Learning Stan

Seun Odeyemi

2019-06-08

Contents

Load Libraries

```
library(kableExtra)
library(dplyr)
```

In this Stan program, we let theta be a transformation of mu, eta, and tau instead of declaring theta in the parameters block, which allows the sampler will run more efficiently (see detailed explanation). We can prepare the data (which typically is a named list) in R with:

```
schools_dat <- list(J = 8,
                    y = c(28,  8, -3,  7, -1,  1, 18, 12),
                    sigma = c(15, 10, 16, 11,  9, 11, 10, 18))
```

And we can get a fit with the following R command. Note that the argument to file = should point to where the file is on your file system unless you have put it in the working directory of R in which case the below will work.

```
library(rstan)
# 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)

# stan(..., control = list(adapt_delta = 0.99))

fit <- stan(file = '8schools.stan', data = schools_dat)
```

```
## Warning: There were 5 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

The object fit, returned from function stan is an `S4` object of class `stanfit`. Methods such as print, plot, and pairs are associated with the fitted result so we can use the following code to check out the results in fit. print provides a summary for the parameter of the model as well as the log-posterior with name `lp__` (see the following example output). For more methods and details of class `stanfit`, see the help of class `stanfit`.

In particular, we can use the extract function on `stanfit` objects to obtain the samples. extract extracts samples from the `stanfit` object as a list of arrays for parameters of interest, or just an array. In addition, S3 functions `as.array`, `as.matrix`, and `as.data.frame` are defined for `stanfit` objects (using `help("as.array.stanfit")` to check out the help document in R).

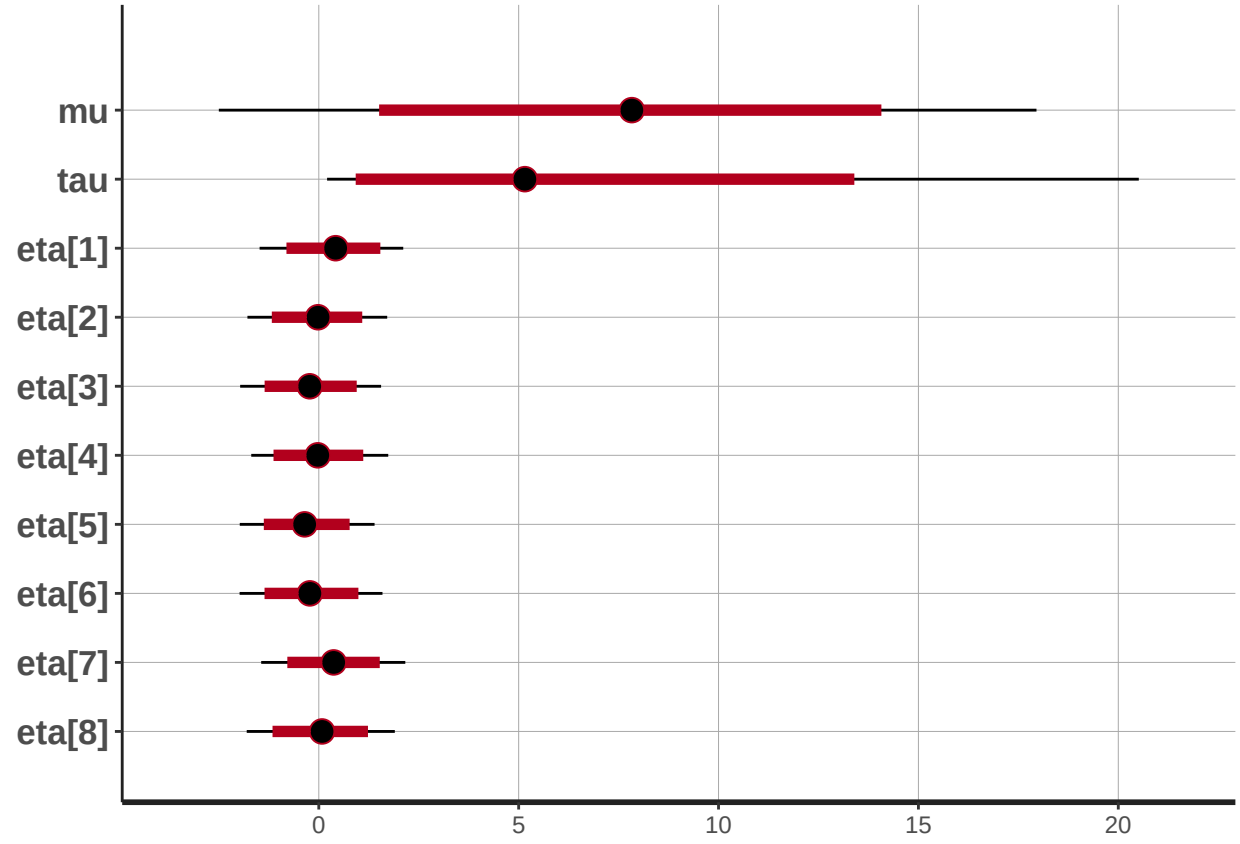
```
# print(fit)

# fit2 <- as.data.frame(fit)
```

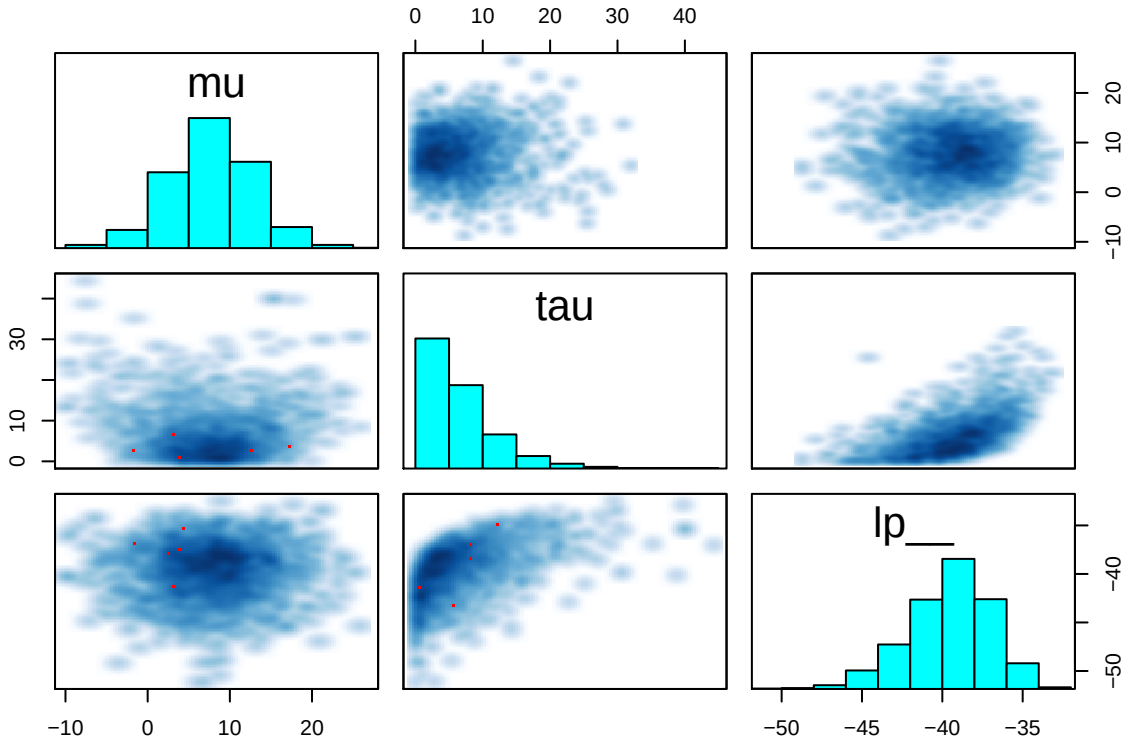
```
#
# kable(fit2[1:10, 1:5]) %>%
#   kable_styling(bootstrap_options = "striped", full_width = T)

plot(fit)
```

```
## 'pars' not specified. Showing first 10 parameters by default.
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```




```
pairs(fit, pars = c("mu", "tau", "lp__"))
```




```
la <- extract(fit, permuted = TRUE) # return a list of arrays
mu <- la$mu

### return an array of three dimensions: iterations, chains, parameters
a <- extract(fit, permuted = FALSE)

### use S3 functions on stanfit objects
a2 <- as.array(fit)
m <- as.matrix(fit)
d <- as.data.frame(fit)

kable(d[1:10,1:10]) %>%
  kable_styling(bootstrap_options = "striped", full_width = T)
```

mu	tau	eta[1]	eta[2]	eta[3]	eta[4]	eta[5]	eta[6]	eta[7]	eta[8]
4.062029	7.415096	0.3741647	-0.6497821	-1.7048068	0.8644842	0.0266430	0.6834515	-0.8127686	1.3322859
11.160474	5.548925	0.8583508	1.1481532	-0.1533071	0.6550470	-0.2154121	-0.9305865	2.2523395	0.3256498
4.960711	6.393589	0.8980397	1.2602192	0.0982359	-0.9367885	0.5579721	-0.4904987	0.6020941	1.1168740
13.517390	4.008448	-0.0865589	-0.3991567	-0.6225168	-1.3929465	-1.8339205	0.1349290	0.5092926	-1.2091880
6.731566	7.505072	1.6622513	-0.8667096	1.1677195	0.1304088	-0.3450378	-0.0503017	-0.3551435	0.5901903
3.615423	3.464558	0.8991595	-1.0125564	1.6024724	-0.8159319	1.2439974	0.2074838	1.6126933	-0.9470590
11.372425	1.954545	-1.1765249	-1.0638034	-0.4762448	0.6681632	-0.3879305	1.3450456	-0.0992689	0.5362146
3.497657	4.517196	1.0853800	-0.2133642	-0.5447562	0.3906798	0.3709011	-0.7923029	0.9530453	1.2621673
9.235015	4.696925	0.2498281	0.6996509	0.3769027	0.1905113	0.0584545	-0.0143022	-0.7045306	-0.6806086
6.396091	5.225450	2.5788658	2.0116266	0.6477133	-0.4933652	0.1865721	-0.6995097	-0.0916233	-0.3970480