STATS 205P HW2

Chuqi Wang 79167724

2024-05-11

Q1:

```
getwd()
```

```
## [1] "/Users/chuqiwang/Desktop/UCI/STATS205P/hw2"
```

```
setwd("/Users/chuqiwang/Desktop/UCI/STATS205P/hw2")
ratweight = read.table("RatWeight.txt", header = TRUE)
```

```
J = length(unique(ratweight$Rat))  # number of rats
N = nrow(ratweight)                # number of observations
y = ratweight$weight               # All Observations weights
rat = as.numeric(factor(ratweight$Rat))
rat_stan = list(
  J = J,
  N = N,
  y = y,
  rat = rat
)
```

```
library(rstan)
```

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

```
##
## rstan version 2.32.6 (Stan version 2.32.2)
```

```
## 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)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
fit <- stan(file = 'ratweight.stan', data = rat_stan)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
```

```
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.251 seconds (Warm-up)
## Chain 1:                0.084 seconds (Sampling)
## Chain 1:                0.335 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.707 seconds (Warm-up)
## Chain 2:                0.076 seconds (Sampling)
## Chain 2:                0.783 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 9e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.622 seconds (Warm-up)
## Chain 3:                1.234 seconds (Sampling)
## Chain 3:                3.856 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 1.4e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.38 seconds (Warm-up)
## Chain 4:                0.093 seconds (Sampling)
## Chain 4:                0.473 seconds (Total)
## Chain 4:
```

```
print(fit, probs = c(0.025, 0.975))
```

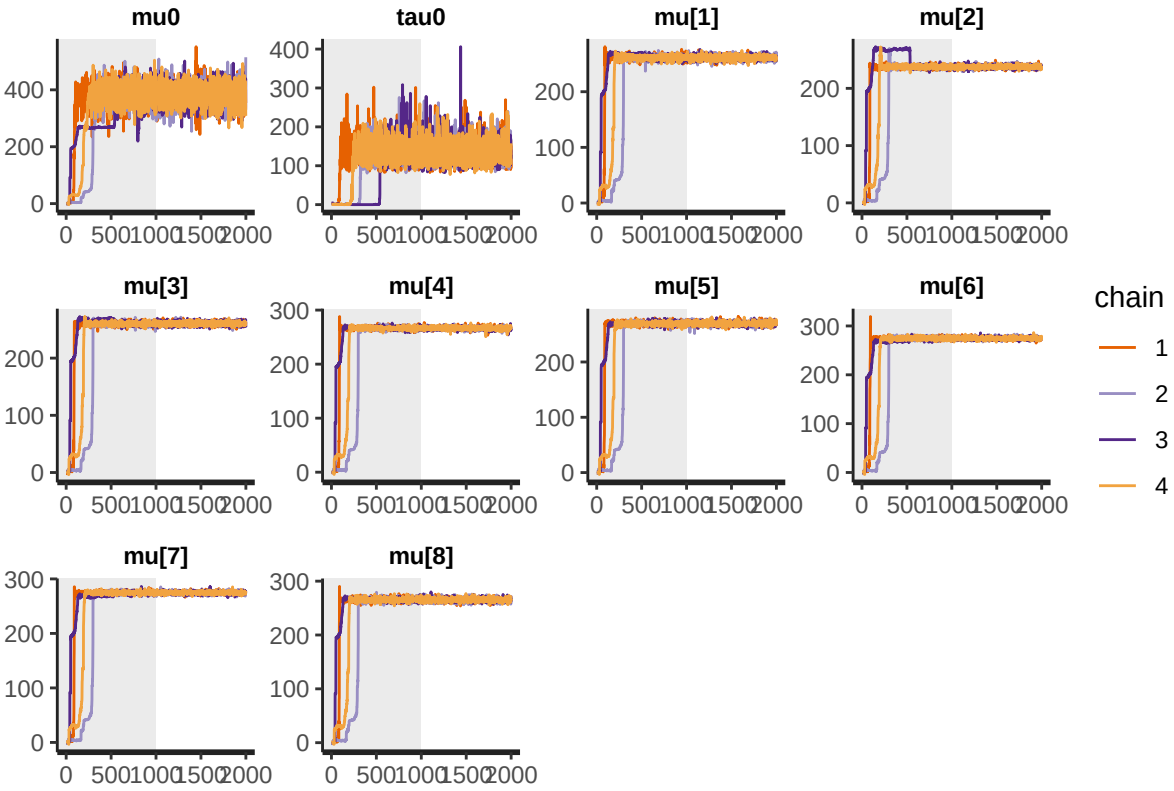
```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean    sd    2.5%   97.5% n_eff Rhat
## mu0        384.02    0.49 32.92  316.60  449.02  4490    1
## tau0       134.94    0.50 26.58   95.62  198.62  2828    1
## mu[1]      261.29    0.06  3.45  254.42  268.09  3467    1
## mu[2]      237.75    0.05  2.41  233.13  242.61  2689    1
## mu[3]      260.34    0.05  2.80  255.01  266.03  3512    1
## mu[4]      266.42    0.06  2.51  261.44  271.12  1851    1
## mu[5]      269.63    0.06  3.06  263.71  275.83  2853    1
## mu[6]      274.75    0.04  2.29  270.31  279.34  3337    1
## mu[7]      274.66    0.04  2.19  270.37  279.07  3337    1
```

```
## mu[8]      265.62    0.06  3.11  259.22  272.04  3111    1
## mu[9]      440.63    0.11  7.05  426.50  454.40  4162    1
## mu[10]     452.41    0.14  8.76  435.20  469.84  4120    1
## mu[11]     454.72    0.05  2.85  449.19  460.29  3428    1
## mu[12]     589.56    0.14  7.77  573.53  605.04  2931    1
## mu[13]     492.61    0.11  6.47  480.09  505.27  3352    1
## mu[14]     536.17    0.05  3.27  529.42  542.37  3565    1
## mu[15]     540.03    0.06  3.68  532.62  547.13  3548    1
## mu[16]     533.47    0.11  6.01  521.67  545.71  3060    1
## sigma[1]    10.97    0.05  2.71    7.13   17.43  3085    1
## sigma[2]     7.72    0.04  1.96    5.02   12.55  2994    1
## sigma[3]     9.03    0.04  2.18    5.96   14.40  3116    1
## sigma[4]     7.74    0.05  2.03    4.97   12.74  1811    1
## sigma[5]     9.40    0.05  2.35    6.15   15.42  2733    1
## sigma[6]     7.47    0.03  1.80    4.87   11.84  2669    1
## sigma[7]     6.95    0.03  1.65    4.57   10.84  3129    1
## sigma[8]     9.89    0.05  2.50    6.38   16.25  2246    1
## sigma[9]    22.06    0.10  5.32   14.48   34.94  3059    1
## sigma[10]   28.77    0.13  6.92   18.85   45.68  2954    1
## sigma[11]    9.06    0.05  2.38    5.88   14.74  2640    1
## sigma[12]   25.13    0.13  6.33   16.46   40.01  2346    1
## sigma[13]   20.30    0.10  5.12   13.13   33.01  2623    1
## sigma[14]   10.46    0.05  2.57    6.95   17.09  2810    1
## sigma[15]   11.76    0.05  2.82    7.73   18.54  3010    1
## sigma[16]   20.12    0.09  5.04   12.97   31.98  2865    1
## lp__      -634.38    0.12  4.53 -644.32 -626.67  1501    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 13 21:42:18 2024.
## 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).
```

The posterior expectation and 95% interval for all parameters are shown above.

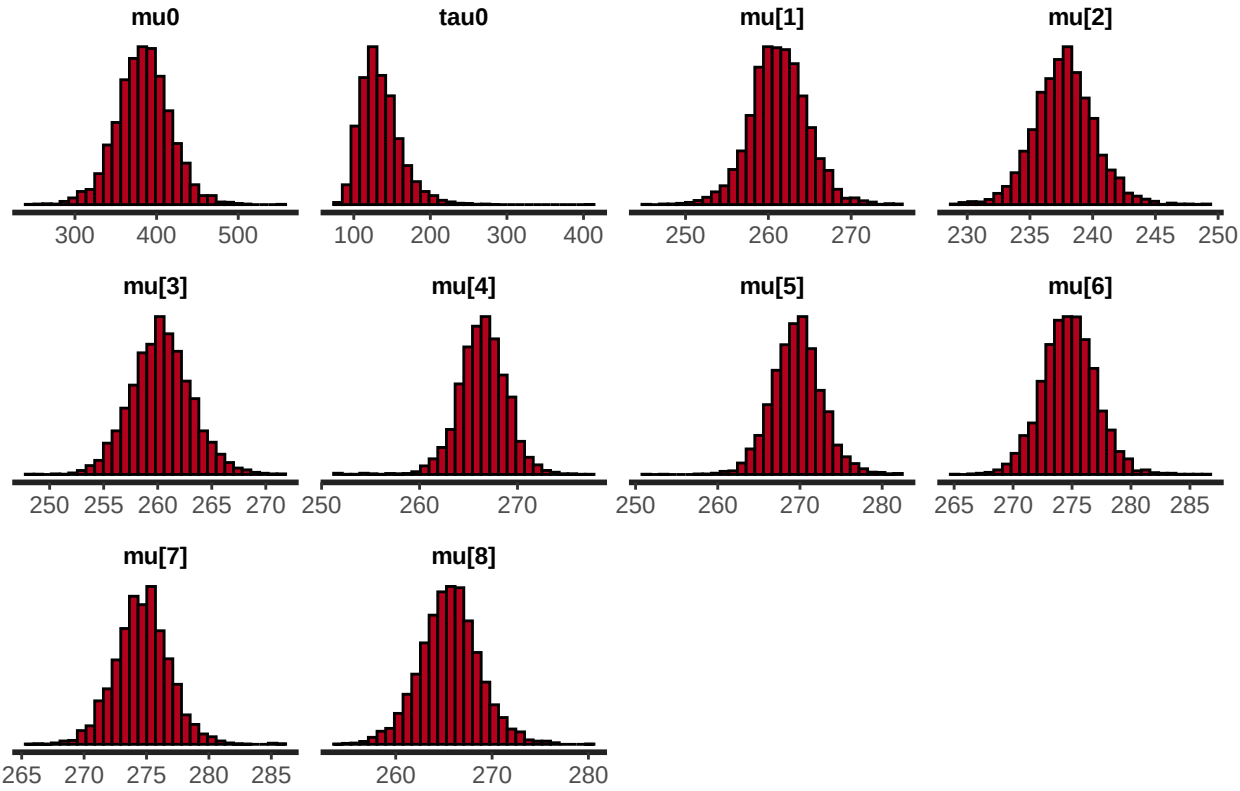
```
traceplot(fit, inc_warmup = TRUE)
```

```
## 'pars' not specified. Showing first 10 parameters by default.
```



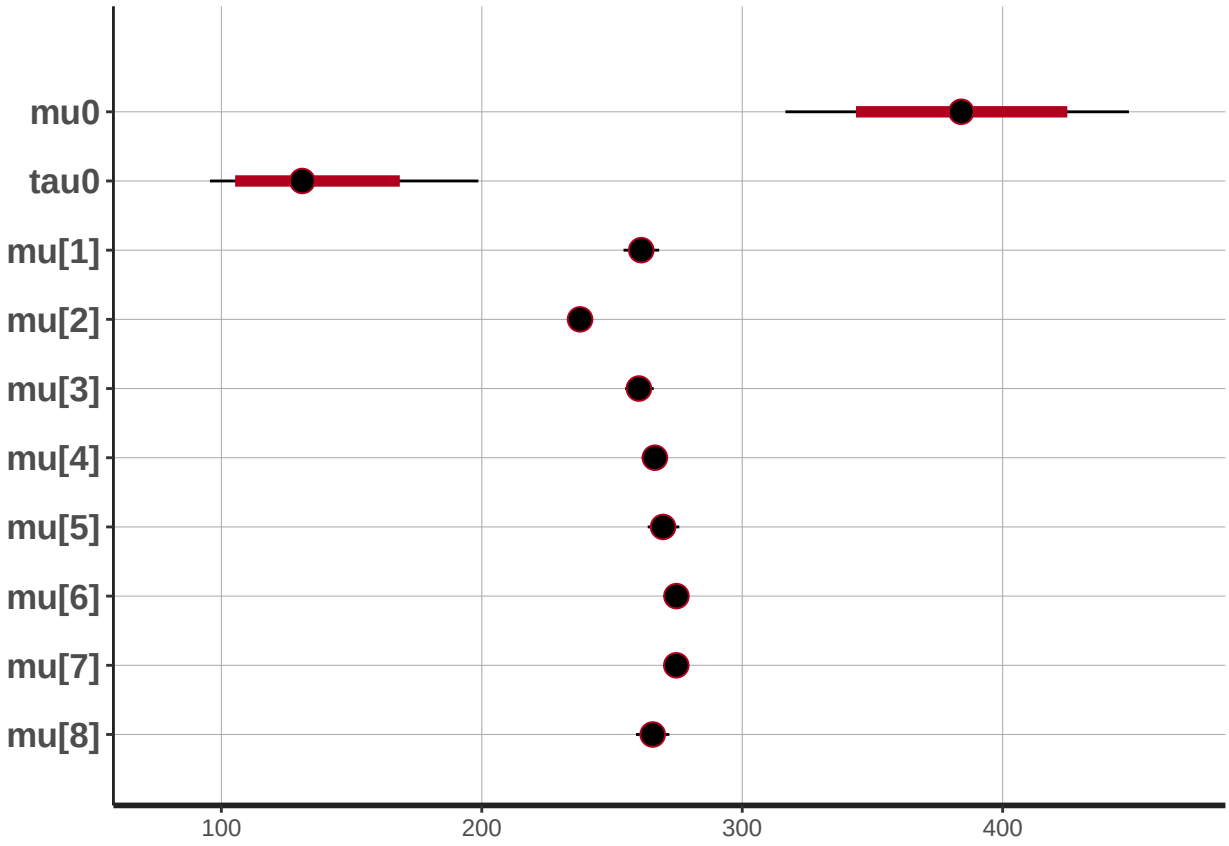

```
plot(fit, plotfun = "hist")
```

```
## 'pars' not specified. Showing first 10 parameters by default.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
plot(fit)
```

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



STAN CODE:

```
# data {
#   int<lower=0> J;
#   int<lower=0> N;
#   vector[N] y;
#   int<lower=1, upper=J> rat[N];
# }
#
# parameters {
#   real mu0;
#   real<lower=0> tau0;
#   vector[J] mu;
#   vector<lower=0>[J] sigma;
# }
#
# model {
#   mu0 ~ normal(0, 1000);
#   tau0 ~ scaled_inv_chi_square(1, 0.05);
#   mu ~ normal(mu0, tau0);
#   sigma ~ scaled_inv_chi_square(1, 0.05);
#
#   for (i in 1:N) {
#     y[i] ~ normal(mu[rat[i]], sigma[rat[i]]);
#   }
# }
```

Q2:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
radon_data = read.csv("radon.csv", header = TRUE)
```

```
# sample mean and sample variance
mean(radon_data$Radon)
```

```
## [1] 4.48016
```

```
var(radon_data$Radon)
```

```
## [1] 84.91611
```

```
# sample mean and standard deviation for 8 states
radon_data %>%
  group_by(state) %>%
  summarize(state_mean = mean(Radon, na.rm = TRUE),
            state_sd = sd(Radon, na.rm = TRUE))
```

```
## # A tibble: 8 x 3
##   state state_mean state_sd
##   <chr>      <dbl>    <dbl>
## 1 AZ          1.54     2.00
## 2 IN          3.67     5.04
## 3 MA          3.42     6.85
## 4 MN          4.77     4.48
## 5 MO          2.71     3.52
## 6 ND          7.37     9.44
## 7 PA          7.54    17.0
## 8 R5          3.25     5.00
```

```
library(rstan)

radon_data$state_index <- as.numeric(factor(radon_data$state))
data_list <- list(N = nrow(radon_data),
                  N_states = length(unique(radon_data$state_index)),
                  state = radon_data$state_index,
                  radon = radon_data$Radon)

fit2 <- stan(file = 'radon.stan', data = data_list)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000512 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.12 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 5.951 seconds (Warm-up)
## Chain 1:                3.598 seconds (Sampling)
## Chain 1:                9.549 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000452 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 4.52 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 6.442 seconds (Warm-up)
## Chain 2:                3.615 seconds (Sampling)
## Chain 2:                10.057 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000452 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 4.52 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 6.402 seconds (Warm-up)
## Chain 3:                3.6 seconds (Sampling)
## Chain 3:                10.002 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000454 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 4.54 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 6.064 seconds (Warm-up)
## Chain 4:                3.609 seconds (Sampling)
## Chain 4:                9.673 seconds (Total)
## Chain 4:
```

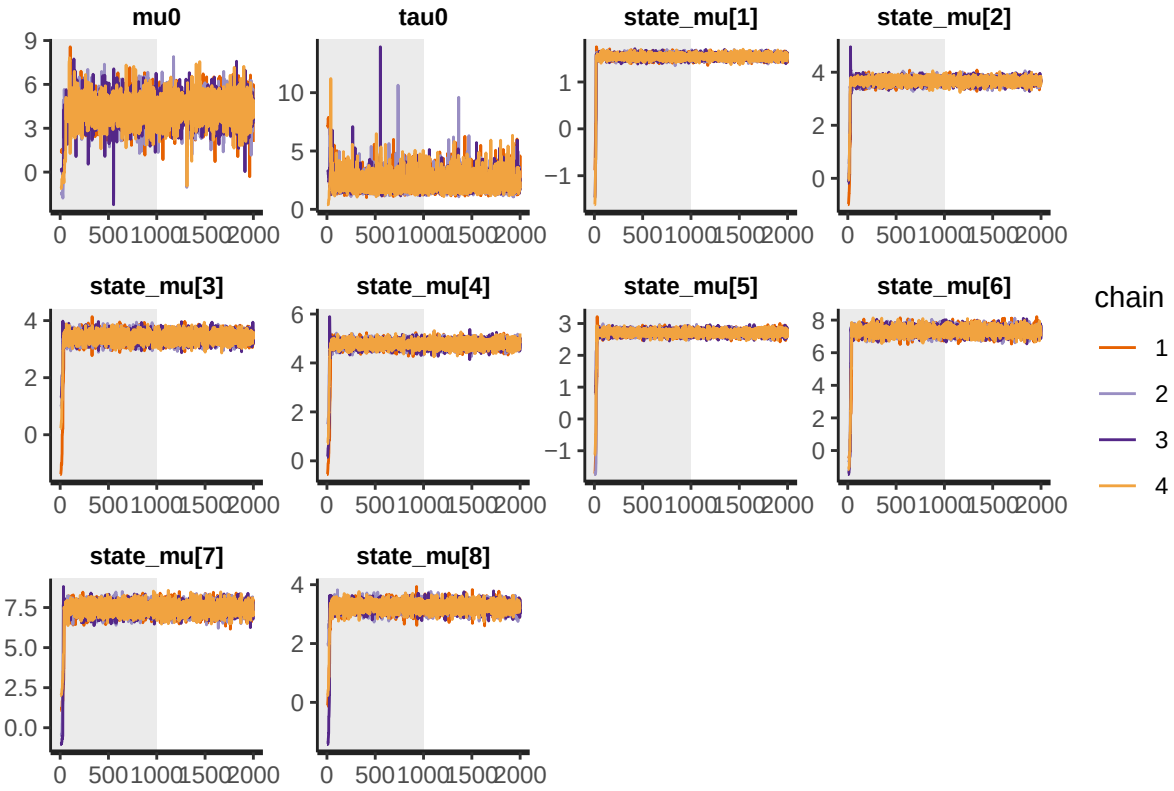
```
print(fit2, probs = c(0.025, 0.975))
```

```
## Inference for Stan model: anon_model.
## 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%     97.5% n_eff Rhat
## mu0              4.26    0.01 0.85      2.50      6.00  4541    1
## tau0             2.32    0.01 0.73      1.35      4.17  3413    1
## state_mu[1]      1.54    0.00 0.05      1.44      1.64  7376    1
## state_mu[2]      3.67    0.00 0.11      3.45      3.89  7443    1
## state_mu[3]      3.43    0.00 0.17      3.10      3.76  7743    1
## state_mu[4]      4.77    0.00 0.15      4.48      5.06  6806    1
```

```
## state_mu[5]      2.71    0.00 0.08      2.55      2.87  7000    1
## state_mu[6]      7.33    0.00 0.24      6.86      7.81  6882    1
## state_mu[7]      7.45    0.00 0.35      6.73      8.11  7680    1
## state_mu[8]      3.26    0.00 0.16      2.94      3.57  6480    1
## sigma[1]         2.00    0.00 0.04      1.93      2.07  9083    1
## sigma[2]         5.04    0.00 0.08      4.88      5.20  7369    1
## sigma[3]         6.85    0.00 0.12      6.62      7.09  7977    1
## sigma[4]         4.48    0.00 0.11      4.28      4.71  8065    1
## sigma[5]         3.52    0.00 0.06      3.41      3.64  8729    1
## sigma[6]         9.45    0.00 0.16      9.14      9.78  8349    1
## sigma[7]        16.96    0.00 0.25     16.47     17.45  9176    1
## sigma[8]         5.00    0.00 0.12      4.78      5.23  7871    1
## lp__        -29308.07    0.08 3.08 -29315.03 -29303.14  1365    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 13 21:43:21 2024.
## 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).
```

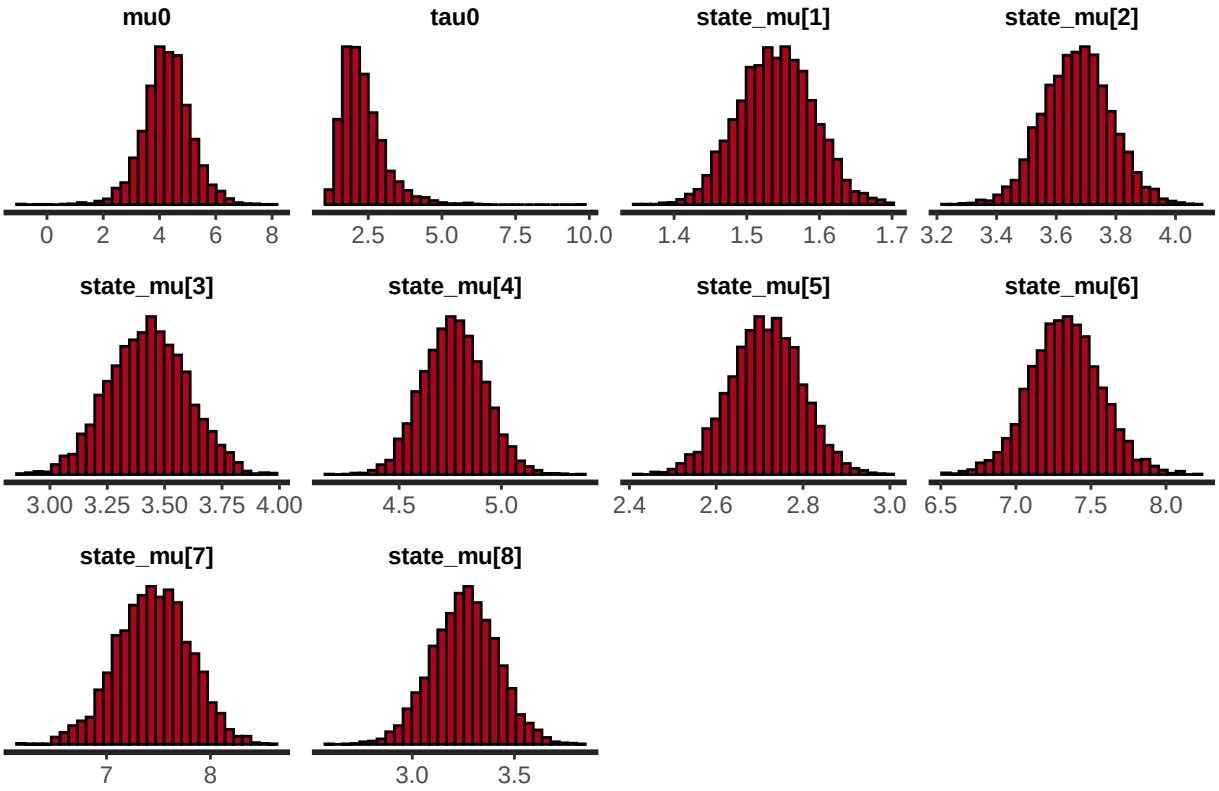
```
traceplot(fit2, inc_warmup = TRUE)
```

```
## 'pars' not specified. Showing first 10 parameters by default.
```



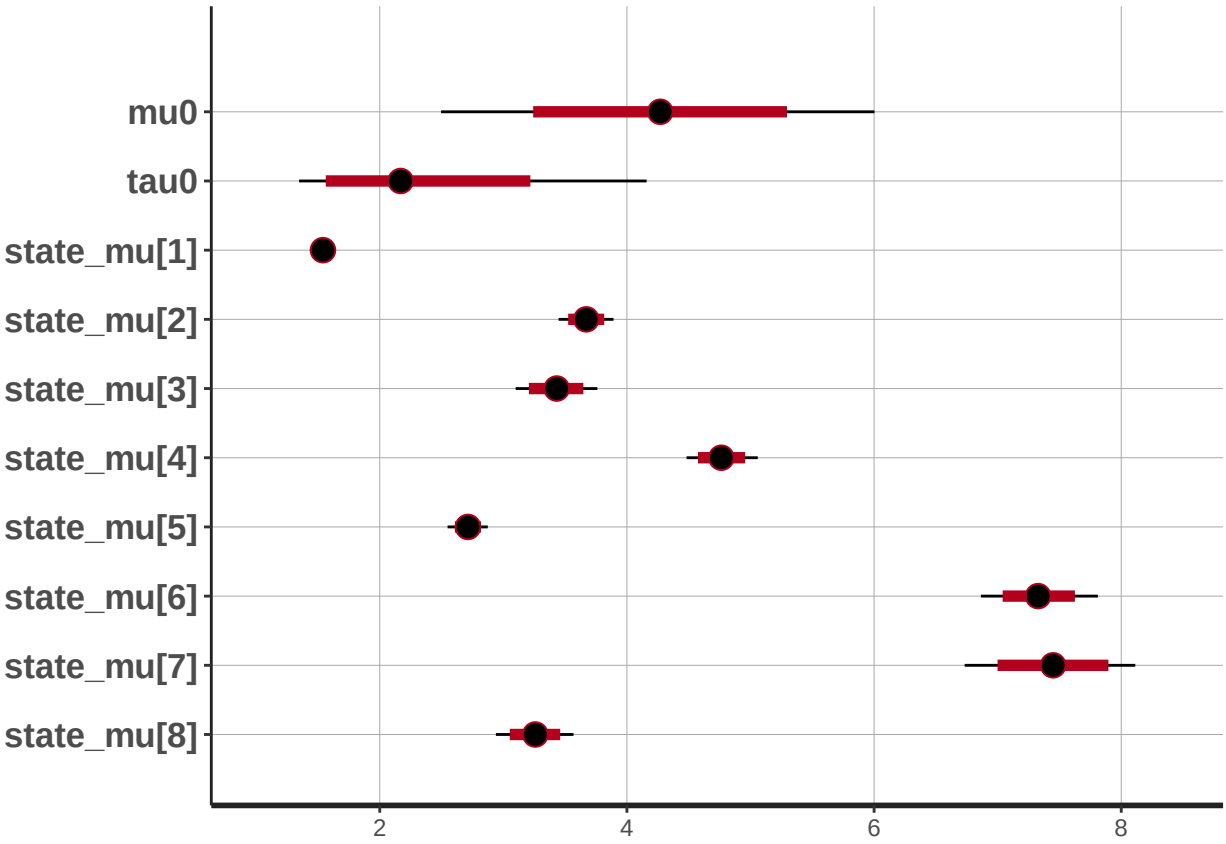

```
plot(fit2, plotfun = "hist")
```

```
## 'pars' not specified. Showing first 10 parameters by default.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
plot(fit2)
```

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



The hierarchy model in this example is similar as Question 1, I denote the i^{th} observed Radon for the j^{th} state as y_{ij}. Assume $y_{ij} \sim N(\mu_j, \sigma_j^2)$ and prior and hyperprior are given by:

$$\mu_j \sim N(\mu_0, \tau_0^2)$$
$$\sigma_j^2 \sim Inv - \chi^2(1, 0.05)$$
$$\mu_0 \sim N(0, 1000^2)$$
$$\tau_0^2 \sim Inv - \chi^2(1, 0.05)$$

The posterior expectation and 95% interval are shown above.

STAN CODE:

```
# data {
#   int<lower=0> N;
#   int<lower=0> N_states;
#   int<lower=1, upper=N_states> state[N];
#   vector[N] radon;
# }
#
# parameters {
#   real mu0;
#   real<lower=0> tau0;
#   vector[N_states] state_mu;
#   vector<lower=0>[N_states] sigma;
# }
#
```

```
# model {
#   // Priors
#   mu0 ~ normal(0, 1000);
#   tau0 ~ scaled_inv_chi_square(1, 0.05);
#   sigma ~ scaled_inv_chi_square(1, 0.05);
#   state_mu ~ normal(mu0, tau0);
#
#   // Model
#   for (i in 1:N) {
#     radon[i] ~ normal(state_mu[state[i]], sigma[state[i]]);
#   }
# }
```