Assignment 6

anonymous

1 General information

2 Stan warm-up: linear model of BDA retention with Stan (2 points)

2.1 (b)

Full corrected stan code.

Fixes:
* sigma has now a lower bound at 0 .
* added semicolon to the transformed parameter statement that was missing
* y_pred samples random normal observations with mean “mu_pred” instead of “mu”.

```
data {
    // number of data points
    int<lower=0> N;
    // covariate / predictor
    vector[N] x;
    // observations
    vector[N] y;
    // number of covariate values to make predictions at
    int<lower=0> no_predictions;
    // covariate values to make predictions at
    vector[no_predictions] x_predictions;
}
parameters {
    // intercept
    real alpha;
    // slope
    real beta;
    // the standard deviation should be constrained to be positive
    real<lower=0> sigma;
```

```
}
transformed parameters {
    // deterministic transformation of parameters and data
    vector[N] mu = alpha + beta * x ;// linear model
}
model {
    // observation model / likelihood
    y ~ normal(mu, sigma);
}
generated quantities {
    // compute the means for the covariate values at which to make predictions
    vector[no_predictions] mu_pred = alpha + beta * x_predictions;
    // sample from the predictive distribution, a normal(mu_pred, sigma).
    array[no_predictions] real y_pred = normal_rng(to_array_1d(mu_pred), sigma);
}
```

Plotting happens here:

```
  ggplot() +
    # scatter plot of the training data:
    geom_point(
      aes(x, y, color=assignment),
      data=data.frame(x=assignment, y=propstudents, assignment="1-8")
  ) +
    # scatter plot of the test data:
    geom_point(
      aes(x, y, color=assignment),
      data=data.frame(x=no_assignments, y=propstudents9, assignment="9")
  ) +
    # you have to tell us what this plots:
    geom_line(aes(x,y=value,linetype=pct), data=mu_quantiles_df, color='grey', linewidth=1.
    # you have to tell us what this plots:
    geom_line(aes(x,y=value,linetype=pct), data=y_quantiles_df, color='red') +
    # adding xticks for each assignment:
    scale_x_continuous(breaks=1:no_assignments) +
    # adding labels to the plot:
    labs(y="assignment submission %", x="assignment number") +
    # specifying that line types repeat:
    scale_linetype_manual(values=c(2,1,2)) +
    # Specify colours of the observations:
    scale_colour_manual(values = c("1-8"="black", "9"="blue")) +
    # remove the legend for the linetypes:
    guides(linetype="none")
```

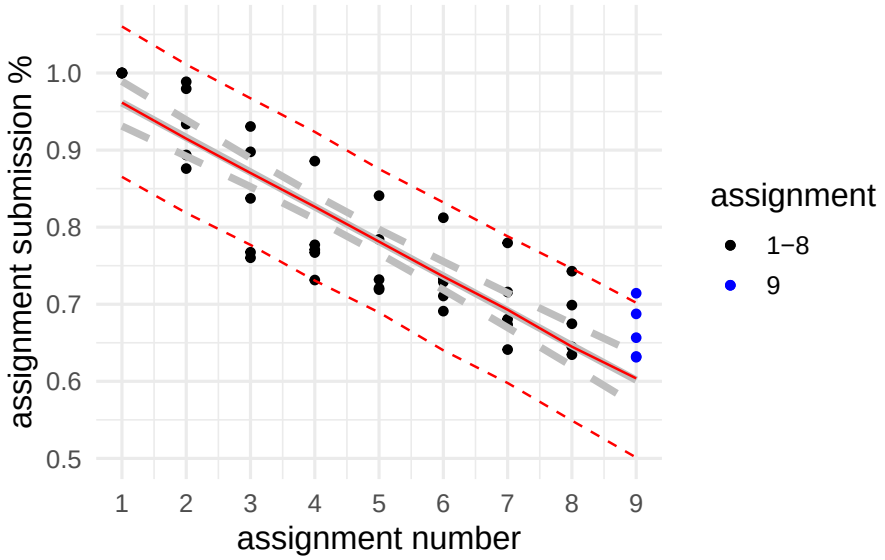


Figure 1: Mean value + 0.05 and 0.95 percentiles for mu (grey) and for the predictions (red). Data is plotted as dots.

2.2 (c)

- The solid red line is the median of the distribution of the predictions of Y.
  The dashed lines are the 0.05 and 0.95 percentiles respectively.
  The grey lines are the same but for the distribution of Mu, the mean of Y given X.

```
summary(draws_df$beta)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.06054 -0.04754 -0.04494 -0.04484 -0.04210 -0.02942
```

- The model estimates that with each assignment, close to 4.4 percentage points of students stop submitting assignments.
- Based on the plot, the model does a "not that bad" job, 3 out of 4 values for assignment 9 are withing 90% interval although all of them are above the mean prediction. Could be better.
- Maybe we could use a binomial likelihood for proportions instead of a normal distribution? Also, include some polynomial component to the predictor to not assume a linear relationship?

3 Generalized linear model: Bioassay with Stan (4 points)

3.1 (d)

```r
data("bioassay")

bioassay_data = list(M=length(bioassay$y),
                     N=bioassay$n,
                     x=bioassay$x,
                     y=bioassay$y
                     )
```

Stan model:

```
data {
  int<lower=0> M;
  vector[M] x; // Dose predictor
  int N[M]; // N subjects
  int y[M]; // Deaths.


}

parameters {
    // intercept
    real alpha;
    // slope
    real beta;
}


transformed parameters {
    // deterministic transformation of parameters and data
    vector[M] logit_p = (alpha + beta * x) ;// linear model
}

model {
  vector[2] ab; // alpha and beta
  vector[2] mu; // prior mean
  matrix[2,2] Sigma = [[4,12],
                       [12,100]]; // prior covariance matrix

  ab = [alpha, beta]';
```

```
  mu = [0,10]';

  ab ~ multi_normal(mu, Sigma); // prior on alpha and beta

  y ~ binomial_logit(N, logit_p); // likelihood
}
```

```
# This reads the file at the specified path and tries to compile it.
# If it fails, an error is thrown.
bioassay_model = cmdstan_model("./assignment6_bioassay.stan")
# This "out <- capture.output(...)" construction suppresses output from cmdstanr
# See also https://github.com/stan-dev/cmdstanr/issues/646
out <- capture.output(
    # Sampling from the model happens here:
    fit <- bioassay_model$sample(data=bioassay_data, refresh=0, show_messages=FALSE)
)
```

```
# We store the draws
bioassay_draws_df = fit$draws(format="draws_df")
```

```
fit$summary()
```

```
# A tibble: 7 x 10
  variable     mean median    sd   mad      q5    q95  rhat ess_bulk ess_tail
  <chr>       <dbl>  <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>    <dbl>
1 lp__       -9.91  -9.61  1.00  0.743 -11.9   -8.94   1.00    1637.    2033.
2 alpha       0.973  0.944 0.895 0.919  -0.391  2.44   1.00    1254.    1713.
3 beta       10.6   10.1   4.59  4.55    4.07  19.2    1.00    1315.    1753.
4 logit_p[1] -8.14  -7.76  3.49  3.36  -14.7   -3.29   1.00    1534.    1931.
5 logit_p[2] -2.21  -2.08  1.12  1.06   -4.27  -0.595  1.00    2456.    2073.
6 logit_p[3]  0.443  0.422 0.782 0.787  -0.798  1.72   1.00    1492.    1953.
7 logit_p[4]  8.71   8.25  3.94  3.93    3.02  16.0    1.01    1189.    1584.
```

3.2 (e)

```
# Useful functions: rhat_basic (from posterior)
warmup = 500
alpha_r = bioassay_draws_df %>%
  group_by(.chain) %>%
  filter(.iteration > warmup) %>%
  ungroup() %>%
```

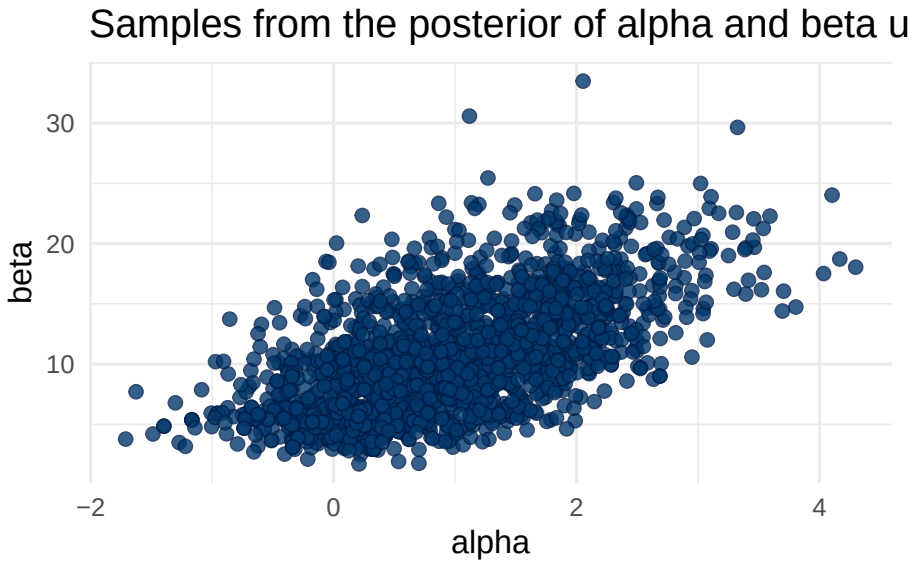
```
  select(alpha)
beta_r = bioassay_draws_df %>%
  group_by(.chain) %>%
  filter(.iteration > warmup) %>%
  ungroup() %>%
  select(beta)

alpha_rhat = rhat_basic(alpha_r)
beta_rhat = rhat_basic(beta_r)
```

I used rhat_basic() which uses the more recent version of Rhat. $\hat{R}$ for α is 1 and
$\hat{R}$ for β is 1.

The idea of $\hat{R}$ is to roughly estimate by how much the variance/scale of the value estimated would be reduced if we kept simulating more values going towards $n- >$ inf Values really close to 1 suggest that the scale reduction would be minimal.

3.3 (f)

```
chains_wo_warmup = bioassay_draws_df %>% group_by(.chain) %>%
  filter(.iteration > warmup)
p = mcmc_scatter(chains_wo_warmup, pars=c("alpha", "beta"))
p + labs(title = "Samples from the posterior of alpha and beta using HMC")
```



3.4 (g)

- Windows 10 Pro.
- R.
- CmdStanR
- No problems installing. CmdStanR automatically provided the command to fix the only error faced.
- I don't know yet the intuition / rule to decide when it's needed to use vector[N] VarName versus int VarName[N]. Encountered a few issues with data type. Vector is for float? Maybe it's really simple but it wasn't obvious to me when doing the assignment.