Assignment 6

anonymous

1 Assignment 1

1.1 Fixed linear model

```stan
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).
    vector[no_predictions] y_pred;
    for (i in 1:no_predictions) {
        y_pred[i] = normal_rng(mu_pred[i], sigma);
    }
}

```

Figure 1: Alt Text

Changed the following three syntax errors:

- Row 20: “real<upper=0> sigma” -> “real<lower=0> sigma” to ensure that the standard deviation is constrained to be positive.
- Row 25: Adding a semicolon at the end of vector[N] mu = alpha + beta * x;.
- Row 38-39:In the generated quantities block, I replaced array[no_predictions] real y_pred with vector[no_predictions] y_pred and corrected the sampling loop to compute y_pred correctly.

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=1.5) +
  # 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")
```

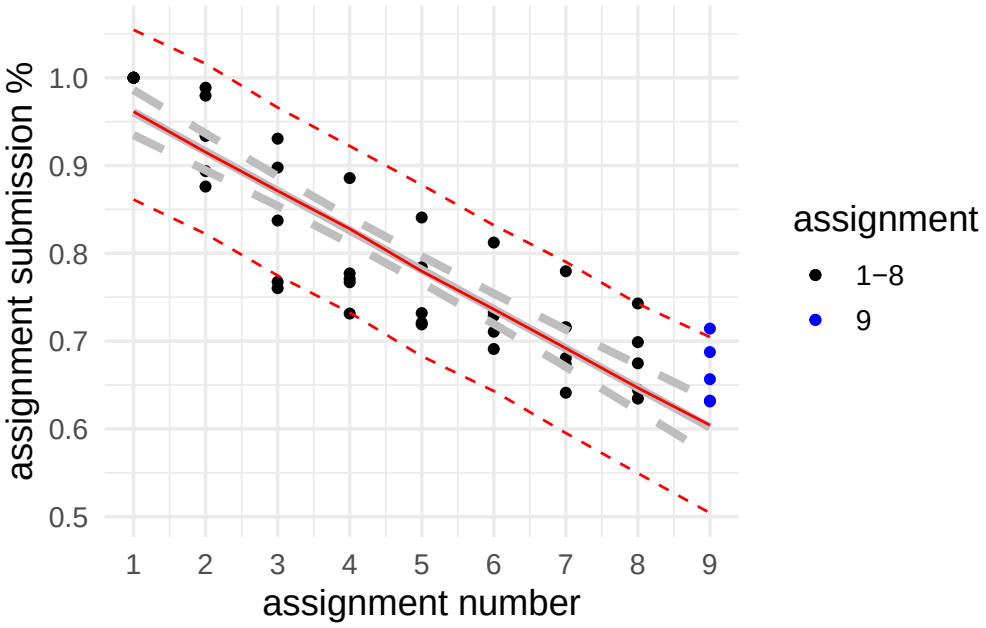


Figure 2: Describe me in your submission!

The grey lines represent the mu_pred which is the expected quantiles (5%, 50% and 95%) of the covariants (x predictions). This is computed by $ _{pred} = + * x $. The dashed lines are the 5% and 95% quantiles while the middle line is the mean.

The red lines represent the posterior distribution which are sampled from a normal distribution with mean given by mu_pred and standard deviation given by sigma. The dashed lines are the 5% and 95% quantiles while the middle line is the mean.

The difference between the red and grey lines, is that red lines use expected values as grey use expected values with incorporating uncertanty around these scpectations.

The student retention rate is represented by a downward linear slope with fewer submissions in the later weeks relativly to the older weeks. The rationale behind this phenomenon lies in is simply that students who start do not finnish the course. Why? This is more speculation and outside of the scope of this course.

The preditive power is not satisfactory with a range of about 10 percentages points but, only 1/5 points is inside the grey dashed lines and 3/5 point inside the red dashed lines.

Improving the model: - Add regularisation to the model to avoid overfit with Ridge or Lasso - Use Informed prioris based on domain knowledge - Try a different model

```
fit$cmdstan_diagnose()
```

```
Processing csv files: /var/folders/lf/v92_5zrn6k3ch09dnnm3cj1r0000gn/T/RtmpqV294t/linear_model-2023102

Checking sampler transitions treedepth.
Treedepth satisfactory for all transitions.

Checking sampler transitions for divergences.
```

```
No divergent transitions found.

Checking E-BFMI - sampler transitions HMC potential energy.
E-BFMI satisfactory.

Effective sample size satisfactory.

Split R-hat values satisfactory all parameters.

Processing complete, no problems detected.
```

2 Assignment 2

2.1 Code for "multi_normal_model.stan"

```
data {
  int<lower=0> N;   // number of data points
  vector[N] x;  // data
  vector[2] mu;  // Mean vector
  array[N] int<lower=0> n;
  array[N] int<lower=0> y;
  matrix[2,2] Sigma;  // Covariance matrix
}

parameters {
  vector[2] theta;  // The parameters we want to estimate
}

model {
  theta ~ multi_normal(mu, Sigma);  // Joint normal prior
  for (i in 1:N) {
    y[i] ~ binomial_logit(n[i],theta[1] + theta[2]*x[i]);
  }
}

```

Figure 3: Multi_normal_model

```
data("bioassay")

model = cmdstan_model("multi_normal_model.stan")

mu_a <- 0  # Mean of the first variable
mu_b <- 10  # Mean of the second variable
sigma_a <- 2  # Standard deviation of the first variable
sigma_b <- 10  # Standard deviation of the second variable
rho <- 0.6  # Correlation coefficient

cov_matrix <- matrix(c(sigma_a^2, rho * sigma_a * sigma_b, rho * sigma_a * sigma_b, sigma_b^2), nrow
mean_vector <- c(mu_a, mu_b)

data_list <- list(
  N = length(bioassay$x),
```

```
    x = bioassay$x,
    y =   bioassay$y,
    n = bioassay$n,
    mu = mean_vector,
    Sigma = cov_matrix
  )

  fit <- stan(file = "multi_normal_model.stan", data = data_list, chains = 4, iter = 2000)
```

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

```
  print(fit)
```

```
Inference for Stan model: anon_model.
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 Rhat
theta[1]  0.95    0.02 0.87 -0.64  0.33  0.89  1.51  2.77  1257    1
theta[2] 10.40    0.13 4.58  3.47  6.98  9.79 13.21 20.69  1181    1
lp__     -7.10    0.03 0.97 -9.72 -7.50 -6.81 -6.40 -6.15  1448    1

Samples were drawn using NUTS(diag_e) at Sun Oct 22 16:28:21 2023.
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).
```

```
  theta_samples <- as.data.frame(fit, pars = "theta")

  colnames(theta_samples) <- c("Alpha","Beta")
  print(paste("R_hat for Alpha", round(rhat_basic(theta_samples["Alpha"]), digits=4)))
```

```
[1] "R_hat for Alpha 1.002"
```

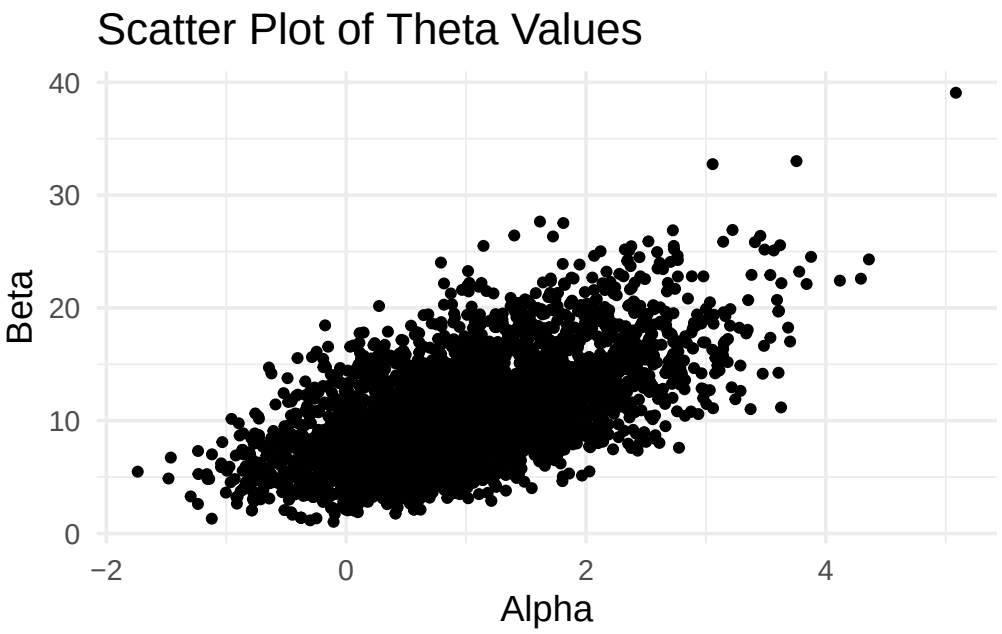
```
  print(paste("R_hat for Beta", round(rhat_basic(theta_samples["Beta"]), digits=4)))
```

```
[1] "R_hat for Beta 1.0013"
```

2.2 Interpetaion of R_hat

R_hat helps determine whether multiple chains have convereged and whether the MCMC sampler has adequately explored the target distribution. Common practice is to consider chains converged if R_hat < 1.05. The Rhat for alpha (Theta[1] in code) is 1 and beta (Theta[2]in code) is 1. Same values confirmed with with rhat_baisc() and fit$summary()

```r
# Create a scatter plot
ggplot(theta_samples, aes(x = Alpha, y = Beta)) +
  geom_point() +
  labs(x = "Alpha", y = "Beta") +
  ggtitle("Scatter Plot of Theta Values")
```



2.3 Scatter plot about draws

The scatter plot looks similar to the one made in Assignment 5.

2.4 Stan setup

- Operating system mac os
- programming eviroment: R
- Interference used: RStan and CmdStanR
- Installed locally. I had some minior porblem but was able to get everything running in 10 min
- Debugging in Stan is not fun for exapmle syntax change about arrays was frustrating. Not a lot of resources out for example youtube videos. Stans own documation is actually the best resource wish I had just started with it.

2.5 AI usage

In this assignment chatgpt was used to learn stan, help with debugging in stan and better formulate assignments