Bayesian Statistics - Project

Ksenia Lepikhina

May 8, 2019

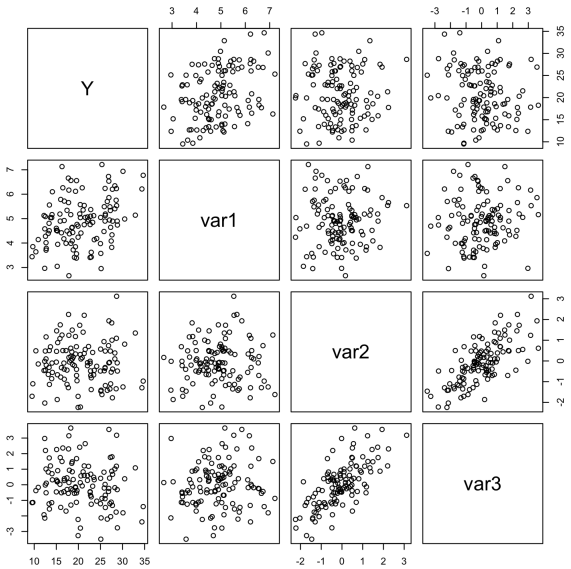
1 Introduction

There are numerous ways to analyze a dataset. While true, there is almost always a method that works best. In this project, we were given very little information about a dataset except that we may be interested in looking into mixture models. Typically, given a certain number of features, the goal is to predict the target value, y. For this project, we were given three features, $var1$, $var2$, and $var3$ and our goal was to model y.

In the following sections we will perform exploratory data analysis (EDA), dive into modeling, and compare the models to determine why the best model is indeed best.

2 Analysis

When given the dataset, the variables each appeared random and it would be easy to suggest that the data was noise. However, upon beginning the analysis, it became clear that the data was more intricate than previously assumed. The first step in the investigation was studying the correlations between each variable. This was achieved by looking at a pair plot as seen below.



Simply looking at the plot, we can see that there is a positive correlation between variable 2 and variable 3. For linear models, multicolinearity can result in solutions that vary greatly and may be unstable numerically. In order to mitigate this issue, it is best to look at the R summary from the linear model: $y = \beta 0 + \beta_1 X_1 + \beta_2 X2 + \beta_3 X_3$ where the X's correspond to $var1$, $var2$, and $var3$ to determine which variable we can remove.

```
Call:
lm(formula = y ~ var1 + var2 + var3, data = project_data)

Residuals:
     Min       1Q   Median       3Q      Max
-11.0651  -4.3088   0.2126   4.2177  11.3239

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.2960     2.5340   4.063 8.93e-05 ***
var1          2.1369     0.5083   4.204 5.24e-05 ***
var2          1.3093     0.6903   1.897  0.06041 .
var3         -1.3668     0.4905  -2.787  0.00624 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.331 on 114 degrees of freedom
Multiple R-squared:  0.1698,    Adjusted R-squared:  0.148
F-statistic: 7.772 on 3 and 114 DF,  p-value: 9.095e-05
```

Figure 1: Summary of linear model

```
Call:
lm(formula = y ~ var1 + var3, data = project_data)

Residuals:
     Min       1Q   Median       3Q      Max
-10.2886  -4.2850   0.4284   4.3220  12.4803

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.3917     2.5620   4.056 9.12e-05 ***
var1          2.0970     0.5136   4.083 8.24e-05 ***
var3         -0.7069     0.3496  -2.022   0.0455 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.391 on 115 degrees of freedom
Multiple R-squared:  0.1436,    Adjusted R-squared:  0.1287
F-statistic: 9.642 on 2 and 115 DF,  p-value: 0.0001345
```

Figure 2: Exclude var2

By looking at the summary in R of the linear model with three variables, we find that variable 2 is not statistically significant under an alpha value of 0.05. This means we can safely exclude that variable from our model. Our updated linear model is as follows:

$$y = \beta_0 + \beta_1 X_1 + \beta_3 X_3 + \epsilon \tag{1}$$

Typically, in order for a linear model to satisfy least squares assumptions, we want the models residuals, ϵ, to be normal. However upon examination it appears that the residuals exhibit multimodality and therefore the residuals are not normal as seen in the Figure 3 that follows.

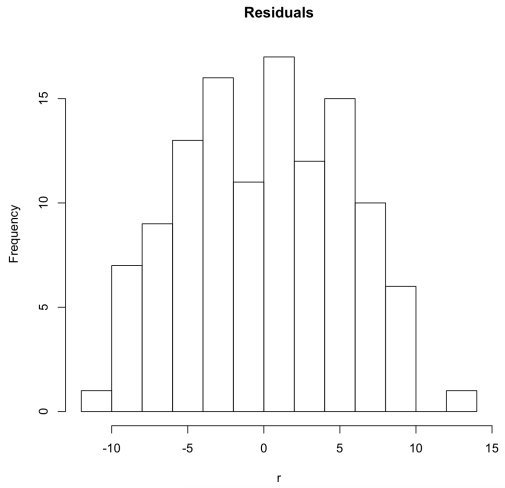


Figure 3: Multimodal Residuals

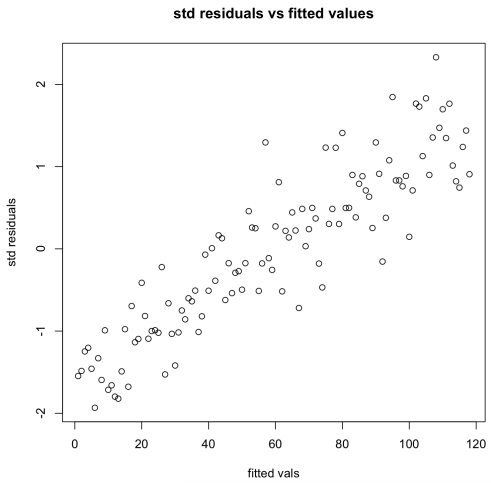


Figure 4: Heteroskedasticity

Non normal residuals means that the homoskedasticity assumption isn't satisfied. This can be verified by looking at the standardized residuals versus fitted values plot (Figure 4.

Since the residuals are not equally spread, but rather, exhibit a clear linear pattern, then it is clear that we have heteroskedasticity. Therefore we must do our modeling on the residuals.

3 Models

The model I chose to use the residuals was a mixture model as suggested in class. The residuals looked like they had approximately two modes so I chose to use a two component mixture.

3.1 Model 1

In my first model, I used two normals as my mixture components. Note however that this model does not take into account heteroskedasticity. Here each component has a distribution of: $y_i \sim N(x\beta, \sigma)$ in vector notation. This mixture assumes a linear regression with normal errors, which explicitly ignores the heteroskedasticity in the data.

The results can be seen in Figures 5 and 6. Here, the black line in Figure 5 represents the density of the residuals and the red line represents the mixture approximation density. In Figure 6, the red line is the density of the y values, and the black line is the y predictions.

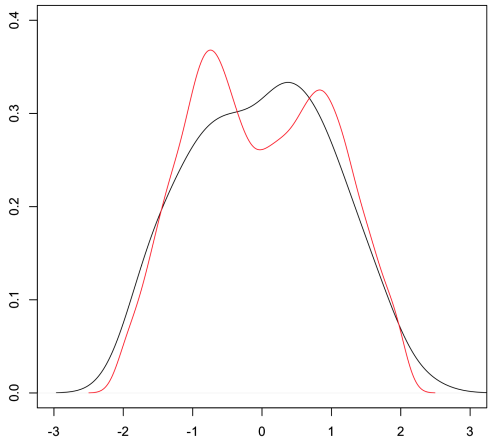


Figure 5: Residual Mixture Approximation

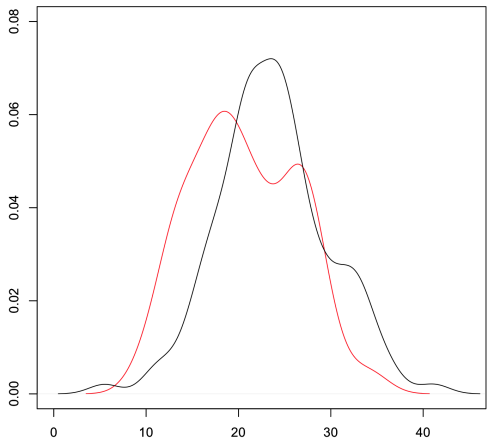


Figure 6: Y Prediction

In this model, I used cauchy distributions for each of the beta values. For model 1 and model 2, I used a β_0 of one component where the distribution was cauchy(0, 2) which is slightly narrower than the other mixtures which was cauchy(0.6, 5). The shifted mean values imply that there are at least two different intercepts in the residual model. The other β values all had cauchy distributions with parameters 1 and 0.5. This is a strong prior because the cauchy curve is sharp but has long tails.

The sigma values were given an improper flat prior, $[0, \infty]$. These sigmas were then use in our normal mixture components.

Once we had approximated the distribution of the residuals, we needed to find our predictions of y. In order to do so, we took the beta values from the linear regression and a sample from the mixture distribution for each data points ($var1$, $var2$, $var3$) and created a list of our predicted values. We then plotted the density of the given y values and the density of the predicted y values.

While we are able to capture the multimodality of the residuals, we find that our predictions for y are poor. This is because we did not introduce any variability in the variance of the residuals. In this model, the variance does not depend on any of the features.

3.2 Model 2

We account for heteroskedasticity by assigning a linear prior to the variance of the residuals. Our component distribution can be rewritten as follows: $y_i \sim N(x\beta, \lambda_i\omega)$. If $\lambda^2 \sim InvGamma(\gamma/2, \gamma/2)$ then instead of two normal mixtures, we can use: $y_i \sim StudentT(\gamma, X\beta, \sigma)$[1] as our distributions of our components. In this model, I assumed that both components had a studentized t-distribution which allowed me to account for the heteroskedasticity in the model under the assumption that the variance is distributed as an inverse gamma.

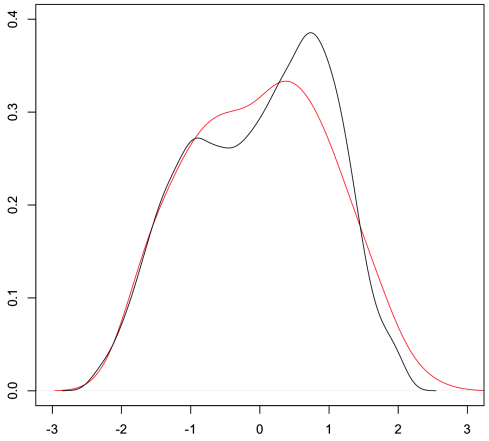


Figure 7: Residual Mixture Approximation

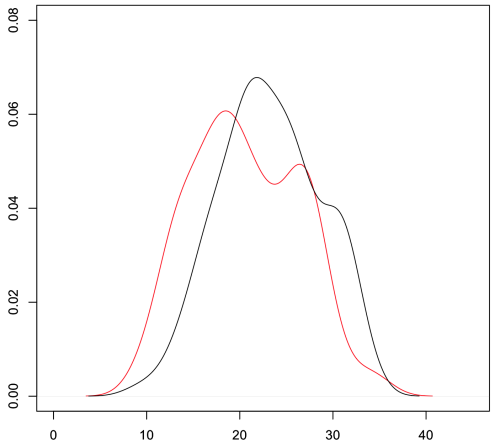


Figure 8: Y Prediction

Here, the red line in Figure 7 represents the density of the residuals and the black line represents the mixture approximation density. In Figure 8, the red line is the density of the y values, and the black line is the y predictions.

In this model, we keep the same priors for the β_0, the intercept, and the values for the hyper priors of the mean for each mixture as we had for Model 1. The other beta values stay the same as Model 1 as well; coming from a cauchy distribution with parameters 1 and 0.5. As mentioned above, this is a strong prior as it is steep with longer tails. However, the longer tails make the prior less informative than a normal.

We can see that we are able to approximate the residuals distribution with the mixture model a bit better when we account for the heteroskedasticity in the model. While we underestimate one of the modes slightly, over estimate the second and underestimate the tail on the right hand side, the shape of the mixture approximation is closer to that of the density of the residuals than Model 1.

We can see that our mixture approximation is modeling the residuals well by taking a sample from the mixture and seeing whether they are on our approximated posterior distribution of the residuals. Four examples of this can be seen in Figures 9 through 12. While it is difficult to see the multimodality of the residuals in this posterior distribution (the histogram), it is clear that the the true residual values (red line) appear to line up with our residual approximation.

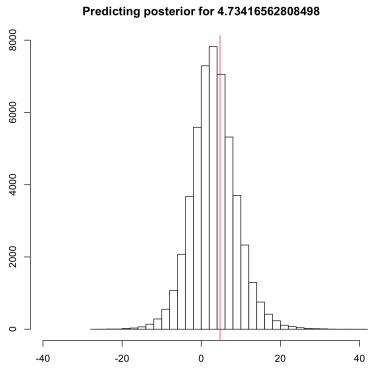


Figure 9: Sample residual 1

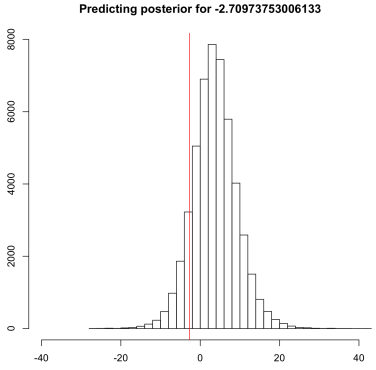


Figure 10: Sample residual 2

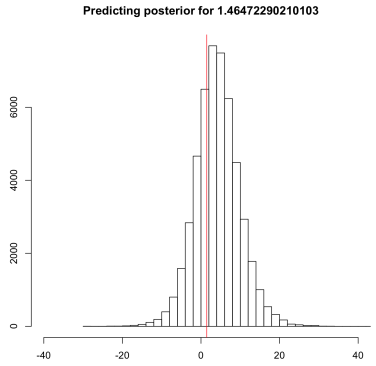


Figure 11: Sample residual 3

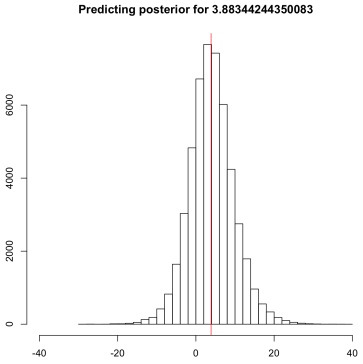


Figure 12: Sample residual 4

We can verify that our model is exploring the space by observing the label switching in Shiny Stan. If the labels appear to be jumping between two modes (given a two component mixture) then we know that our MCMC is properly exploring the space. [2]

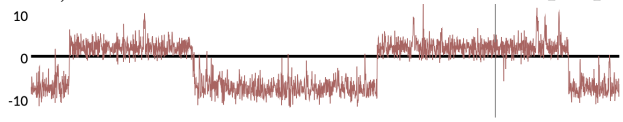


Figure 13: μ_1 chain 1

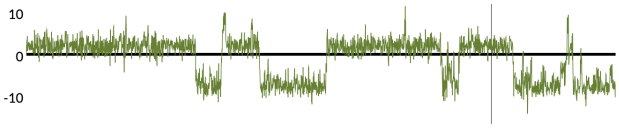


Figure 14: μ_1 chain 2

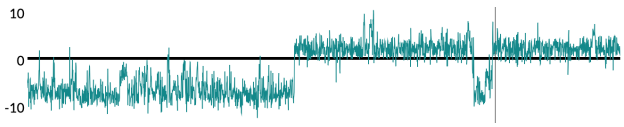


Figure 15: μ_1 chain 3

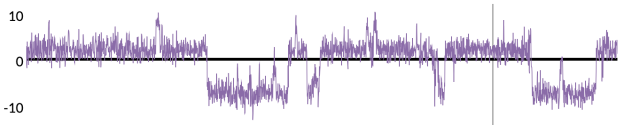


Figure 16: μ_1 chain 4

Because each chain appears to be finding the two modes, then we can say that we are successfully exploring the space. However, in chain 2, chain 3, and chain 4, we see that the chains appear to be lingering by one mode which typically implies a high autocorrelation and that the MCMC is having trouble exploring the space. Due to the fact that the modes of the residuals are so close together, it is reasonable that the mixture is having trouble jumping and separating the two modes. Even simply by observing the density of the residuals, we can see that the modes are nearly indistinct.

4 Comparison

There are multiple ways to analyze how well each model is doing. Three methods that will be used in this section are bayes factors (using the bridgesampling package in R), leave-one-out (LOO) cross-validation and Rhat values. Bayes factors are a form of bayesian hypothesis testing [3]. Bayes factors aim to quantify the support for one model over another and do not quantify how correct the models are. Leave-one-out cross validation is K-fold cross validation at its extreme. In this cross validation method, N separate times, the mixture is trained on all of the data points except for one point and a prediction is made for that point. [4] Rhat values are a basic way of assessing convergence of chains.

Starting with bayes factors, I used the R function bridgesampling. By using the bridge sampler function, I was able to find the log marginal likelihood of each stan fit object (each model). Then, I computed the bayes factor between model 1 and model 2 to find that model 1 is favored over model 2 with a bayes factor of 4.322. Given this result, we want to claim that model 1 is better than model 2.

Below we can see the results of loo and model 1 in Figure 17 and the results of loo and model 2 in Figure 18. We note that the goal of the pareto k k value is to assess the reliability of PSIS (Pareto Smoothed Importance Sampling) which is a method for stabilizing importance ratios. We note that the majority of the k values fall within the "good" range for both models. This means that "the variance of the raw importance ratios is finite, the central limit theorem holds, and the estimate converges quickly" [5]. Note, that while both models return very similar results, we see that there is one more value that falls into the good range for our second model. Here, we want to say that model 2 is better than model 1.

```
Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     101   85.6%   110
 (0.5, 0.7]   (ok)        17   14.4%   35
   (0.7, 1]   (bad)        0    0.0%   <NA>
   (1, Inf)   (very bad)   0    0.0%   <NA>
```

Figure 17: Pareto k diagnostic Model 1

```
Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     102   86.4%   60
 (0.5, 0.7]   (ok)        16   13.6%   41
   (0.7, 1]   (bad)        0    0.0%   <NA>
   (1, Inf)   (very bad)   0    0.0%   <NA>
```

Figure 18: Pareto k diagnostic Model 2

We can compare the two models using loo by first finding the loo objects of each stan model and then comparing them with loo compare.

```
> loo_compare(loo_fit1, loo_fit)
       elpd_diff se_diff
model2   0.0       0.0
model1 -13.9       1.2
```

A lower elpd diff value indicates the preferred model. Here, since model 2 appears to be lower, we can verify that loo thinks that model 2 is the best model.

Looking at the autocorrelation of lambda, the mixture coefficient, seen below

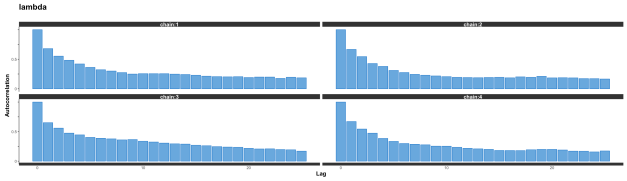


Figure 19: Autocorrelation Model 1

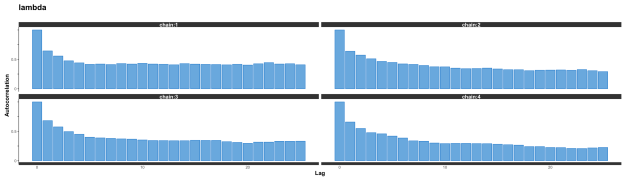


Figure 20: Autocorrelation Model 2

we can see that the autocorrelation is actually greater for more lag in model 2. This means

that our lambda values in our second model correlate with values more than one time period apart. The lambdas on our second model correlate with more previous time periods than in model 1. So, while we may have dealt with heteroskedasticity in model 2, we fail to account for autocorrelation.

Finally, we can look at the Rhat values:

	Rhat
lambda	1.011970
beta_val[1,1]	1.036227
beta_val[1,2]	1.018949
beta_val[2,1]	1.033151
beta_val[2,2]	1.017157
sigma[1]	1.011384
sigma[2]	1.009783
coef_int[1]	1.002112
coef_int[2]	1.003285
mu[1]	1.034981
mu[2]	1.033652

Figure 21: Model 1

	Rhat
lambda	1.045916
beta_val[1,1]	1.132409
beta_val[1,2]	1.067075
beta_val[2,1]	1.124016
beta_val[2,2]	1.067283
sigma[1]	1.040732
sigma[2]	1.026968
coef_int[1]	1.003367
coef_int[2]	1.002849
mu[1]	1.127151
mu[2]	1.119821

Figure 22: Model 2

We want an Rhat value that is close to 1. Here again, we find that the first model appears to have better Rhat values than model two. Given that some of the second models rhat values are greater than 1.1, a popular threshold, we can assume that some of these chains have not converged. Possible solutions for this are either running the chains for more iterations until they eventually converge or by spending more time tuning parameter.

5 Conclusion

Given these conflicting results, it is unclear which model fits our data best. While bayes factors prefer model 1 over model 2, LOO prefers model 2 and the rhat values seem to favor model 1. Since the normal distribution is well understood by beginning statistics students (the "bell curve"), the first model may be suitable. The first model is less complex than the second, though only slightly. However, since the second model takes the heteroskedasticity into account, the second model is likely more correct. The hyper parameters for the hyper priors and priors likely need to be tuned more, but overall, the non normal, multimodal residuals must be accounted for for a valid linear model. While the second model is theoretically better, it does not account for autocorrelation which will be accounted for in future work.

6 Discussion

While the second model does consider heteroskedasticity, there are various limitations that still exist in this model as is. For one, it is entirely possible that the residuals have more than one mode, in which case we would need to increase the number of components in our mixture approximation respectively. For two, it may be useful to look into dimensionality reduction methods such as principal component analysis to reduce the amount of random variables under consideration. Additionally, in order to work on the same scale, it may be useful to standardize the data so that each feature has an equal weight to start with.

Finally, as mentioned in the sections above, we fail to deal with autocorrelation with these two models. In future models, we will look into accounting for autocorrelation.

7 Collaborators

- Discussed the project with **Kathryn Gray** in order to figure out how to start this project
- Listened to **Colin Korbisch**'s talk about the project to finally get my stan code to run
- Talked to **Mark Tedder** about physically sampling from the mixture in order to get my y predictions

8 Appendix

Model 1 EDA:

```
project_data = read.table('Documents/Bayes/Project/ProjectData.csv',
    sep="\t", header=TRUE)
linear_reg <- lm(Y ~ var1+var3, data=project_data)
r = residuals(linear_reg)
pairs(project_data)
plot(rstandard(linear_reg), main="std residuals vs fitted values",
    xlab = "fitted vals", ylab = "std residuals")
library(car)
crPlots(linear_reg)
hist(r,10, main="Residuals")
```

Model 1 stan:

```
model<- '
data {
int<lower=0> N;
int<lower=1> K;
int<lower=1> J;
matrix[N,K] x;
vector[N] y;
}

parameters {
real<lower=0.1, upper=0.9> lambda;
vector[K-1] beta_val[J]; //slope coefficients
vector<lower=0>[J] sigma;
vector<lower=0, upper=40>[J] coef_int;
```

```
}

transformed parameters{
  vector[J] mu;
  vector[N] log_lik;
  for (n in 1:N){
    for (j in 1:J){
      mu[j] = x[n, 1:K-1]*beta_val[j]+x[n,K]*coef_int[j];
    }
  }
  for (n in 1:N) {
    vector[J] val;
    val[1] = log(lambda);
    val[2] = log(lambda);
    for(j in 1:J){
      val[j] += normal_lpdf(y[n] | mu[j], sigma[j]);
    }
    log_lik[n] = log_sum_exp(val);
  }
}

model{
  lambda ~ cauchy(5,5); // how likely is one mixture over another
  coef_int[1] ~  cauchy(0.3,2);
  coef_int[2] ~ cauchy(0.6,5);
  for(j in 1:J){
    beta_val[j,1:K-1] ~ cauchy(1,0.5); //(0,2.5)
  }
  {
    vector[J] x_beta_jj;
    real sample;
    for (n in 1:N){
      for (j in 1:J){
        x_beta_jj[j] = x[n, 1:K-1]*beta_val[j]+x[n,K]*coef_int[j];
      }

      target += log_mix(lambda,
      normal_lpdf(y[n] | x_beta_jj[1], sigma[1]),
      normal_lpdf(y[n] | x_beta_jj[2], sigma[2]));
    }
  }
}
'
```

Model 1 analyze stan results

```
x_temp = as.matrix(sapply(project_data[,c("var1", "var3")], as.numeric))
ones = as.vector(rep(1, 118))
x = cbind(x_temp, ones)
y = as.vector(project_data$Y)
r = as.vector(r)
dat = list(N=118, K=3, J=2, x=x, y=r)

fit1 <- stan(model_code = model, data = dat, iter = 5000, seed = 1234, chains = 4)
posterior1 <- as.data.frame(extract(fit, include = T))

theta = 0.7
b01 = 0.3
b02 = 1.1
b11 = 0.55
b12 = 0.9
b31 = 0.55
b32 = 0.66
sig1 = 4.5
sig2 = 5.9

entry_posterior <- function (entry, N){
  post = numeric(N)
  dat = as.numeric(entry)
  for ( i in 1:N){
    if (runif(1) < theta){ post[i] = rnorm(1, b01 + b11*dat[2] + b31*dat[4], sig1)  }
    else { post[i] = rnorm(1, b02 + b12*dat[2] + b32*dat[4], sig2) } }
  return(post)
}

n.sample = 50000
random.entry = sample(1:118, 1)
entry = project_data[random.entry,]
nomix.post1 = entry_posterior(entry, n.sample)
hist(nomix.post1, breaks=30, main=paste("Predicting posterior for",
    r[random.entry]), xlim = c(-40,40))
abline(v =  r[random.entry], col='red')

y_pred_list = c()
for (i in 1:118){
  print(i)
  entry = project_data[1,]
  nomix.post1 = entry_posterior(entry, n.sample)
  y_pred = linear_reg$coefficients[1] + linear_reg$coefficients[2]*project_data[i,2] +
    linear_reg$coefficients[3]*project_data[i,4] + sample (nomix.post1, size=1)
  y_pred_list = append(y_pred_list, y_pred)
```

```
}
plot(density(y), col='red', xlim=range(c(0,45)), ylim=range(c(0,0.08)))
par(new=TRUE)
plot(density(y_pred_list), xlim=range(c(0,45)), ylim=range(c(0,0.08)) )

plot(density((r-mean(r))/sqrt(var(r))), xlim=range(c(-3,3)), ylim=range(c(0,0.4)))
par(new=TRUE)
plot(density((posterior$lambda-mean(posterior$lambda))/sqrt(var(posterior$lambda))),
    col='red', xlim=range(c(-3,3)), ylim=range(c(0,0.4)))

library(shinystan)
launch_shinystan(fit)
```

Model 2 stan:

```
model<- '
  data {
    int<lower=0> N;
    int<lower=1> K;
    int<lower=1> J;
    matrix[N,K] x;
    vector[N] y;
  }

  parameters {
    real<lower=0.1, upper=0.9> lambda;
    vector[K-1] beta_val[J]; //slope coefficients
    vector<lower=0>[J] sigma;
    vector<lower=0, upper=40>[J] coef_int;
  }

  transformed parameters{
    vector[J] mu;
    vector[N] log_lik;
    for (n in 1:N){
      for (j in 1:J){
        mu[j] = x[n, 1:K-1]*beta_val[j]+x[n,K]*coef_int[j];
      }
    }
    for (n in 1:N) {
      vector[J] val;
      val[1] = log(lambda);
      val[2] = log(lambda);
      for(j in 1:J){
        val[j] += student_t_lpdf(y[n] | 10, mu[j], sigma[j]);
      }
```

```
      log_lik[n] = log_sum_exp(val);
    }
  }

  model{
    lambda ~ cauchy(5,5); // how likely is one mixture over another
    coef_int[1] ~  cauchy(0,2);
    coef_int[2] ~ cauchy(0.6,5);
    for(j in 1:J){
      beta_val[j,1:K-1] ~ cauchy(1,0.5); //(0,2.5)
    }
    {
      vector[J] x_beta_jj;
      real sample;
      for (n in 1:N){
        for (j in 1:J){
          x_beta_jj[j] = x[n, 1:K-1]*beta_val[j]+x[n,K]*coef_int[j];
        }
        target += log_mix(lambda,
            student_t_lpdf(y[n] | 10.1, x_beta_jj[1], sigma[1]),
            student_t_lpdf(y[n] | 10, x_beta_jj[2], sigma[2]));
      }
    }
  }
'
```

Model 2 analysis:

```
x_temp = as.matrix(sapply(project_data[,c("var1", "var3")], as.numeric))
ones = as.vector(rep(1, 118))
x = cbind(x_temp, ones)
y = as.vector(project_data$Y)
r = as.vector(r)
dat = list(N=118, K=3, J=2, x=x, y=r)

fit <- stan(model_code = model, data = dat, iter = 5000, seed = 1234, chains = 4)
posterior <- as.data.frame(extract(fit, include = T))

theta = 0.7
b01 = 0.3
b02 = 1.1
b11 = 0.55
b12 = 0.9
b31 = 0.55
b32 = 0.66
sig1 = 4.5
```

```
sig2 = 5.9

entry_posterior <- function (entry, N){
  post = numeric(N)
  dat = as.numeric(entry)
  for ( i in 1:N){
    if (runif(1) < theta){ post[i] = rst(1, mu=b01 + b11*dat[2] + b31*dat[4],
        sigma=sig1, nu=10.1)}  }
    else { post[i] = rst(1, mu=b02 + b12*dat[2] + b32*dat[4], sigma=sig2, nu=10) } }
  return(post)
}

n.sample = 50000
library("LaplacesDemon")
y_pred_list = c()
for (i in 1:118){
  print(i)
  entry = project_data[1,]
  nomix.post1 = entry_posterior(entry, n.sample)
  y_pred = linear_reg$coefficients[1] + linear_reg$coefficients[2]*project_data[i,2] +
    linear_reg$coefficients[3]*project_data[i,4] + sample (nomix.post1, size=1)
  y_pred_list = append(y_pred_list, y_pred)
}
plot(density(y), col='red', xlim=range(c(0,45)), ylim=range(c(0,0.08)))
par(new=TRUE)
plot(density(y_pred_list), xlim=range(c(0,45)), ylim=range(c(0,0.08)) )
plot(density((r-mean(r))/sqrt(var(r))), col='red' , xlim=range(c(-3,3)),
    ylim=range(c(0,0.4)))
par(new=TRUE)
plot(density((posterior$lambda-mean(posterior$lambda))/sqrt(var(posterior$lambda))),
    xlim=range(c(-3,3)), ylim=range(c(0,0.4)))
plot(density(y), col='black')
plot(density(r))
plot(density(posterior$lambda), col='red')

library(shinystan)
launch_shinystan(fit)

library(bridgesampling)
bs_fit1 = bridge_sampler(fit1)
bs_fit = bridge_sampler(fit)
bf(bs_fit1, bs_fit)

library(loo)
loo_fit1 = loo(fit1)
```

```
loo_fit = loo(fit)
loo_compare(loo_fit1, loo_fit)
```

References

[1] *Heteroskedasticity*. URL: https://jrnold.github.io/bayesian_notes/heteroskedasticity.html.

[2] *Label Switching*. URL: https://staffblogs.le.ac.uk/bayeswithstata/2014/05/18/label-switching/.

[3] *Bayes Factor*. URL: https://en.wikipedia.org/wiki/Bayes_factor.

[4] *Cross Validation*. URL: https://www.cs.cmu.edu/~schneide/tut5/node42.html.

[5] *pareto-k diagnostic*. URL: https://www.rdocumentation.org/packages/loo/versions/1.0.0/topics/pareto-k-diagnostic.