Problem2_14.13

```
library(rstan)
```

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

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

```
## rstan (Version 2.19.2, 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)
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.7.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(rstanarm)
```

```
## Loading required package: Rcpp
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## rstanarm (Version 2.19.2, packaged: 2019-10-01 20:20:33 UTC)
```

```
## - Do not expect the default priors to remain the same in future rstanarm versions.
## Thus, R scripts should specify priors explicitly, even if they are just the defaults.
```

```
## - For execution on a local, multicore CPU with excess RAM we recommend calling
```

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

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
##
## Attaching package: 'rstanarm'
```

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

Baysian Linear Regression

1. Data

```
X1 <- log(c(31.2, 24.0, 19.8, 18.2, 9.6, 6.5, 3.2)) #Body Mass
X2 <- log(c(10750, 8805, 7500, 7662, 5286, 3724, 2423)) #Body Surface
Y <- log(c(1113, 982, 908, 842, 626, 430, 281)) #Metabolic Rate
stan_data <- list(x1 = X1, x2 = X2, N = length(Y), y = Y)
```

2. Bayesian Linear Model

Assume data $y \sim N(\alpha + \beta X, \sigma^2)$ with prior $\alpha \sim N(0, 10^2)$, $\beta \sim N(0, 10^2)$ and $\sigma \sim Cauchy(0, 2.5)$. Please write stan file and name it as linear_model.stan.

And then fit your stan model with iteration 1000 times and 4 chains.

```
stan_model <- "~/Documents/BU_2019_Fall/HW6/linear_model_4_13.stan"
fit <- stan(file = stan_model, data = stan_data, iter = 1000, chains = 4, control=list(a
dapt_delta=0.99,max_treedepth = 12))
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info: Comments beginning with # are deprecated.  Please use // in place of # for line
comments.
##
##
## SAMPLING FOR MODEL 'linear_model_4_13' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seco
nds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.84275 seconds (Warm-up)
## Chain 1:                2.66092 seconds (Sampling)
## Chain 1:                4.50367 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'linear_model_4_13' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seco
nds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1.76222 seconds (Warm-up)
## Chain 2:                1.94251 seconds (Sampling)
## Chain 2:                3.70474 seconds (Total)
## Chain 2: 
## 
## SAMPLING FOR MODEL 'linear_model_4_13' NOW (CHAIN 3).
## Chain 3: 
## Chain 3: Gradient evaluation took 7e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seco
nds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3: 
## Chain 3: 
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3: 
## Chain 3:  Elapsed Time: 1.6537 seconds (Warm-up)
## Chain 3:                1.91889 seconds (Sampling)
## Chain 3:                3.5726 seconds (Total)
## Chain 3: 
## 
## SAMPLING FOR MODEL 'linear_model_4_13' NOW (CHAIN 4).
## Chain 4: 
## Chain 4: Gradient evaluation took 7e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seco
nds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4: 
## Chain 4: 
## Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4: 
## Chain 4:  Elapsed Time: 2.69719 seconds (Warm-up)
## Chain 4:                1.69641 seconds (Sampling)
## Chain 4:                4.3936 seconds (Total)
## Chain 4:
```

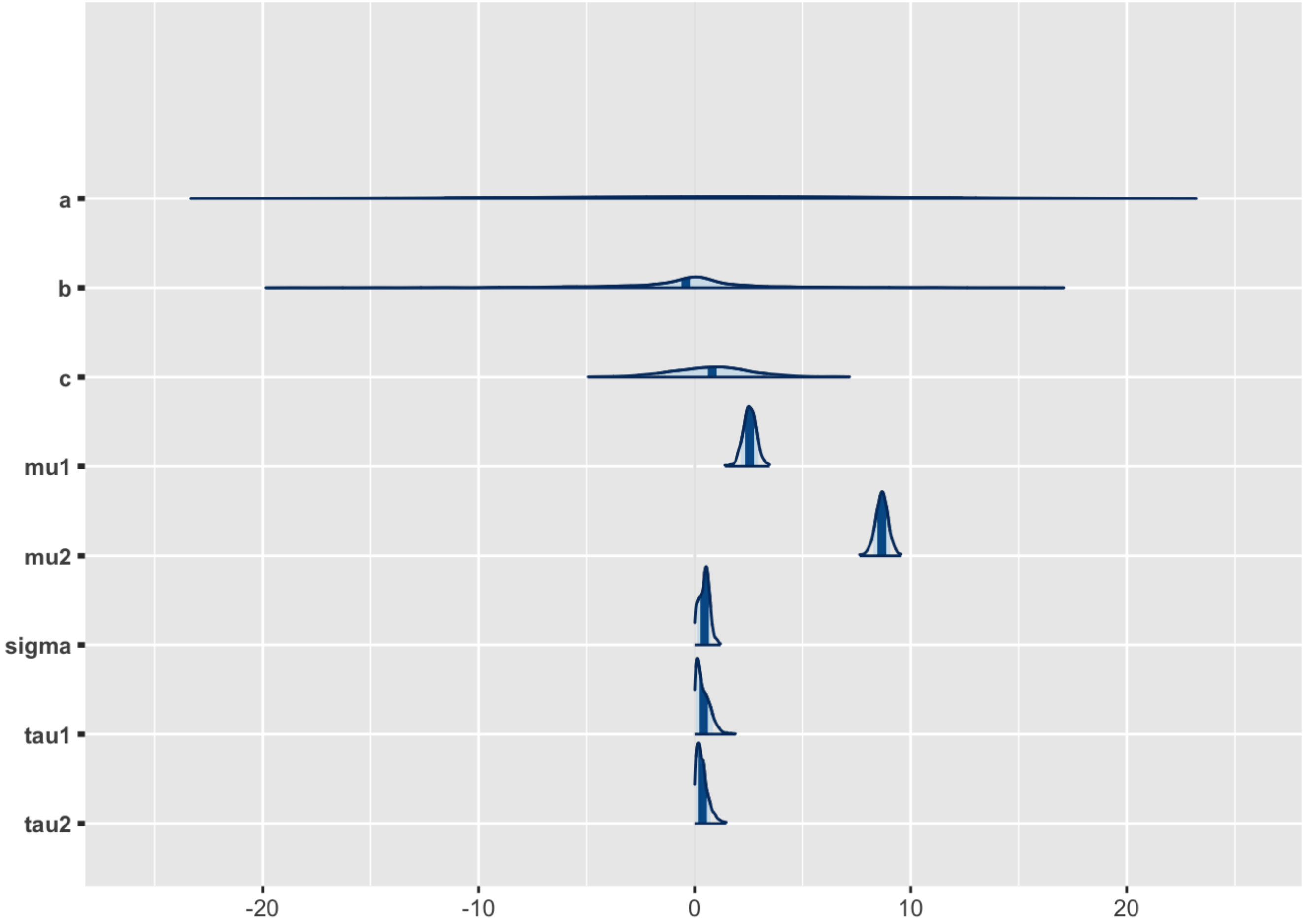
```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and
medians may be unreliable.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances
and tail quantiles may be unreliable.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#tail-ess
```

```
print(fit)
```

```
## Inference for Stan model: linear_model_4_13.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##        mean se_mean   sd   2.5%   25%   50%   75% 97.5% n_eff Rhat
## a      0.55    0.31 9.34 -18.90 -5.56  1.01  6.92 17.91   898 1.00
## b     -0.42    0.28 5.24 -12.83 -1.80 -0.08  1.11 11.27   355 1.03
## c      0.81    0.10 2.08  -3.29 -0.47  0.83  2.03  5.09   454 1.03
## mu1    2.54    0.01 0.33   1.89  2.35  2.55  2.75  3.19   809 1.00
## mu2    8.66    0.01 0.32   8.02  8.46  8.66  8.86  9.27  1166 1.00
## sigma  0.45    0.01 0.25   0.03  0.26  0.47  0.61  0.96   404 1.00
## tau1   0.41    0.02 0.35   0.02  0.13  0.32  0.60  1.19   397 1.00
## tau2   0.36    0.01 0.28   0.01  0.15  0.30  0.50  1.08   679 1.00
## lp__  -7.45    0.14 2.59 -13.44 -8.86 -7.20 -5.55 -3.51   352 1.00
##
## Samples were drawn using NUTS(diag_e) at Tue Nov 26 11:10:15 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).
```

```
posterior <- extract(fit, permuted = FALSE)
mcmc_areas(
  posterior,
  pars = c("a","b","c","mu1","mu2","sigma","tau1","tau2"),
  prob = 0.8, # 80% intervals
  prob_outer = 0.99, # 99%
  point_est = "mean"
)
```


a
b
c
mu1
mu2
sigma
tau1
tau2
-20
-10
0
10
20