# BDA - Project

## Contents

Data loading

```
data <- read.csv("data/airfoil_self_noise.csv", sep=";")
print(data[1:5,])
```

```
##   frequency angle_of_attack chord_length free_stream_velocity
## 1       800               0       0.3048                 71.3
## 2      1000               0       0.3048                 71.3
## 3      1250               0       0.3048                 71.3
## 4      1600               0       0.3048                 71.3
## 5      2000               0       0.3048                 71.3
##   suction_side_displacement_thickness scaled_sound_pressure_level
## 1                          0.00266337                      126.201
## 2                          0.00266337                      125.201
## 3                          0.00266337                      125.951
## 4                          0.00266337                      127.591
## 5                          0.00266337                      127.461
```

### Model 1)

```
model_code_1 = root("models/model1.stan")
writeLines(readLines(model_code_1))
```

```
## data {
##   int<lower=0> n;  // number of data items
##   int<lower=0> k;  // number of predictors
##   matrix[n,k] X;   // predictor matrix
##   vector[n] Y;          // outcome vector
## }
##
## parameters {
##   real alpha;           // intercept
##   matrix[k,1] beta;     // coefficients for predictors
##   real<lower=0> sigma;  // error scale
## }
##
## transformed parameters{
##   matrix[n,1] mu;
##   vector[n] mu2;
##   mu = X * beta + alpha; // regression
```

```
##   mu2 = to_vector(mu);   // normal distribution
## }
##
## model {
##   // priors
##   for (i in 1:k) {            // normal priors for predictors
##     beta[i] ~ normal(0, 100);
##   }
##   sigma ~ gamma(2, 0.1);      // gamma prior for standard deviation
##
##   // likelihood
##   Y ~ normal(mu2, sigma);
## }
##
## generated quantities {
##   vector[n] log_lik;
##   for (i in 1:n)
##     log_lik[i] = normal_lpdf(Y[i] | mu2[i], sigma);
## }
```

```
dat <- list(n = length(data[[1]]),
            k = 5,
            X = subset(data, select=-scaled_sound_pressure_level),
            Y = data$scaled_sound_pressure_level
            )

fit_1 <- stan(file = model_code_1, data = dat, refresh = 0)
```

Rhat check

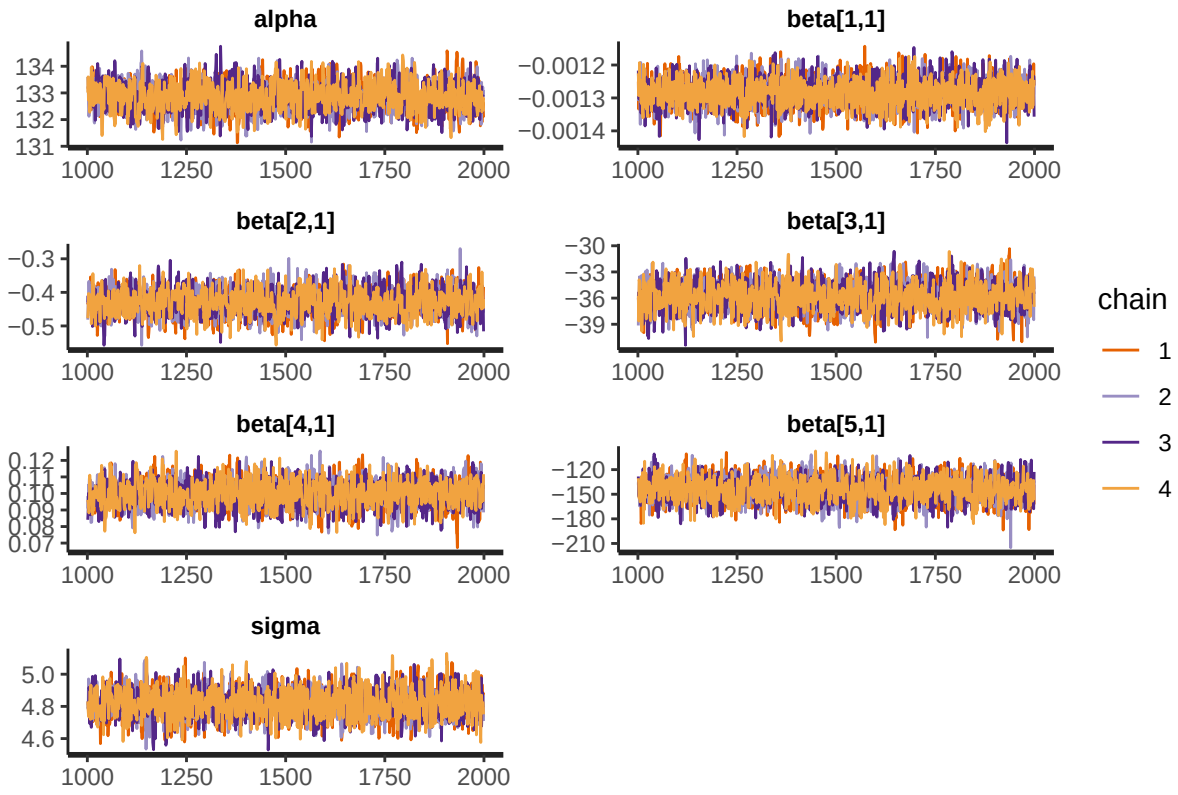
```
print(fit_1, pars = c("alpha","beta","sigma"))
```

```
## Inference for Stan model: model1.
## 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%     25%     50%     75%   97.5% n_eff
## alpha      132.86    0.01  0.53  131.84  132.50  132.86  133.21  133.91  1665
## beta[1,1]    0.00    0.00  0.00    0.00    0.00    0.00    0.00    0.00  3747
## beta[2,1]   -0.43    0.00  0.04   -0.50   -0.46   -0.43   -0.40   -0.35  2026
## beta[3,1]  -35.77    0.04  1.60  -38.89  -36.85  -35.80  -34.67  -32.68  1809
## beta[4,1]    0.10    0.00  0.01    0.08    0.09    0.10    0.11    0.11  1934
## beta[5,1] -144.13    0.30 14.81 -172.91 -154.57 -143.91 -133.94 -115.39  2482
## sigma        4.82    0.00  0.09    4.65    4.76    4.82    4.87    4.99  2224
##           Rhat
## alpha        1
## beta[1,1]    1
## beta[2,1]    1
## beta[3,1]    1
## beta[4,1]    1
## beta[5,1]    1
## sigma        1
##
```

```
## Samples were drawn using NUTS(diag_e) at Thu Nov 19 03:23:43 2020.
## 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).
```

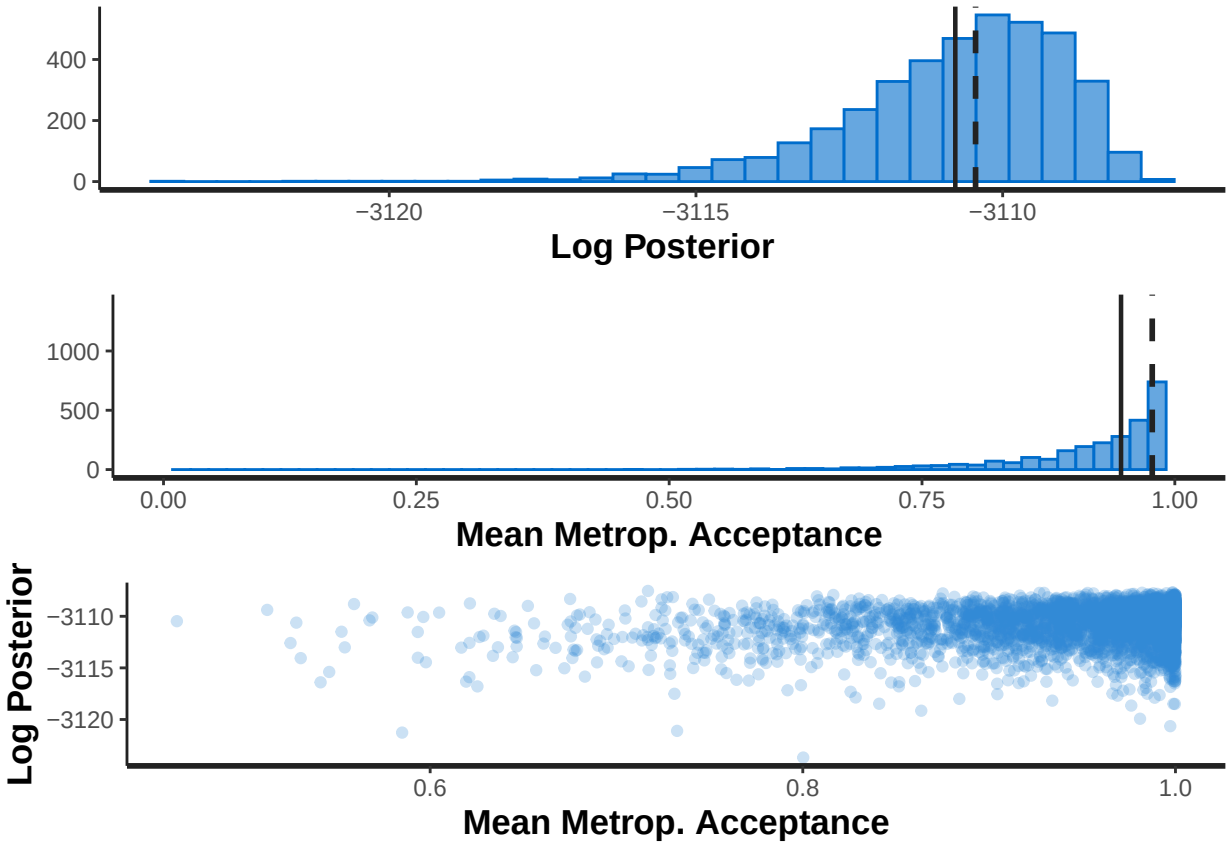
Convergence visual check

```
traceplot(fit_1, pars = c("alpha","beta","sigma"), inc_warmup = FALSE,  nrow = 4)
```




```
stan_diag(fit_1)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




Loo check

```
log_lik_1 <- extract_log_lik(fit_1, merge_chains = FALSE)
r_eff_1 <- relative_eff(exp(log_lik_1))
loo_1 <- loo(log_lik_1, r_eff = r_eff_1)
print(loo_1$estimates)
```

```
##              Estimate         SE
## elpd_loo -4497.497314 30.9624507
## p_loo        7.939411  0.4786785
## looic     8994.994628 61.9249014
```

```
pareto_k_table(loo_1)
```

```
##
## All Pareto k estimates are good (k < 0.5).
```

In here, the effective number of parameters is 8

## Model 2)

The same code as Model 1 but with different priors

```
model_code_2 = root("models/model2.stan")
writeLines(readLines(model_code_2))
```

```
## data {
##   int<lower=0> n; // number of data items
##   int<lower=0> k; // number of predictors
##   matrix[n,k] X;  // predictor matrix
##   vector[n] Y;    // outcome vector
## }
##
## parameters {
##   real alpha;             // intercept
##   matrix[k,1] beta;       // coefficients for predictors
##   real<lower=0> sigma;    // error scale
##   real<lower=0> mu0;      // prior mean, half-normal prior
##   real<lower=0> musigma0; // prior std
## }
##
## transformed parameters{
##   matrix[n,1] mu;
##   vector[n] mu2;
##   mu = X * beta + alpha; // regression
##   mu2 = to_vector(mu);   // normal distribution
## }
##
## model {
##   // hyperprior
##   mu0 ~ normal(0, 10);            // weakly informative prior
##   musigma0 ~ inv_chi_square(0.1); // weakly informative prior
##
##   // priors
##   for (i in 1:k) {                    // weakly informative prior for predictors
##     beta[i] ~ normal(mu0, musigma0);
##   }
##   sigma ~ inv_chi_square(0.1);       // weakly informative prior for standard deviation
##
##   // likelihood
##   Y ~ normal(mu2, sigma);
## }
##
## generated quantities {
##   vector[n] log_lik;
##
##   for (i in 1:n)
##     log_lik[i] = normal_lpdf(Y[i] | mu2[i], sigma);
## }
```

```
dat <- list(n = length(data[[1]]),
            k = 5,
            X = subset(data, select=-scaled_sound_pressure_level),
            Y = data$scaled_sound_pressure_level
            )
```

```
fit_2 <- stan(file = model_code_2, data = dat, refresh = 0)
```

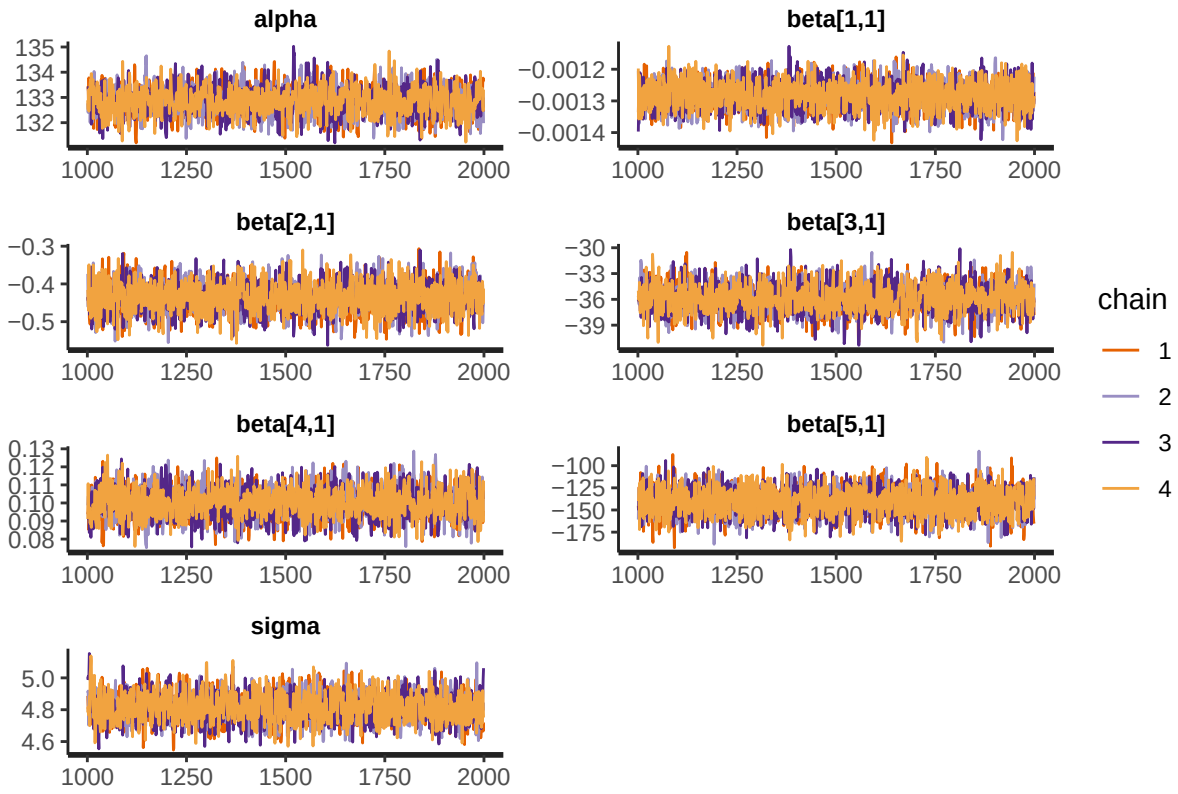
Rhat check

```
print(fit_2, pars = c("alpha","beta","sigma"))
```

```
## Inference for Stan model: model2.
## 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%     25%     50%     75%   97.5% n_eff
## alpha      132.86    0.01  0.54  131.77  132.50  132.87  133.24  133.91  1761
## beta[1,1]    0.00    0.00  0.00    0.00    0.00    0.00    0.00    0.00  3873
## beta[2,1]   -0.44    0.00  0.04   -0.51   -0.46   -0.44   -0.41   -0.36  2105
## beta[3,1]  -35.88    0.03  1.58  -38.88  -36.95  -35.94  -34.81  -32.70  2197
## beta[4,1]    0.10    0.00  0.01    0.08    0.09    0.10    0.11    0.12  2258
## beta[5,1] -140.22    0.29 14.79 -169.21 -149.93 -139.97 -130.49 -111.27  2678
## sigma        4.81    0.00  0.09    4.65    4.75    4.81    4.87    4.99  2280
##           Rhat
## alpha        1
## beta[1,1]    1
## beta[2,1]    1
## beta[3,1]    1
## beta[4,1]    1
## beta[5,1]    1
## sigma        1
##
## Samples were drawn using NUTS(diag_e) at Thu Nov 19 03:26:52 2020.
## 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).
```

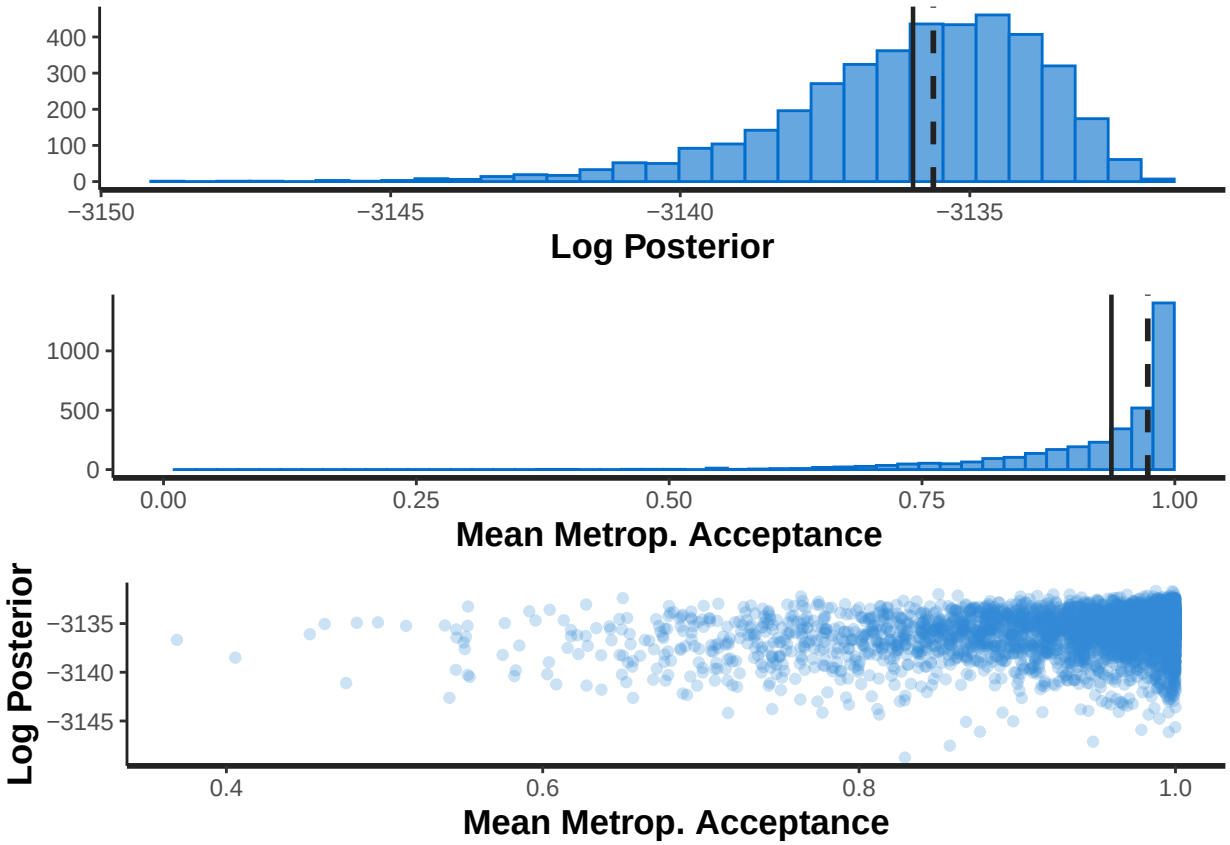
Convergence visual check

```
traceplot(fit_2, pars = c("alpha","beta","sigma"), inc_warmup = FALSE,  nrow = 4)
```




```
stan_diag(fit_2)
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```




Loo check

```
log_lik_2 <- extract_log_lik(fit_2, merge_chains = FALSE)
r_eff_2 <- relative_eff(exp(log_lik_2))
loo_2 <- loo(log_lik_2, r_eff = r_eff_2)
print(loo_2$estimates)
```

```
##              Estimate         SE
## elpd_loo -4497.817594 30.9689032
## p_loo        8.259098  0.5028464
## looic     8995.635188 61.9378064
```

```
pareto_k_table(loo_2)
```

```
##
## All Pareto k estimates are good (k < 0.5).
```

## Model comparision)

```
comp <- loo_compare(loo_1, loo_2)
print(comp)
```

```
##        elpd_diff se_diff
## model1  0.0       0.0
## model2 -0.3       0.3
```