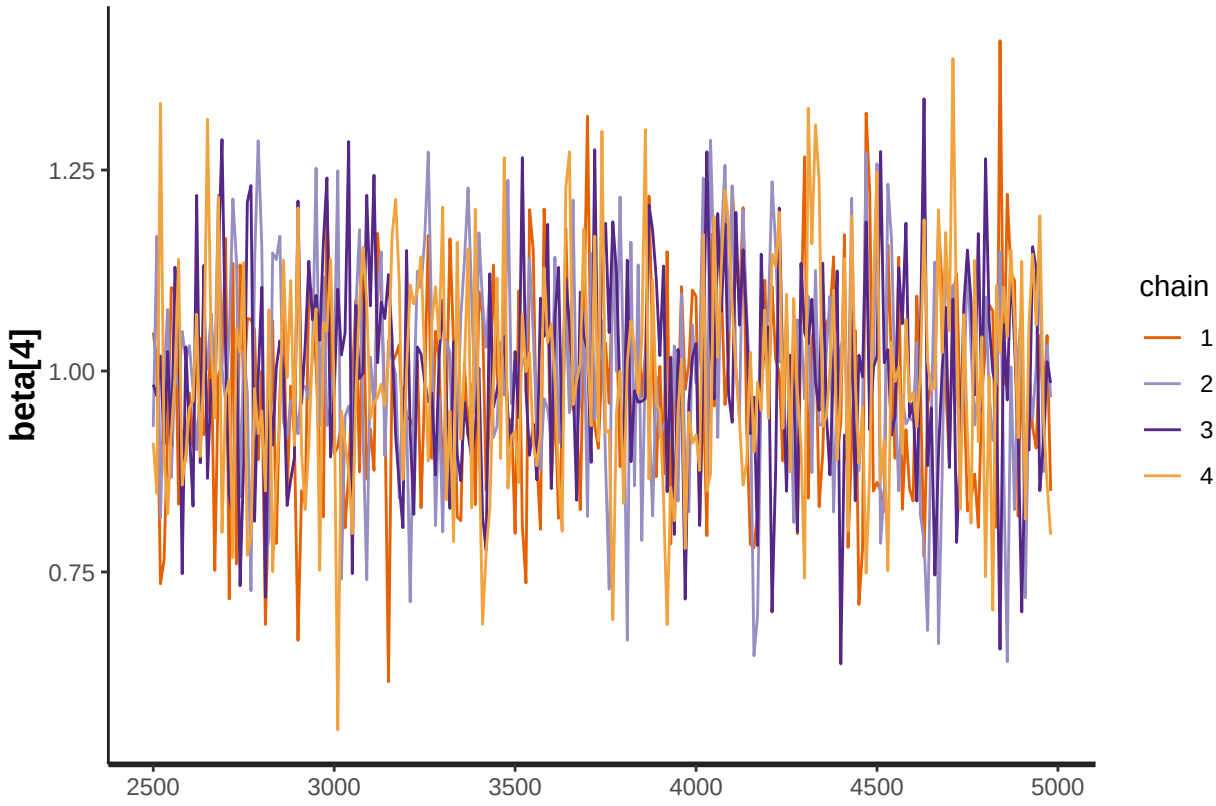
```
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.231078 seconds (Warm-up)
## Chain 1:                0.194629 seconds (Sampling)
## Chain 1:                0.425707 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '7b080505fdc1ab8b066e2ee81db2afab' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.38 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 2: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 2: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 2: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 2: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 2: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 2: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 2: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 2: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.177892 seconds (Warm-up)
## Chain 2:                0.329345 seconds (Sampling)
## Chain 2:                0.507237 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '7b080505fdc1ab8b066e2ee81db2afab' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1.3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 3: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 3: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 3: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 3: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 3: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 3: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 3: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 3: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.187076 seconds (Warm-up)
## Chain 3:                0.210964 seconds (Sampling)
```

```
## Chain 3:                0.39804 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '7b080505fdc1ab8b066e2ee81db2afab' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 4: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 4: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 4: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2500 / 5000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2501 / 5000 [ 50%]  (Sampling)
## Chain 4: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 4: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 4: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 4: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 4: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.184386 seconds (Warm-up)
## Chain 4:                0.185156 seconds (Sampling)
## Chain 4:                0.369542 seconds (Total)
## Chain 4:
```

```
# summary
print(fit, pars = c('beta', 'sigma'), digits = 3, probs = c(.025, .5, .975))
```

```
## Inference for Stan model: 7b080505fdc1ab8b066e2ee81db2afab.
## 4 chains, each with iter=5000; warmup=2500; thin=10;
## post-warmup draws per chain=250, total post-warmup draws=1000.
##
##           mean se_mean    sd   2.5%    50%  97.5% n_eff  Rhat
## beta[1]  4.941   0.004 0.126  4.697  4.943  5.197  1191 0.998
## beta[2]  0.322   0.004 0.120  0.101  0.324  0.574  1032 0.998
## beta[3] -1.269   0.004 0.117 -1.496 -1.272 -1.047   855 0.998
## beta[4]  0.991   0.004 0.134  0.728  0.988  1.258   976 1.002
## sigma    1.954   0.003 0.089  1.787  1.949  2.136   965 0.998
##
## Samples were drawn using NUTS(diag_e) at Wed Mar  6 20:50:44 2019.
## 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).
```

```
# visualize
stan_trace(fit, pars = c('beta[4]'))
```



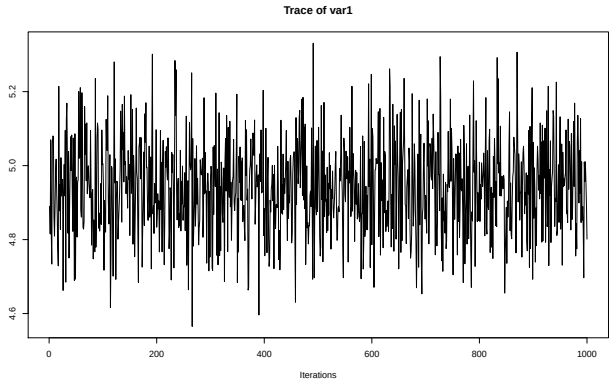
Diagnostics with `coda` package:

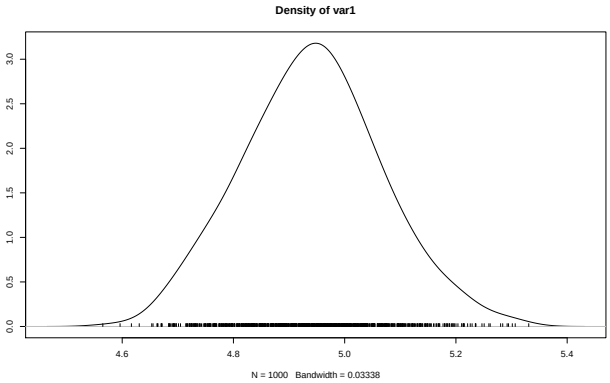
```
library(coda)
```

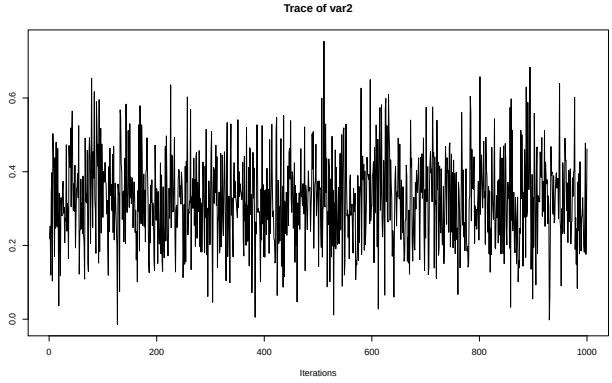
```
##
## Attaching package: 'coda'
```

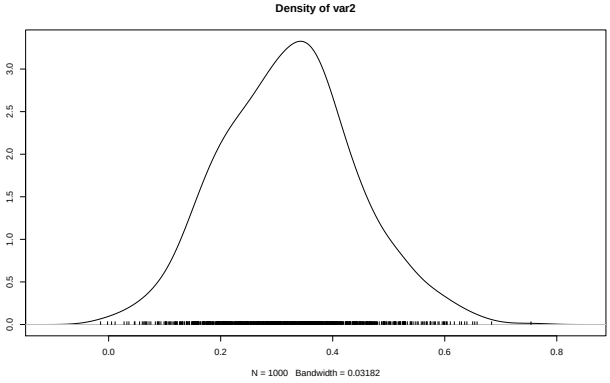
```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

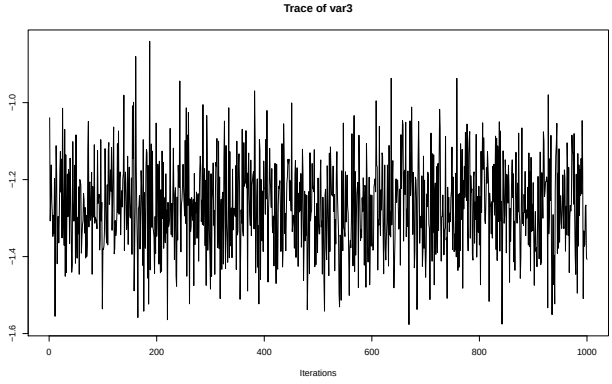
```
betas = extract(fit, pars='beta')$beta
betas.mcmc = as.mcmc(betas)
plot(betas.mcmc)
```

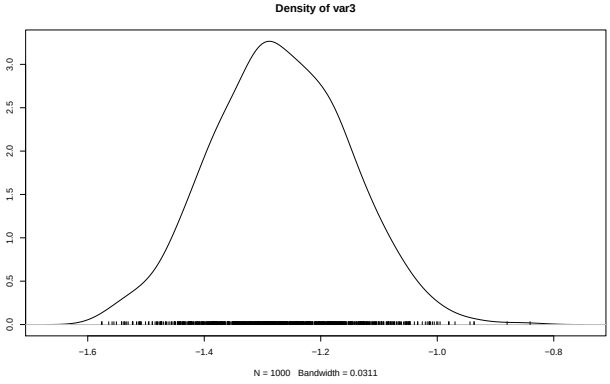

Trace of var1
Iterations

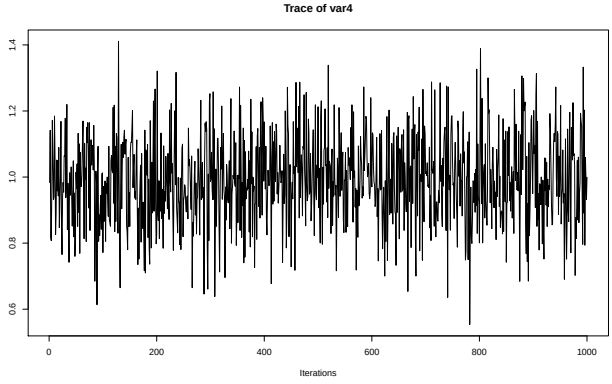

Density of var1
N = 1000 Bandwidth = 0.03338

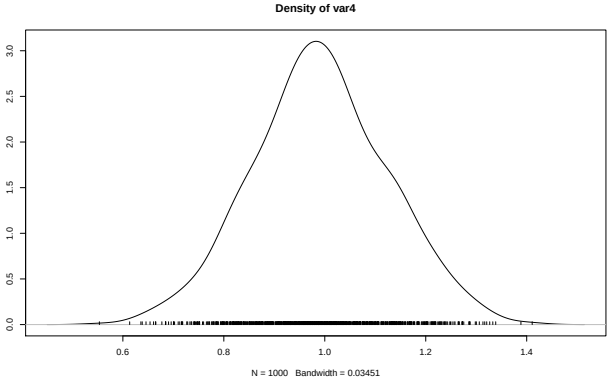

Trace of var2
Iterations


Density of var2
N = 1000 Bandwidth = 0.03182


Trace of var3
Iterations


Density of var3
N = 1000 Bandwidth = 0.0311


Trace of var4
Iterations


Density of var4
N = 1000 Bandwidth = 0.03451

Reference

Rstan: Regression Models