BDA - Assignment 9

Anonymous

Contents

```r
library(markmyassignment)
```

```
## Warning: package 'markmyassignment' was built under R version 4.1.3
```

```r
assignment_path <-
paste("https://github.com/avehtari/BDA_course_Aalto/",
"blob/master/assignments/tests/assignment9.yml", sep="")
set_assignment(assignment_path)
```

```
## Assignment set:
## assignment9: Bayesian Data Analysis: Assignment 9
## The assignment contain the following task:
## - utility
```

```r
# To check your code/functions, just run
#mark_my_assignment()
```

Decision analysis for the factory data

Decision analysis for the factory data (3p)

This exercise is an example of a decision analysis (DA). In a broad context, this means optimizing over different decisions that lead to different outcomes that all have different utilities. In a Bayesian context, this means using posterior distributions to make decisions.

In this exercise, you work as a data analyst in the company that owns the six machines that have produced the data in the factory dataset. To access the data, just use:

```r
library(aaltobda)
data("factory")
```

Your task is to decide whether or not to buy a new (7th) machine for the company. The decision should be based on our best knowledge about the machines.

The following is known about the production process:

- The given data contains quality measurements of single products from the six machines that are ordered from the same seller. (columns: different machines, rows: measurements)
- Customers pay 200 euros for each product. If the quality of the product is below 85, the product cannot be sold. All the products that have sufficient quality are sold.
- Raw-materials, the salary of the machine user and the usage cost of the machine for each product cost 106 euros in total. Usage cost of the machine also involves all investment and repair costs divided by the number of products a machine can create. So there is no need to take the investment cost into account as a separate factor.
- The only thing the company owner cares about is money. Thus, as a utility function, use the profit of a new product from a machine.

As noticed in the previous assignment, the hierarchical model fits best with the dataset, so use it to compute the utilities. The assumptions for the hierarchical model are the same as in assignment 7 and 8. If you did things correctly then, solving the assignment only requires you to change the "generated quantities"-block in the Stan-code to compute the correct predictive samples for products of all 7 (= 6 + 1) machines.

Your task is the following:

1.

For each of the six machines, compute and report the expected utility of one product of that machine. Below is a test case on how the utility function should work (and that you can test with markmyassignment)
Note! The expected utility should be computed from the quality measurements of the products of each machine, not from the means of the qualities of each machine.
Note! This is just a test case to test that your utility function works. In the report, you should report the expected utility using your posterior draws from Stan.
Note! The value below is only a test case, you need to use correct draws from the predictive distribution in the final report.

Let x be the quality of each product. We can define a piecewise binary function, where $piecewise(x > 85) = 1$ which means the customer buys this product, and $piecewise(x \leq 85) = 0$, which means the customer will not buy this product. Whether the customers buy or not buy the product, all products cost 106 euros to produce. Hence, the expected utility formula of one product of the machine is:

$$utility(x) = 200 * piecewise(x > 85) - 106$$

```r
utility <- function(draws){
  result = 0
  for (i in 1:length(draws)) {
    if (draws[i] > 85) {
      purchased = 1
    } else {
      purchased = 0
    }
    result = result + purchased * 200 - 106
  }
  result = result/length(draws)
  return(result)
}
y_pred <- c(123.80, 85.23, 70.16, 80.57, 84.91)
utility(draws = y_pred)
```

```
## [1] -26
```

```
mark_my_assignment()
```

```
## Warning: package 'testthat' was built under R version 4.1.3
```

```
## v | F W S  OK | Context
##
## / |         0 | task-1-subtask-1-tests
## / |         0 | utility()
## v |         5 | utility()
##
## == Results =====================================================================
## [ FAIL 0 | WARN 0 | SKIP 0 | PASS 5 ]
## Good work!
```

2.

Rank the machines based on the expected utilities. In other words order the machines from worst to best: X(worst), X, X, X, X, X(best), where each X should be a number of a machine. Also briefly explain what the utility values tell about the quality of these machines. E.g. Tell which machines are profitable and which are not (if any).

The hierarchical stan model is chosen for this task
The priors are chosen from assignment 7, part 2d
$\mu_0 \sim Normal(0, 10)$
$\sigma_0 \sim Gamma(1, 1)$
$\sigma \sim Gamma(1, 1)$

```
"
data {
  int<lower=0> N; // number of measurements per machine
  int<lower=0> K; // number of machines
  array[N] vector[K] y; // An array of vectors containing the table data
}

parameters {
  vector[K] mu;
  real sigma;
  real mu_0;
  real sigma_0;
  real mu_ypred7;
}

model {
  mu_0 ~ normal(0, 10);
  sigma_0 ~ gamma(1, 1);
  sigma ~ gamma(1, 1);
  mu_ypred7 ~ normal(mu_0, sigma_0);
  for (k in 1:K) {
    mu[k] ~ normal(mu_0, sigma_0);
  }
  for (k in 1:K) {
    y[, k] ~ normal(mu[k], sigma);
  }
}

generated quantities {

  array[K+1] real ypred;
  for (k in 1:K) {
    ypred[k] = normal_rng(mu[k] , sigma);
  }

  real ypred7 = normal_rng(mu_ypred7, sigma);
  ypred[K+1] = ypred7;
}

"
```

```
## [1] "\ndata {\n  int<lower=0> N; // number of measurements per machine\n  int<lower=0> K; // number
```

```
stan_data <- list(
  y = factory,
  N = nrow(factory),
  K = ncol(factory)
)

file_hierarchical <- file.path("C:/Users/nguye/Desktop/BDA/assignments/assignment 9", "model_hierarchica
model_hierarchical <- cmdstan_model(file_hierarchical)
model_hierarchical$compile(quiet = FALSE)
result_hierarchical <- model_hierarchical$sample(data = stan_data, chains=4, show_messages=FALSE)
```

```
## Running MCMC with 4 sequential chains...

## Chain 1 Rejecting initial value:

## Chain 1   Error evaluating the log probability at the initial value.

## Chain 1 Exception: gamma_lpdf: Random variable is -1.17515, but must be positive finite! (in 'C:/User
## Chain 1 Exception: gamma_lpdf: Random variable is -1.17515, but must be positive finite! (in 'C:/User
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.4 seconds.
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 1.4 seconds.

## Chain 3 Rejecting initial value:

## Chain 3   Error evaluating the log probability at the initial value.

## Chain 3 Exception: gamma_lpdf: Random variable is -1.50197, but must be positive finite! (in 'C:/Use
## Chain 3 Exception: gamma_lpdf: Random variable is -1.50197, but must be positive finite! (in 'C:/Use
## Chain 3 Rejecting initial value:

## Chain 3   Error evaluating the log probability at the initial value.

## Chain 3 Exception: gamma_lpdf: Random variable is -1.56092, but must be positive finite! (in 'C:/Use
## Chain 3 Exception: gamma_lpdf: Random variable is -1.56092, but must be positive finite! (in 'C:/Use

## Chain 3 Rejecting initial value:

## Chain 3   Error evaluating the log probability at the initial value.

## Chain 3 Exception: gamma_lpdf: Random variable is -0.479742, but must be positive finite! (in 'C:/Us
## Chain 3 Exception: gamma_lpdf: Random variable is -0.479742, but must be positive finite! (in 'C:/Us

## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 1.5 seconds.

## Chain 4 Rejecting initial value:

## Chain 4   Error evaluating the log probability at the initial value.

## Chain 4 Exception: gamma_lpdf: Random variable is -1.21894, but must be positive finite! (in 'C:/Use
## Chain 4 Exception: gamma_lpdf: Random variable is -1.21894, but must be positive finite! (in 'C:/Use

## Chain 4 Rejecting initial value:

## Chain 4   Error evaluating the log probability at the initial value.

## Chain 4 Exception: gamma_lpdf: Random variable is -0.0829367, but must be positive finite! (in 'C:/U
## Chain 4 Exception: gamma_lpdf: Random variable is -0.0829367, but must be positive finite! (in 'C:/U
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 1.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.5 seconds.
## Total execution time: 6.6 seconds.
```

```
## Warning: 73 of 4000 (2.0%) transitions ended with a divergence.
## See https://mc-stan.org/misc/warnings for details.
```

```
ypred1 <- utility(draws = result_hierarchical$draws("ypred[1]"))
ypred2 <- utility(draws = result_hierarchical$draws("ypred[2]"))
ypred3 <- utility(draws = result_hierarchical$draws("ypred[3]"))
ypred4 <- utility(draws = result_hierarchical$draws("ypred[4]"))
ypred5 <- utility(draws = result_hierarchical$draws("ypred[5]"))
ypred6 <- utility(draws = result_hierarchical$draws("ypred[6]"))
cat("\nThe chosen priors are from assignment 7 part 2d")
```

```
##
## The chosen priors are from assignment 7 part 2d
```

```
cat("\nmu0 ~ Normal(0, 10)")
```

```
##
## mu0 ~ Normal(0, 10)
```

```
cat("\nsigma0 ~ Gamma(1,1)")
```

```
##
## sigma0 ~ Gamma(1,1)
```

```
cat("\nsigma ~ Gamma(1,1)")
```

```
##
## sigma ~ Gamma(1,1)
```

```
cat("\nThe expected utilities of the six machines are")
```

```
##
## The expected utilities of the six machines are
```

```
cat("\nMachine      1  |  2  |  3  |  4   |  5   |   6  \n")
```

```
##
## Machine      1  |  2  |  3  |  4   |  5   |   6
```

```
paste(" ",ypred1, ypred2, ypred3, ypred4, ypred5, ypred6, sep="  ")
```

```
## [1] "   -53.15  64.85  -4.05  74.35  6.95  -11.8"
```

From the results, we can rank the machines as follows: 1(worst), 6, 3, 5, 2, 4(best)

Interpretation of the predicted utility: it predicts how the future products made by each machine are likely to earn profits. We can see that negative utility values means this machine will be likely to incur loss, while positive ones will belikely to earn profits. In this case, machines 1, 3 and 6 are likely to incur loss, while machines 2, 4, 5 are likely to earn profits.

3.

Compute and report the expected utility of the products of a new (7th) machine.

```
ypred7 <- utility(draws = result_hierarchical$draws("ypred[7]"))
cat("\nThe chosen priors are from assignment 7 part 2d")
```

```
##
## The chosen priors are from assignment 7 part 2d
```

```
cat("\nmu0 ~ Normal(0, 10)")
```

```
##
## mu0 ~ Normal(0, 10)
```

```
cat("\nsigma0 ~ Gamma(1,1)")
```

```
##
## sigma0 ~ Gamma(1,1)
```

```
cat("\nsigma ~ Gamma(1,1)")
```

```
##
## sigma ~ Gamma(1,1)
```

```
cat("\nThe expected utility of the seventh machine is",ypred7)
```

```
##
## The expected utility of the seventh machine is -59.2
```

4.

Based on your analysis, discuss briefly whether the company owner should buy a new (7th) machine.

Based on my analysis, the company owner should not buy a new 7th machine because its expected utility value is negative, so it is likely to incur loss for the company.

5.

As usual, remember to include the source code for both Stan and R (or Python).

I have already included everything above.