Homework IV

Bayesian Data Analysis - UPC Spring 2022

Alexander J Ohrt

11 april, 2022

Exercise 4.4: Two Methods of Training Workers; Comparison of Means

The Human Resources Department of a large company wishes to compare two methods of training industrial workers to perform a skilled task. Twenty workers are selected: 10 of them are randomly assigned to be trained using method A, and the other 10 are assigned to be trained using method B. After the training is complete, all the workers are tested on the speed of performance at the task. The times taken to complete the task are

```
method.A <- c(115, 120, 111, 123, 116, 121, 118, 116, 127, 129)
method.B <- c(123, 131, 113, 119, 123, 113, 128, 126, 125, 128)
df <- rbind(c(method.A,mean(method.A), sd(method.A)), c(method.B,mean(method.B), sd(method.B)))
rownames(df) <- c("Method A", "Method B")
colnames(df) <- c(rep("", 10), "Mean", "Standard Error")
```

Table 1: Times Taken to Complete Task for each Method

											Mean	Standard Error
Method A	115	120	111	123	116	121	118	116	127	129	119.6	5.581716
Method B	123	131	113	119	123	113	128	126	125	128	122.9	6.172520

a) Find Posterior Distributions of Parameters μ_A, μ_B

We assume that the observations come from $N(\mu_A, \sigma)$ and $N(\mu_A, \sigma)$, where $\sigma = 6$. Use independent $N(m, s)$ prior distributions for μ_A, μ_B, where $m = 100, s = 20$. The posterior distributions of the parameters are found using Stan (via R).

First we define the Stan model and fit it.

```
# Define model and call stan.
data_list <- list(
  nA = length(method.A),
  nB = length(method.B),
  tA = method.A,
  tB = method.B
)

fit <- stan("4-4_training_workers.stan", iter = 1000, chains = 4,
            data = data_list, seed = 1)
```

```
## Trying to compile a simple C file
```

```
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/ajo/R/x86_64-pc-linux-gnu-library/4.1/Rcpp/include/
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: err
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: e
##   628 | namespace Eigen {
##       |                 ^
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: complex
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL '4-4_training_workers' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## 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: 0.003438 seconds (Warm-up)
## Chain 1:                0.002713 seconds (Sampling)
## Chain 1:                0.006151 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4-4_training_workers' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## 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: 0.003435 seconds (Warm-up)
## Chain 2:                0.003011 seconds (Sampling)
## Chain 2:                0.006446 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4-4_training_workers' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 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: 0.003385 seconds (Warm-up)
## Chain 3:                0.003323 seconds (Sampling)
## Chain 3:                0.006708 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4-4_training_workers' 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)
```

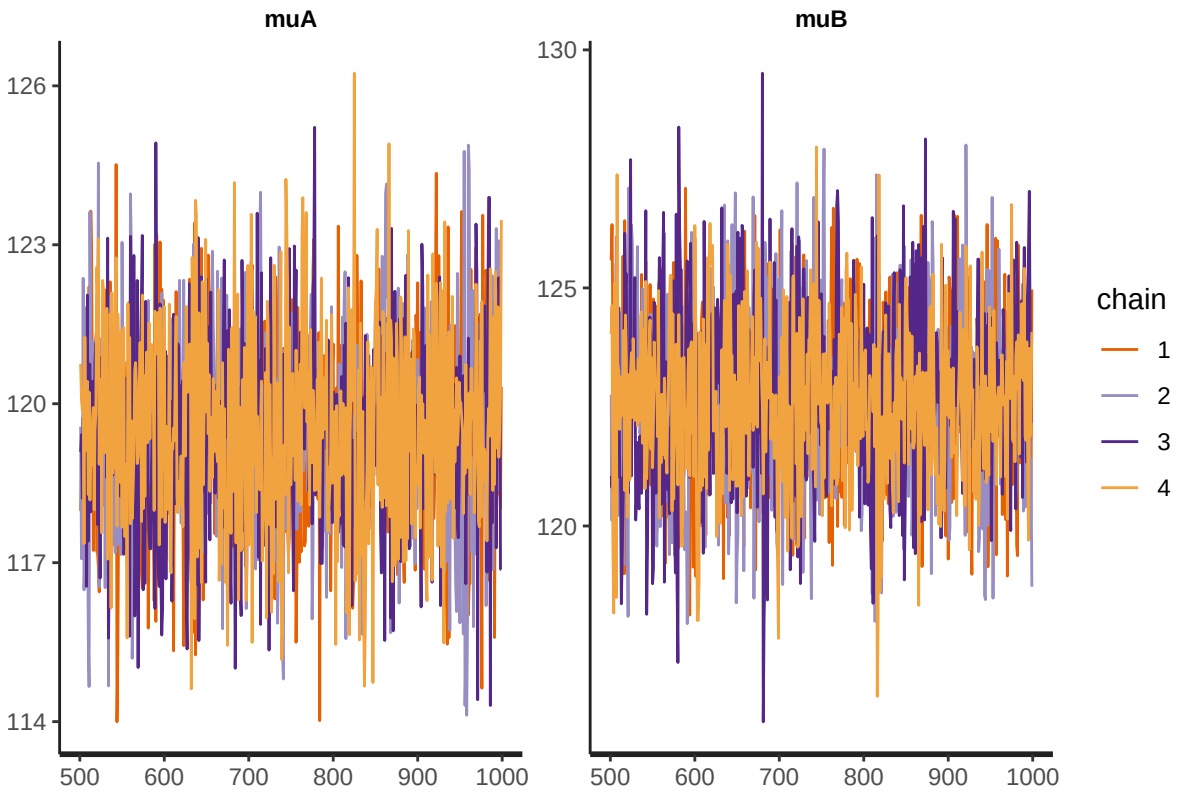
```
## Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.003541 seconds (Warm-up)
## Chain 4:                0.00298 seconds (Sampling)
## Chain 4:                0.006521 seconds (Total)
## Chain 4:
```

The convergence analysis shows that the chains have converged, because `Rhat = 1`. Moreover, the traceplots seem to show that the chains have converged to similar values.

```
# Convergence analysis.
print(fit)
```

```
## Inference for Stan model: 4-4_training_workers.
## 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
## muA  119.47    0.04 1.83 115.77 118.22 119.49 120.65 123.06  2183    1
## muB  122.70    0.04 1.86 119.02 121.42 122.73 123.97 126.29  2018    1
## lp__ -10.73    0.04 0.94 -13.13 -11.11 -10.45 -10.05  -9.81   668    1
##
## Samples were drawn using NUTS(diag_e) at Mon Apr 11 12:39:47 2022.
## 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(fit)
```



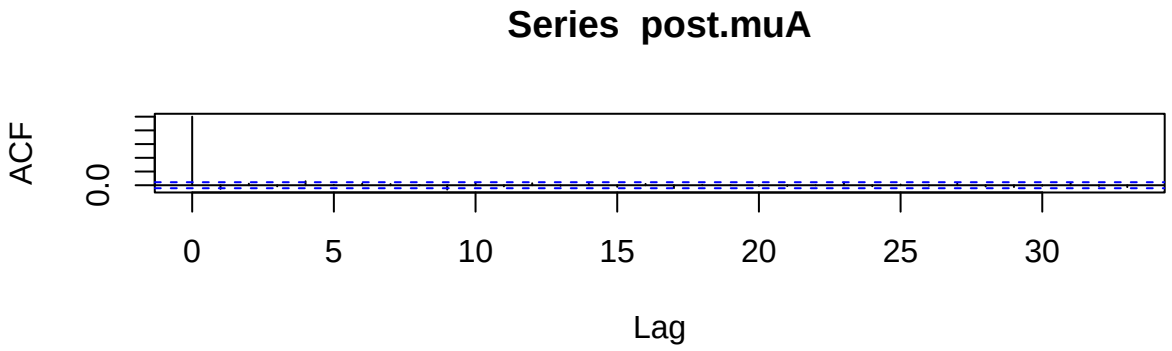

```
posterior <- as.data.frame(fit)
head(posterior)
```

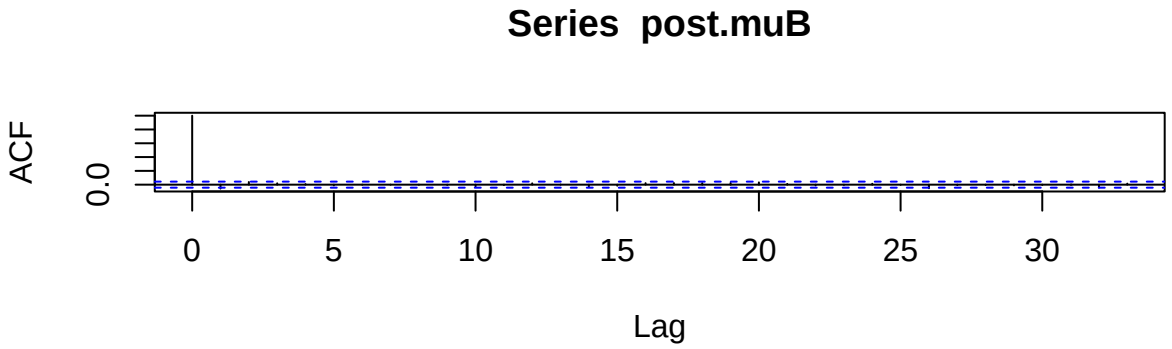
```
##        muA      muB       lp__
## 1 117.9793 125.5803 -11.241568
## 2 119.0696 126.3306 -11.651754
## 3 118.8418 124.1635 -10.132137
## 4 118.1667 124.8950 -10.682264
## 5 120.1579 122.0561  -9.915097
## 6 118.8075 123.2357  -9.876855
```

```
dim(posterior)
```

```
## [1] 2000    3
```

```
post.muA <- posterior[,1]
post.muB <- posterior[,2]

# Are the autocorrelation plots needed for something here?
par(mfrow = c(2, 1))
acf(post.muA)
acf(post.muB)
```

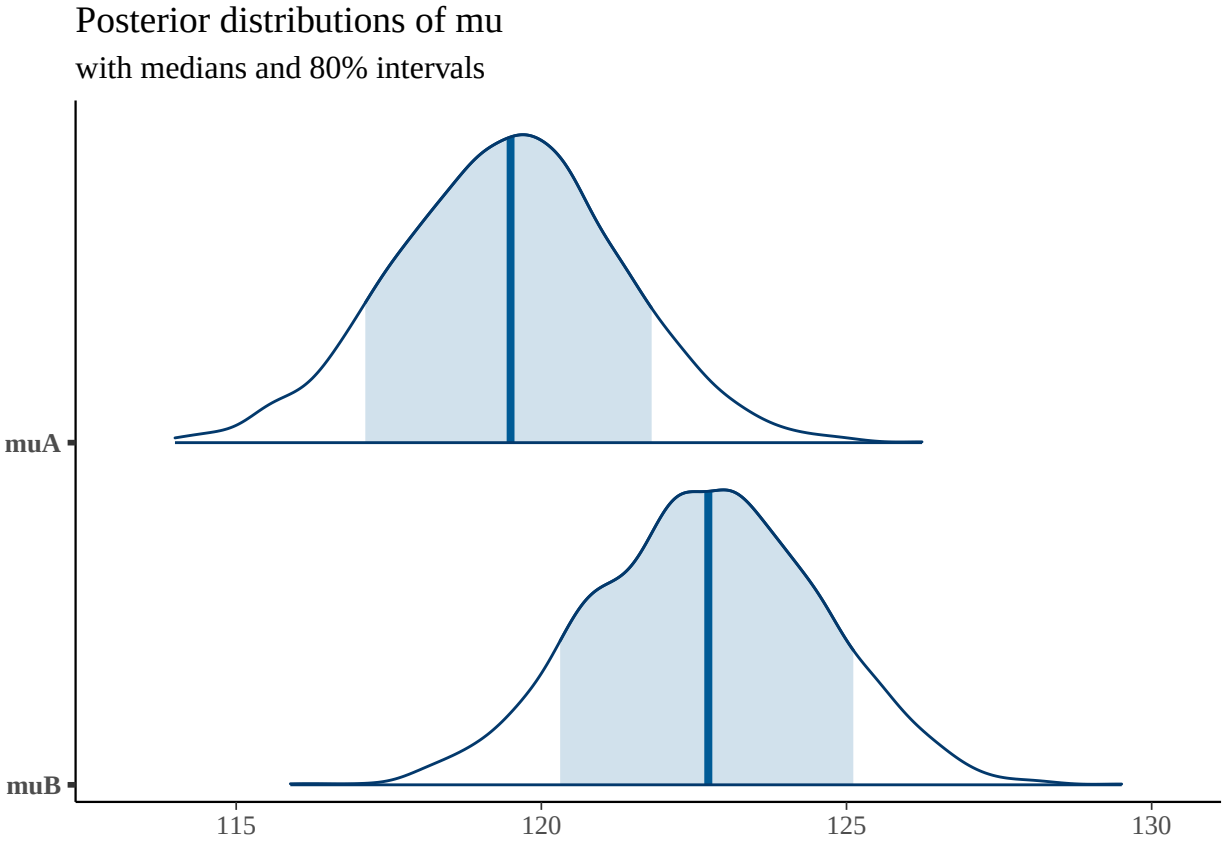





```
par(mfrow = c(1, 1))
```

The posterior distributions are plotted with medians and 80% intervals below.

```
plot_title <- ggtitle("Posterior distributions of mu", "with medians and 80% intervals")
mcmc_areas(posterior,
  pars = c("muA", "muB"),
  prob = 0.8) + plot_title
```



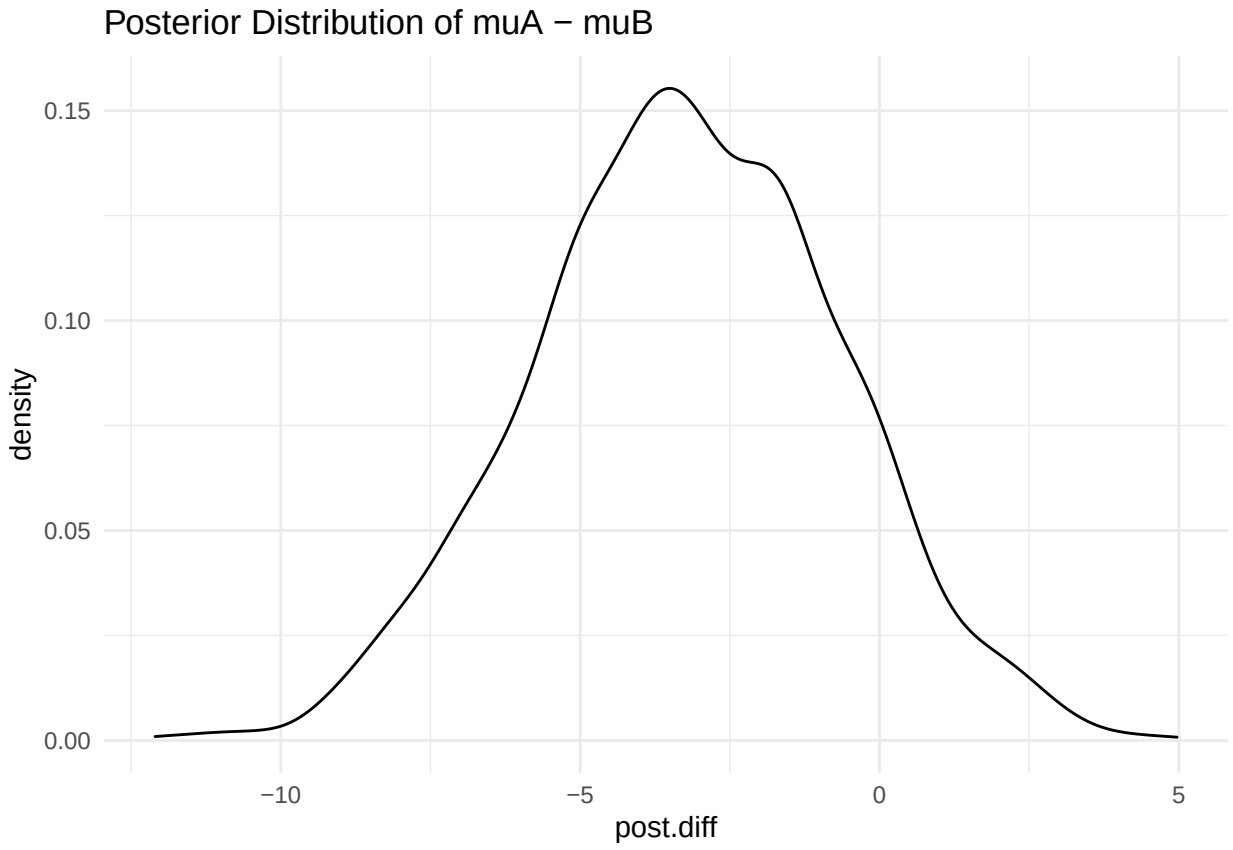
The Stan code is given below for completeness. The `.stan` files are also submitted.

```
data{
  int<lower=0> nA;
  int<lower=0> nB;

  real<lower=0> tA[nA];
  real<lower=0> tB[nB];
}

parameters{
  real muA;
  real muB;
}

model{
  tA ~ normal(muA, 6);
  tB ~ normal(muB, 6);

  muA ~ normal(100, 20);
  muB ~ normal(100, 20);
}
```

b) Find Posterior Distribution of $\mu_A - \mu_B$

The posterior distribution of $\mu_A - \mu_B$ is found and plotted below.

```r
post.diff <- post.muA - post.muB
#t <- seq(105, 135, length.out=length(post.muA))
ggplot(tibble(post.diff)) +
  geom_density(aes(post.diff)) +
  theme_minimal() +
  ggtitle("Posterior Distribution of muA - muB")
```



c) Find a 95% Bayesian Credible Interval (CI) for $\mu_A - \mu_B$

A 95% quantile CI for $\mu_A - \mu_B$ is found below.

```r
(CI.perc <- quantile(post.diff, probs = c(0.025, 0.975)))
```

```
##      2.5%     97.5%
## -8.226049  1.763004
```

d) Repeat the Previous Problems, Supposing that σ is Unknown

The steps are repeated supposing that σ is unknown. Notice that we still assume that it is equal in both populations. When σ is unknown, we have to define a prior distribution for this parameter as well. We will solve the problem assuming two different priors for σ;

1) $\pi_1 \sim U[0, 50]$ (uninformative)
2) $\pi_1 \sim N(6, 10)$

Case 1)

Using the uninformative prior, the results are given in the following.

```
fit2 <- stan("4-4_training_workers2.stan", iter = 1000, chains = 4,
             data = data_list, seed = 1)
```

```
## Trying to compile a simple C file
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/ajo/R/x86_64-pc-linux-gnu-library/4.1/Rcpp/include/
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: er
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: e
##   628 | namespace Eigen {
##       |                 ^
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: complex
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL '4-4_training_workers2' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
## 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: 0.007179 seconds (Warm-up)
## Chain 1:                0.234381 seconds (Sampling)
## Chain 1:                0.24156 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4-4_training_workers2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-06 seconds
```

```
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## 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: 0.009481 seconds (Warm-up)
## Chain 2:                0.005954 seconds (Sampling)
## Chain 2:                0.015435 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4-4_training_workers2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.04 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: 0.009348 seconds (Warm-up)
## Chain 3:                0.011817 seconds (Sampling)
## Chain 3:                0.021165 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4-4_training_workers2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.03 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: 0.009366 seconds (Warm-up)
## Chain 4:                0.006318 seconds (Sampling)
## Chain 4:                0.015684 seconds (Total)
## Chain 4:
```

```
## Warning: There were 1610 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: There were 33 transitions after warmup that exceeded the maximum treedepth. Increase max_tr
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## https://mc-stan.org/misc/warnings.html#bfmi-low
```

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

```
## Warning: The largest R-hat is 2.88, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
```

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

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

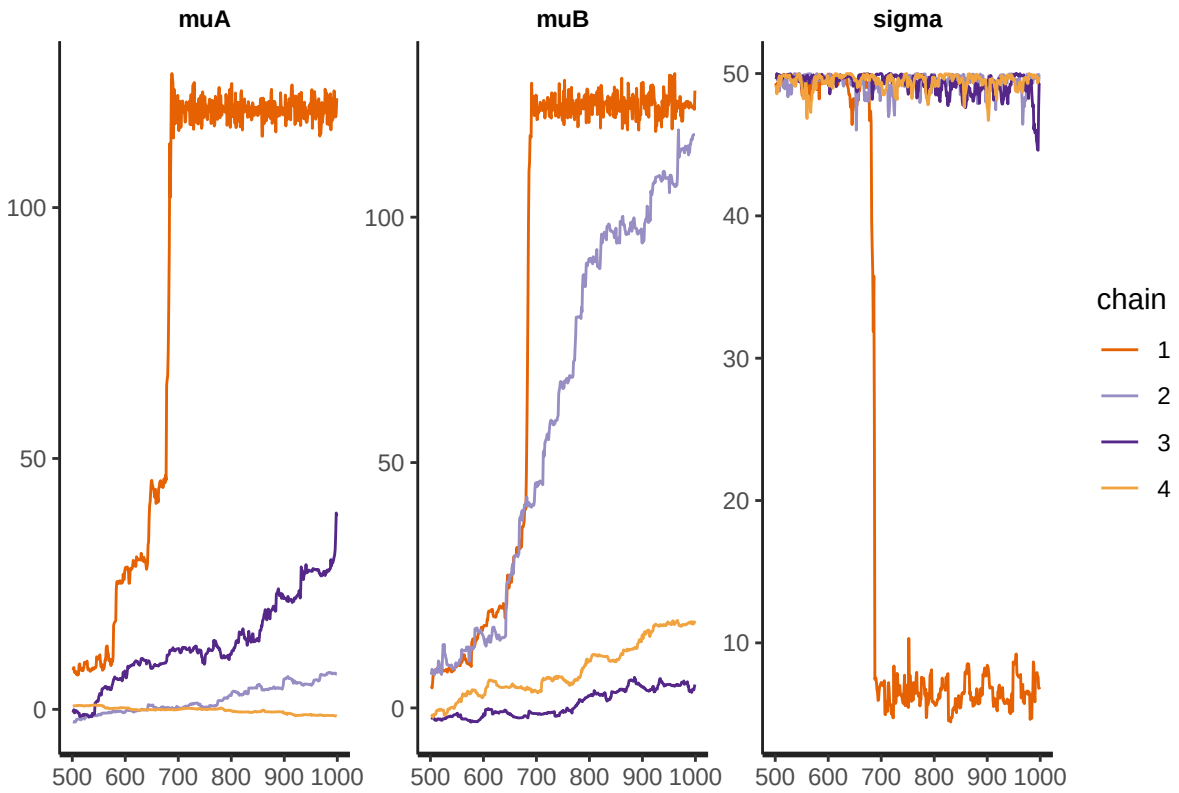
```
# Hva gjør jeg hvis den divergerer?
```

```
# Convergence analysis.
print(fit2) # This has not converged!
```

```
## Inference for Stan model: 4-4_training_workers2.
## 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
## muA     24.92   23.66 42.26   -1.30    0.00    5.23   22.91 121.73     3 2.79
## muB     38.58   22.23 47.82   -2.21    3.66   11.61   91.64 125.02     5 2.73
## sigma   42.63    7.76 15.59    5.64   48.58   49.46   49.78  49.98     4 2.49
## lp__  -127.68   20.24 38.23 -160.47 -153.29 -147.52 -114.47 -44.32     4 2.71
##
## Samples were drawn using NUTS(diag_e) at Mon Apr 11 13:15:23 2022.
## 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)
```



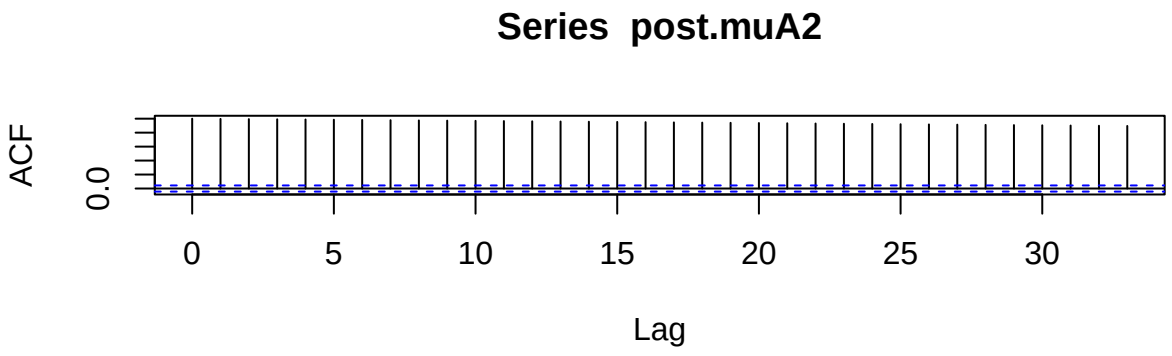

```
posterior2 <- as.data.frame(fit2)
head(posterior2)
```

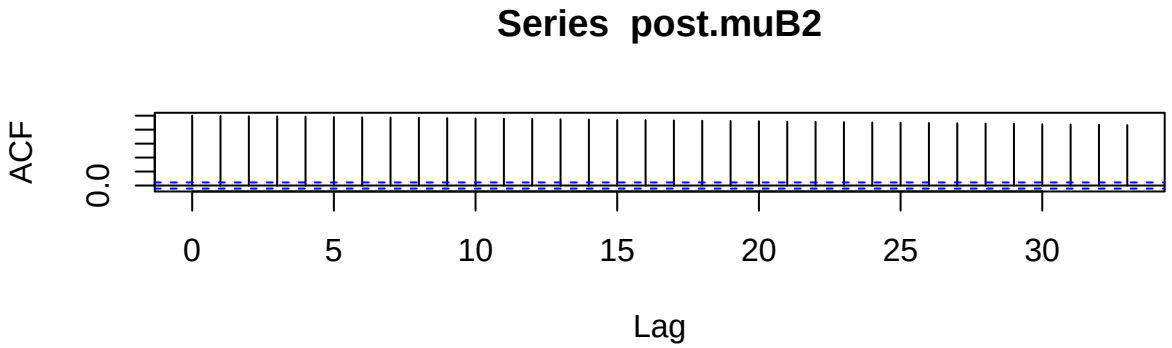
```
##        muA      muB    sigma      lp__
## 1 7.850939 4.376298 49.30773 -150.8080
## 2 8.063769 4.404942 49.08983 -151.0420
## 3 8.344247 4.073972 49.21769 -150.8537
## 4 7.422083 5.015900 49.31794 -150.6225
## 5 7.309578 6.521104 49.02922 -150.1460
## 6 7.507586 6.953364 49.55823 -148.7505
```

```
dim(posterior2)
```

```
## [1] 2000    4
```

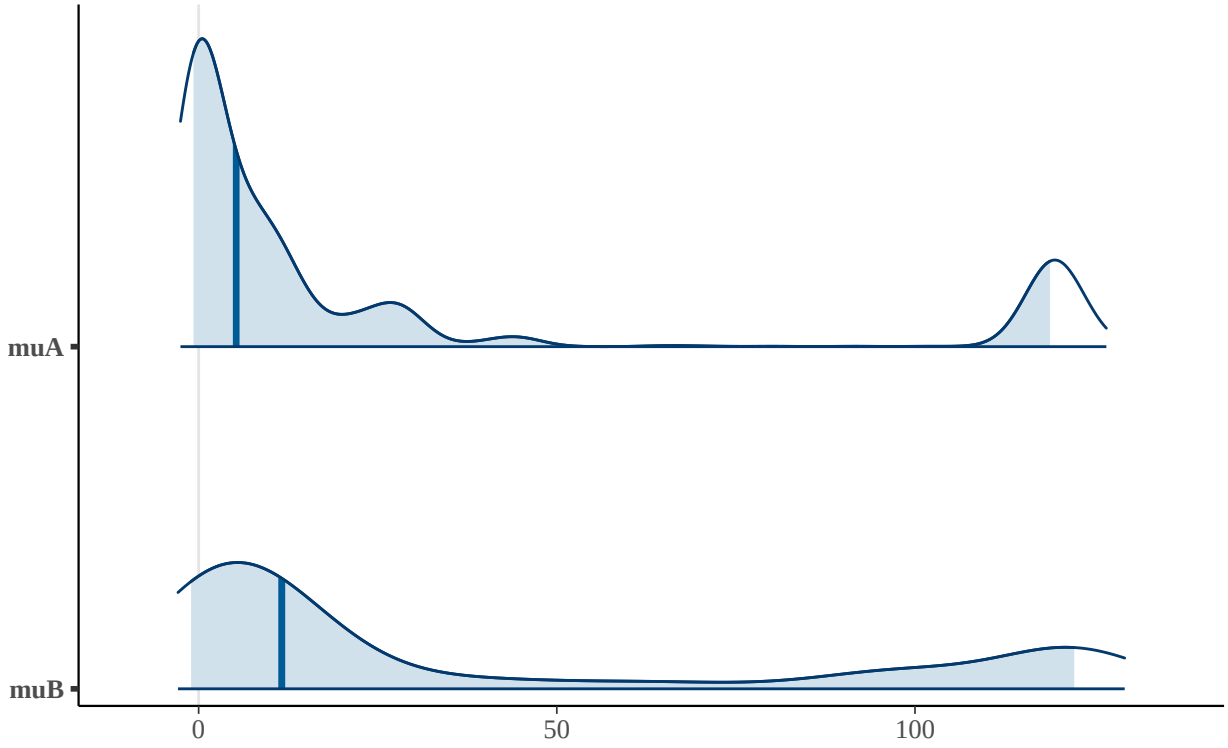
```
post.muA2 <- posterior2[,1]
post.muB2 <- posterior2[,2]

# Are the autocorrelation plots needed for something here?
par(mfrow = c(2, 1))
acf(post.muA2)
acf(post.muB2)
```





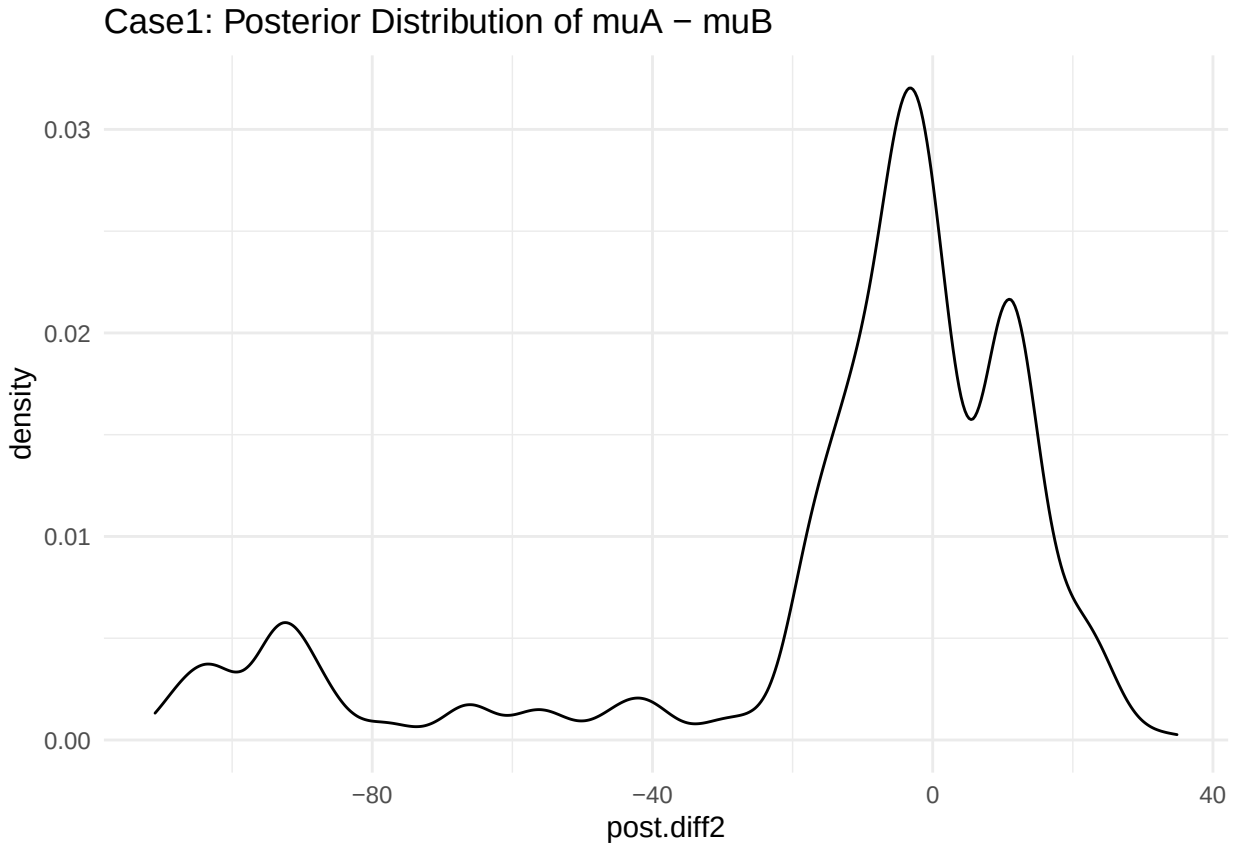

```
par(mfrow = c(1, 1))
```

```
plot_title <- ggtitle("Case 1: Posterior distributions of mu", "with medians and 80% intervals")
mcmc_areas(posterior2,
  pars = c("muA", "muB"),
  prob = 0.8) + plot_title
```



The posterior of $\mu_A - \mu_B$ is given below.

```
post.diff2 <- post.muA2 - post.muB2
#t <- seq(105, 135, length.out=length(post.muA))
ggplot(tibble(post.diff2)) +
  geom_density(aes(post.diff2)) +
  theme_minimal() +
  ggtitle("Case1: Posterior Distribution of muA - muB")
```



A 95% quantile CI for $\mu_A - \mu_B$ is found below.

```
(CI.perc2 <- quantile(post.diff2, probs = c(0.025, 0.975)))
```

```
##       2.5%      97.5%
## -102.96318   22.62197
```

The Stan code used in this case is very similar to the code used earlier, but is shown below.

```
data{
  int<lower=0> nA;
  int<lower=0> nB;

  real<lower=0> tA[nA];
  real<lower=0> tB[nB];
}

parameters{
  real muA;
  real muB;
  real<lower=0> sigma;
}

model{
  tA ~ normal(muA, sigma);
  tB ~ normal(muB, sigma);

  muA ~ normal(100, 20);
  muB ~ normal(100, 20);
```

```
  sigma ~ uniform(0,50);
}
```

Case 2)

Using the normal prior, the results are given in the following.

```
fit3 <- stan("4-4_training_workers3.stan", iter = 1000, chains = 4,
             data = data_list, seed = 1)
```

```
## Trying to compile a simple C file
## Running /usr/lib/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/ajo/R/x86_64-pc-linux-gnu-library/4.1/Rcpp/include/
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:88,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: er
##   628 | namespace Eigen {
##       | ^~~~~~~~~
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: e
##   628 | namespace Eigen {
##       |                 ^
## In file included from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Dense:1,
##                  from /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/StanHeaders/include/stan/math/prim/
##                  from <command-line>:
## /home/ajo/R/x86_64-pc-linux-gnu-library/4.1/RcppEigen/include/Eigen/Core:96:10: fatal error: complex
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib/R/etc/Makeconf:168: foo.o] Error 1
##
## SAMPLING FOR MODEL '4-4_training_workers3' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 8e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## 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: 0.017864 seconds (Warm-up)
## Chain 1:                0.00834 seconds (Sampling)
```

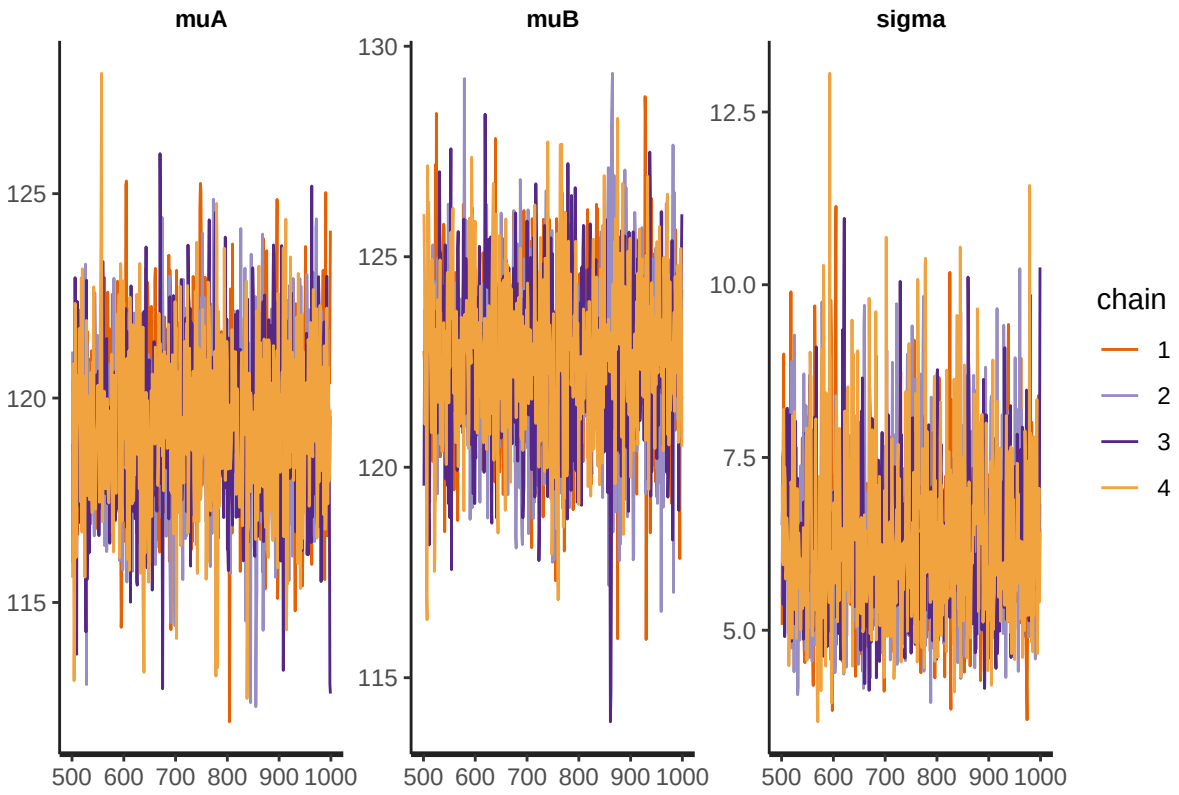
```
## Chain 1:                0.026204 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '4-4_training_workers3' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 2: Adjust 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: 0.01987 seconds (Warm-up)
## Chain 2:                0.006788 seconds (Sampling)
## Chain 2:                0.026658 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '4-4_training_workers3' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.04 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: 0.016577 seconds (Warm-up)
## Chain 3:                0.007805 seconds (Sampling)
## Chain 3:                0.024382 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '4-4_training_workers3' NOW (CHAIN 4).
```

```
## Chain 4:
## Chain 4: Gradient evaluation took 6e-06 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06 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: 0.01298 seconds (Warm-up)
## Chain 4:                0.007739 seconds (Sampling)
## Chain 4:                0.020719 seconds (Total)
## Chain 4:
```

```
print(fit3)
```

```
## Inference for Stan model: 4-4_training_workers3.
## 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
## muA   119.41    0.05 2.03 115.56 118.02 119.38 120.84 123.19  1747    1
## muB   122.64    0.05 1.95 118.80 121.37 122.71 123.93 126.21  1666    1
## sigma   6.33    0.03 1.17   4.46   5.52   6.17   7.01   9.05  1551    1
## lp__  -45.41    0.04 1.28 -48.75 -45.99 -45.07 -44.47 -43.92   828    1
##
## Samples were drawn using NUTS(diag_e) at Mon Apr 11 13:16:04 2022.
## 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(fit3)
```



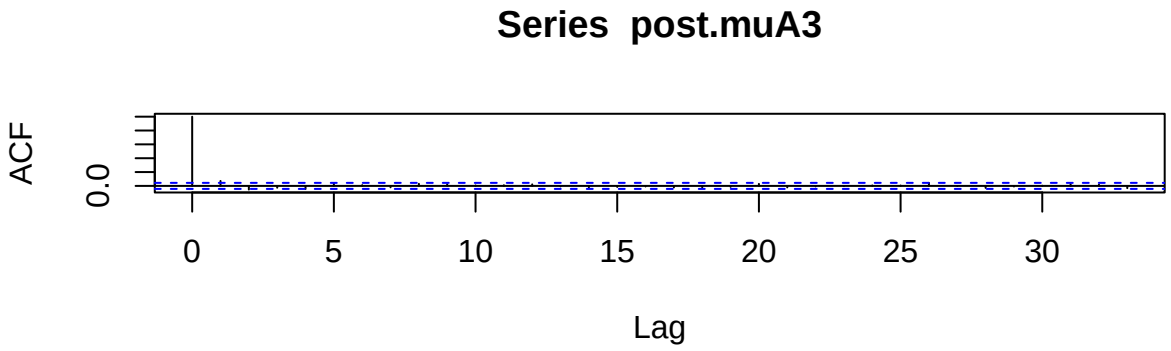

```
posterior3 <- as.data.frame(fit3)
head(posterior3)
```

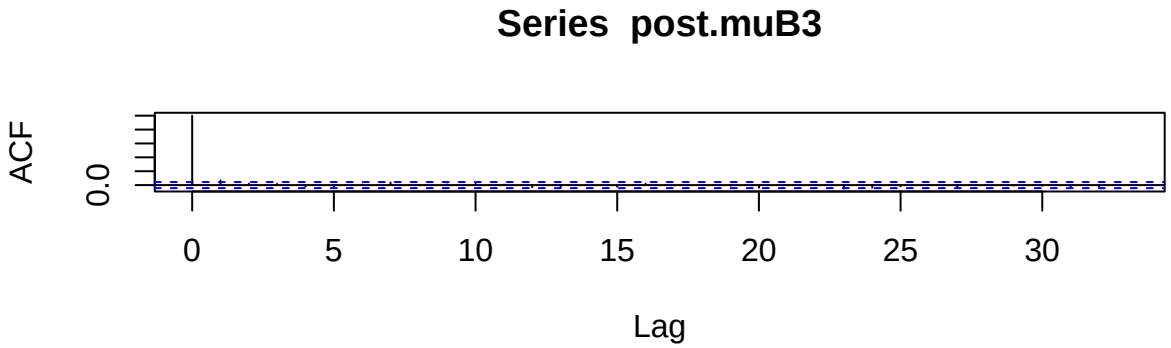
```
##        muA      muB    sigma      lp__
## 1 120.4746 122.7607 5.076882 -44.28855
## 2 117.3510 122.4354 7.514575 -45.32296
## 3 115.7993 123.7562 7.609505 -46.28177
## 4 116.6465 123.0047 8.998260 -47.18480
## 5 121.0754 121.8027 5.646579 -44.34474
## 6 120.9411 121.6110 5.843328 -44.30687
```

```
dim(posterior3)
```

```
## [1] 2000    4
```

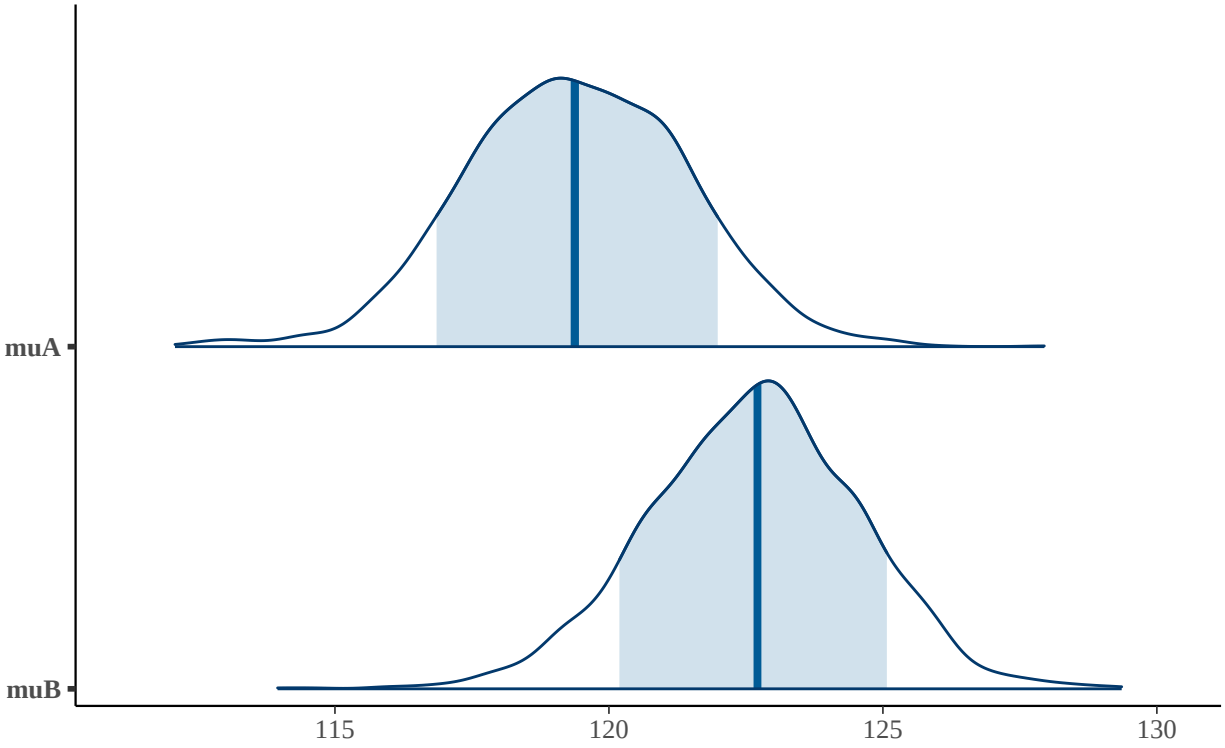
```
post.muA3 <- posterior3[,1]
post.muB3 <- posterior3[,2]

# Are the autocorrelation plots needed for something here?
par(mfrow = c(2, 1))
acf(post.muA3)
acf(post.muB3)
```





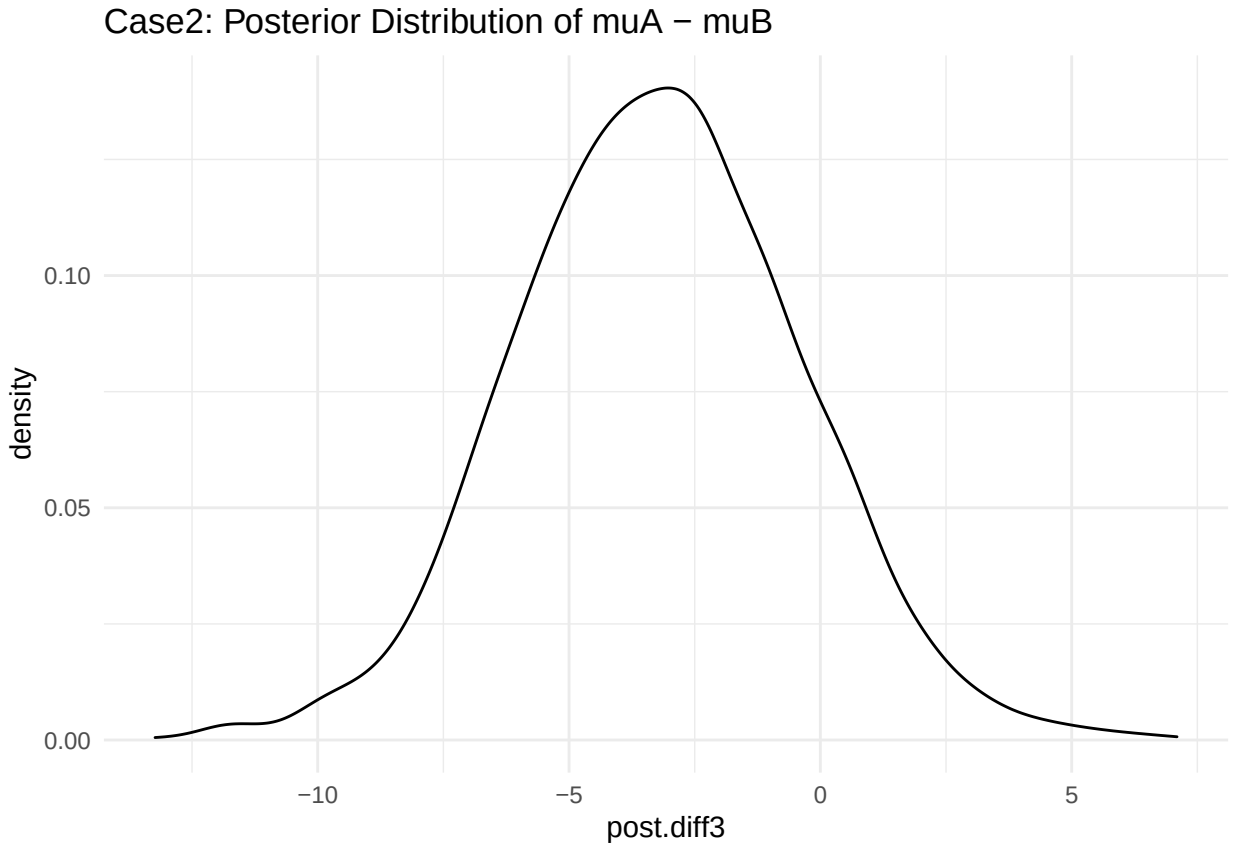

```
par(mfrow = c(1, 1))
```

```
plot_title <- ggtitle("Case 2: Posterior distributions of mu", "with medians and 80% intervals")
mcmc_areas(posterior3,
  pars = c("muA", "muB"),
  prob = 0.8) + plot_title
```



The posterior of $\mu_A - \mu_B$ is given below.

```
post.diff3 <- post.muA3 - post.muB3
#t <- seq(105, 135, length.out=length(post.muA))
ggplot(tibble(post.diff3)) +
  geom_density(aes(post.diff3)) +
  theme_minimal() +
  ggtitle("Case2: Posterior Distribution of muA - muB")
```



A 95% quantile CI for $\mu_A - \mu_B$ is found below.

```
(CI.perc3 <- quantile(post.diff3, probs = c(0.025, 0.975)))
```

```
##      2.5%     97.5%
## -8.785599  2.284193
```

The Stan code is again given below. The only difference compared to in Case 1 is the change in the prior for sigma.

```
data{
  int<lower=0> nA;
  int<lower=0> nB;

  real<lower=0> tA[nA];
  real<lower=0> tB[nB];
}

parameters{
  real muA;
  real muB;
  real<lower=0> sigma;
}

model{
  tA ~ normal(muA, sigma);
  tB ~ normal(muB, sigma);

  muA ~ normal(100, 20);
```

```
  muB ~ normal(100, 20);
  sigma ~ normal(6,10);
}
```