DS4420_Project_Bayesian

Jing Cheng

2025-11-20

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(data.table)
library(brms)
```

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

```
## Loading 'brms' package (version 2.23.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

```
library(scoringRules)
```

```
# Load and format data
df <- read.csv("household_power_cleaned.csv")
setDT(df)
df[, Datetime := as.POSIXct(Datetime, format = "%Y-%m-%d %H:%M:%S", tz = "UTC")]
```

```
# Subsample for faster modeling
set.seed(123)
subsample_size <- 100000
df_sub <- df[sample(.N, subsample_size)]

# Train/test split
set.seed(123)
n <- nrow(df_sub)
train_idx <- sample(seq_len(n), size = round(0.8 * n))
train <- df_sub[train_idx]
test  <- df_sub[-train_idx]
y_test <- test$Global_active_power
```

```
# Define priors
prior <- default_prior(
  Global_active_power ~ Voltage + Global_reactive_power + Global_intensity,
```

```
  data = train,
  family = gaussian()
)

# Fit the Bayesian regression model
power_brm <- brm(
  Global_active_power ~ Voltage + Global_reactive_power + Global_intensity,,
  family = gaussian(),
  data = train,
  chains = 4,
  iter = 3000,
  warmup = 1000,
  thin = 2,
  prior = prior,
  control = list(adapt_delta = 0.95)
)
```

```
## Compiling Stan program...
```

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

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 17.0.0 (clang-1700.4.4.1)'
## using SDK: 'MacOSX26.1.sdk'
## clang -arch arm64 -std=gnu2x -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
##   679 | #include <cmath>
##       |          ^~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00023 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.3 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 1: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 1: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 1: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 1: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 1: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 1: Iteration: 2800 / 3000 [ 93%]  (Sampling)
```

```
## Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 17.889 seconds (Warm-up)
## Chain 1:                44.234 seconds (Sampling)
## Chain 1:                62.123 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000213 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.13 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 2: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 2: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 2: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 2: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 2: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 2: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 19.019 seconds (Warm-up)
## Chain 2:                43.003 seconds (Sampling)
## Chain 2:                62.022 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000213 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.13 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 3: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 3: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 3: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 3: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 3: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 3: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 17.367 seconds (Warm-up)
## Chain 3:                43.143 seconds (Sampling)
```

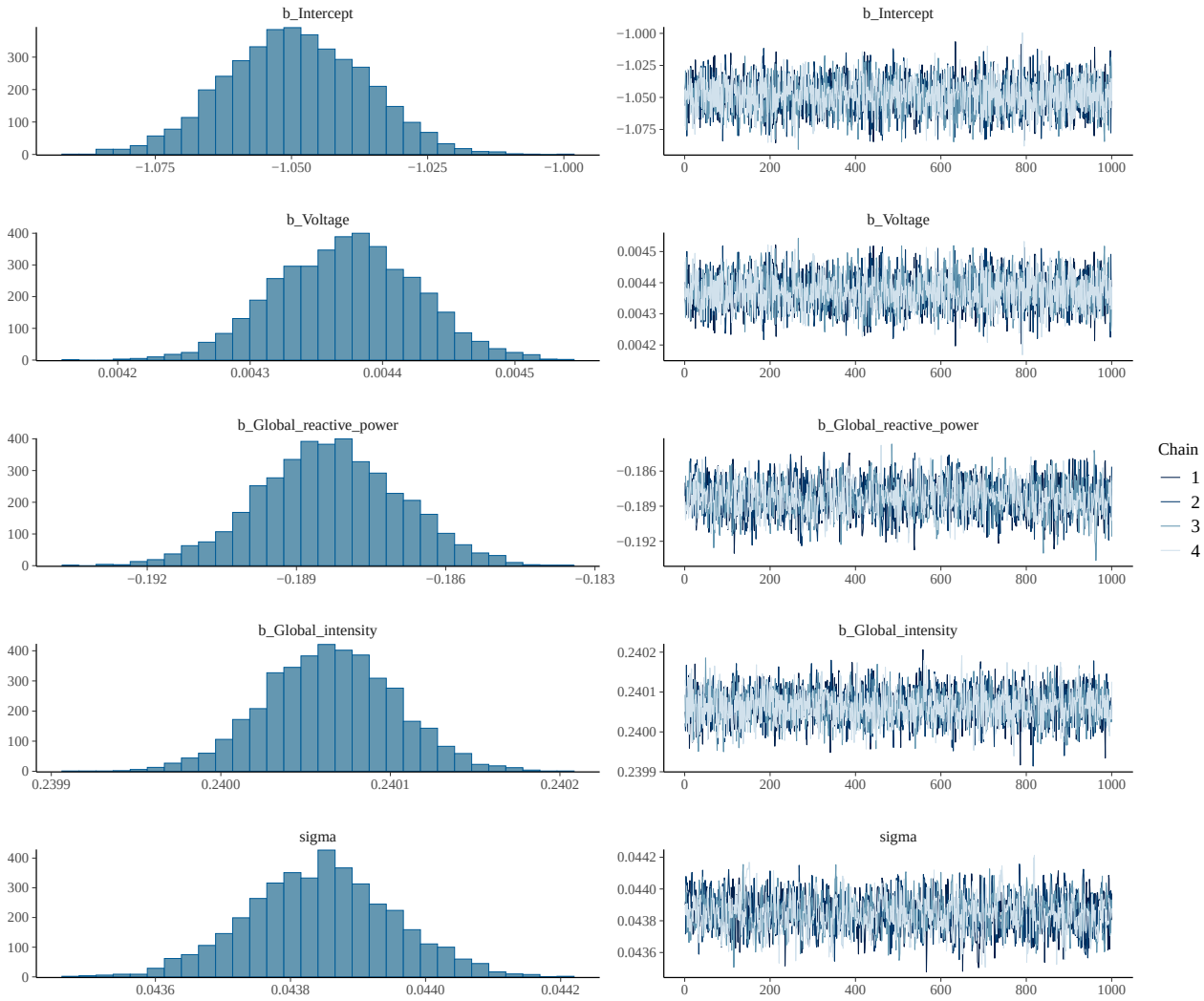
```
## Chain 3:                60.51 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000229 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.29 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 4: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 4: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 4: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 4: Iteration: 1300 / 3000 [ 43%]  (Sampling)
## Chain 4: Iteration: 1600 / 3000 [ 53%]  (Sampling)
## Chain 4: Iteration: 1900 / 3000 [ 63%]  (Sampling)
## Chain 4: Iteration: 2200 / 3000 [ 73%]  (Sampling)
## Chain 4: Iteration: 2500 / 3000 [ 83%]  (Sampling)
## Chain 4: Iteration: 2800 / 3000 [ 93%]  (Sampling)
## Chain 4: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 23.881 seconds (Warm-up)
## Chain 4:                51.477 seconds (Sampling)
## Chain 4:                75.358 seconds (Total)
## Chain 4:
```

```
# Summary & convergence diagnostics
summary(power_brm)
```

```
##  Family: gaussian
##   Links: mu = identity
## Formula: Global_active_power ~ Voltage + Global_reactive_power + Global_intensity
##    Data: train (Number of observations: 80000)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 2;
##          total post-warmup draws = 4000
##
## Regression Coefficients:
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                -1.05      0.01    -1.07    -1.02 1.00     3653
## Voltage                   0.00      0.00     0.00     0.00 1.00     3642
## Global_reactive_power    -0.19      0.00    -0.19    -0.19 1.00     2211
## Global_intensity          0.24      0.00     0.24     0.24 1.00     3695
##                       Tail_ESS
## Intercept                 3230
## Voltage                   3315
## Global_reactive_power     2209
## Global_intensity          3597
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.04      0.00     0.04     0.04 1.00     1487     1622
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
```

```
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
plot(power_brm)
```



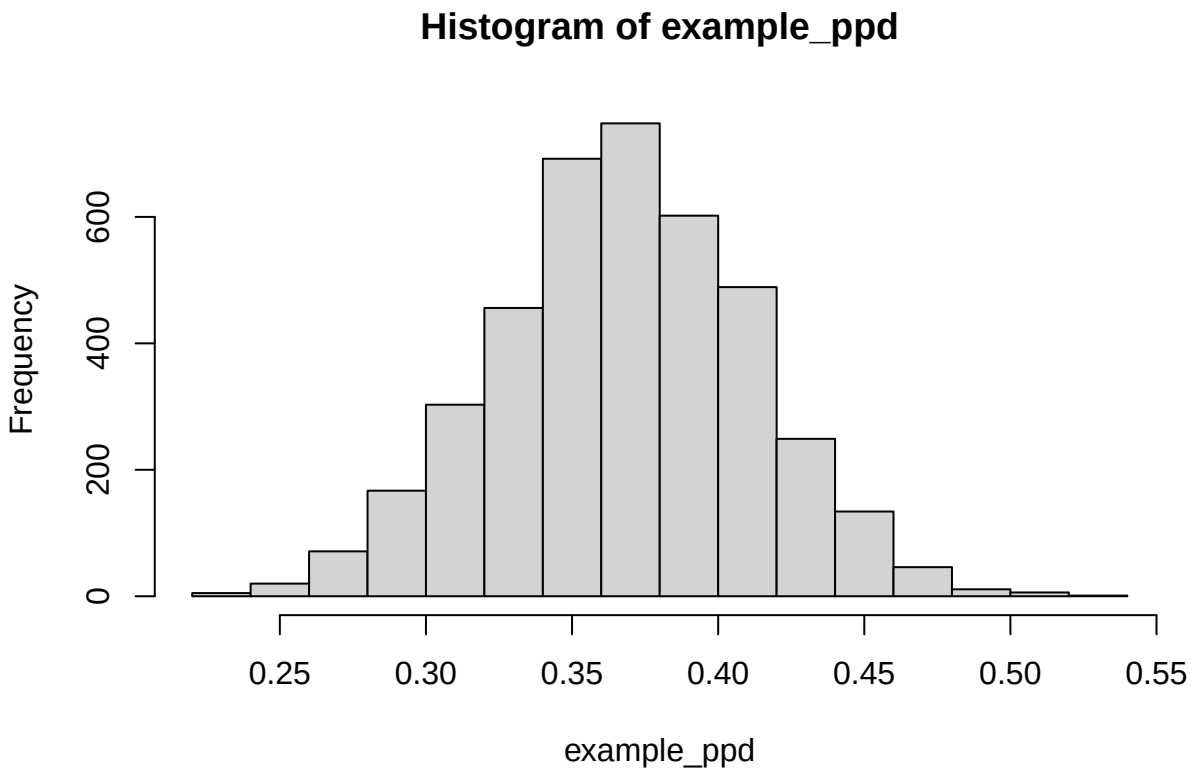

```
bayes_R2(power_brm)
```

```
##      Estimate    Est.Error      Q2.5     Q97.5
## R2 0.9983007 4.942526e-07 0.9982997 0.9983016
```

```
# Posterior predictive distribution
pred_draws <- posterior_predict(power_brm, newdata = test)
dim(pred_draws)
```

```
## [1]  4000 20000
```

```
# Example posterior predictive distribution for first test case
example_ppd <- pred_draws[, 1]
hist(example_ppd)
```




```r
mean(example_ppd)
```

```
## [1] 0.3676449
```

```r
# Point predictions
point_pred <- colMeans(pred_draws)
```

```r
# Accuracy metrics
rmse <- sqrt(mean((point_pred - y_test)^2))
mae  <- mean(abs(point_pred - y_test))

# CRPS
crps_vals <- sapply(seq_along(y_test), function(j) {
  crps_sample(y_test[j], pred_draws[, j])
})
mean_crps <- mean(crps_vals)

# Predictive intervals
lower <- apply(pred_draws, 2, quantile, probs = 0.025)
upper <- apply(pred_draws, 2, quantile, probs = 0.975)
coverage95 <- mean(y_test >= lower & y_test <= upper)
interval_width <- mean(upper - lower)

cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.04349098
```

```r
cat("MAE:", mae, "\n")
```

```
## MAE: 0.02884011
```

```r
cat("CRPS:", mean_crps, "\n")
```

```
## CRPS: 0.02150204
```

```r
cat("Coverage (95%):", coverage95, "\n")
```

```
## Coverage (95%): 0.9701
```

```r
cat("Avg Interval Width:", interval_width, "\n")
```

```
## Avg Interval Width: 0.171693
```

```r
# Posterior summaries
posterior_summary(power_brm)
```

```
##                              Estimate    Est.Error          Q2.5         Q97.5
## b_Intercept             -1.049269e+00 1.276279e-02 -1.074205e+00 -1.024850e+00
## b_Voltage                4.372565e-03 5.267808e-05  4.271244e-03  4.475775e-03
## b_Global_reactive_power -1.883303e-01 1.413750e-03 -1.911453e-01 -1.855613e-01
## b_Global_intensity       2.400645e-01 3.869169e-05  2.399885e-01  2.401404e-01
## sigma                    4.384457e-02 1.081458e-04  4.363357e-02  4.406070e-02
## Intercept                1.095310e+00 1.551639e-04  1.095010e+00  1.095606e+00
## lprior                  -3.167416e+00 1.621552e-05 -3.167447e+00 -3.167385e+00
## lp__                     1.366475e+05 1.542958e+00  1.366438e+05  1.366496e+05
```