# Problem2_14.12

```
## 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)
```

```
## 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
```

```
## 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

```
X <- log(c(31.2, 24.0, 19.8, 18.2, 9.6, 6.5, 3.2)) #Body Mass
Y <- log(c(1113, 982, 908, 842, 626, 430, 281)) #Metabolic Rate
stan_data <- list(x = X, N = length(Y), y = Y)
```

## 2. Bayesian Linear Model

fit stan model with iteration 1000 times and 4 chains.

```
stan_model <- "~/Documents/BU_2019_Fall/HW6/linear_model.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' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.34 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: 0.909552 seconds (Warm-up)
## Chain 1:                0.7606 seconds (Sampling)
## Chain 1:                1.67015 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'linear_model' NOW (CHAIN 2).
## Chain 2: Rejecting initial value:
## Chain 2:   Error evaluating the log probability at the initial value.
## Chain 2: Exception: normal_lpdf: Scale parameter is -3.55319, but must be > 0!  (in
'model12c91442e212_linear_model' at line 42)
##
## Chain 2:
## Chain 2: Gradient evaluation took 1.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 0.898299 seconds (Warm-up)
## Chain 2:                0.906538 seconds (Sampling)
## Chain 2:                1.80484 seconds (Total)
## Chain 2: 
## 
## SAMPLING FOR MODEL 'linear_model' NOW (CHAIN 3).
## Chain 3: 
## Chain 3: Gradient evaluation took 5e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## 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: 0.207983 seconds (Warm-up)
## Chain 3:                0.217939 seconds (Sampling)
## Chain 3:                0.425922 seconds (Total)
## Chain 3: 
## 
## SAMPLING FOR MODEL 'linear_model' NOW (CHAIN 4).
## Chain 4: 
## Chain 4: Gradient evaluation took 5e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.05 seconds.
## 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: 0.350326 seconds (Warm-up)
## Chain 4:                0.287811 seconds (Sampling)
## Chain 4:                0.638137 seconds (Total)
## Chain 4:
```

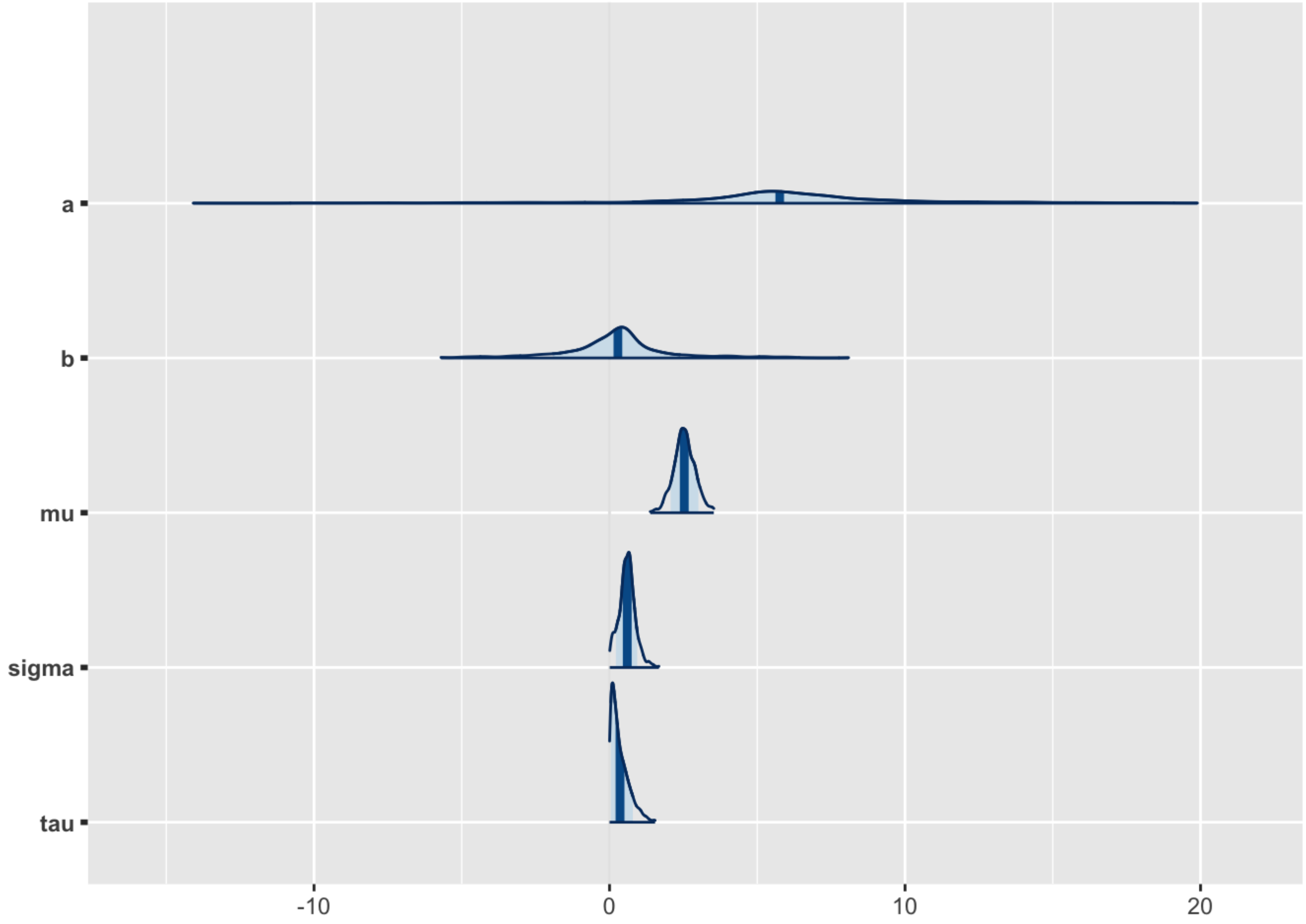
```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and
medians may be unreliable.
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## 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 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      5.76    0.27 5.05 -7.20  4.14  5.80  7.77 16.33   350 1.01
## b      0.28    0.11 2.00 -4.18 -0.52  0.29  0.92  5.26   350 1.01
## mu     2.53    0.02 0.38  1.79  2.30  2.52  2.77  3.30   402 1.01
## sigma  0.60    0.01 0.29  0.07  0.42  0.60  0.75  1.24   500 1.01
## tau    0.35    0.02 0.32  0.01  0.11  0.26  0.52  1.15   359 1.00
## lp__  -5.00    0.10 1.85 -9.19 -6.06 -4.69 -3.60 -2.38   322 1.01
##
## Samples were drawn using NUTS(diag_e) at Tue Nov 26 10:46:06 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","mu","sigma","tau"),
  prob = 0.8, # 80% intervals
  prob_outer = 0.99, # 99%
  point_est = "mean"
)
```


a
b
mu
sigma
tau
-10
0
10
20