Final Project Report

Multilevel Bayesian Model for U.S. Labor Force Participation Rate

Course Code	STAT 5224 GR
Course Title	Bayesian Statistics
Instructor	Andrew Gelman

Member Name	**UNI**
Fan, Yang	fy2230
Xiaotong, Li	xl2788
Xiaotong, Lin	xl2506

Submission Date: December 9, 2018

Table of Content

1. Introduction

Labor force participation rate, defined as the inverse of the unemployment rate, is the percentage of working people in a group of people who are currently available for work. Rising labor force participation rate is seen as a sign of a strong economy, with fast growth and great spending. The determination of the health of the economy also helps to set monetary policy.

To gain more insights about U.S labor force participation rate, our group looks for the U.S. Current Population Survey from 1976 to 2015 providing individual-level data on geographic and demographic characteristics, including sex, race, age, skill, annual income, state, year, and labor market outcomes.

As we set each variable affecting the U.S. labor force participation rate into different groups, multinomial logistic regression, which is a classification method that generalizes logistic regression to multiclass problems, seems to be a good fit to our dataset with categorically distributed dependent variables to predict whether a person participates in a job or not.

This project will use R version 3.5.0 and Stan version 2.17.3. It also requires the following R packages:

```
library(rtan)
library(lme4)
library(rstanarm)
library(gridExtra)
library(ggplot2)
library(lattice)
```

1.1 Data Example

We will analyze the U.S. Current Population Survey from 5,219,430 respondents by grouping total 8 variables as follows:

Variable	Group #1	Group #2	Group #3	Group #4
sex	Male	Female	NA	NA
race	White	Non-white	NA	NA
age	20-29	30-39	40-49	50-59
skill	Skilled	Non-skilled	NA	NA
annual income	0-40k	40k-100k	100k-200k	>200k
state	North-west	South-west	North-east	South-east
year	1976-1985	1986-1995	1996-2005	2006-2015
labor force participation	Unemployed	Employed	NA	NA

Then the dataset consists of variables:

```
> summary(wage)
     sex             race             age            skilled      
 Min.   :1.000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000  
 1st Qu.:1.000   1st Qu.:1.0000   1st Qu.:1.000   1st Qu.:0.0000  
 Median :2.000   Median :1.0000   Median :1.000   Median :0.0000  
 Mean   :1.528   Mean   :0.8375   Mean   :1.374   Mean   :0.4243  
 3rd Qu.:41.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:1.0000  
 Max.   :2.000   Max.   :3.000    Max.   :1.0000  Max.   :170.000 

     income          state            year            lfp        
 Min.   :  0.000 Min.   : 1.00    Min.   :1976    Min.   :0.0000  
 1st Qu.:  0.000 1st Qu.:13.00    1st Qu.:1987    1st Qu.:0.0000  
 Median :  1.000 Median :29.00    Median :1998    Median :1.0000  
 Mean   :  1.738 Mean   :28.02    Mean   :1997    Mean   :0.6428  
 3rd Qu.:  2.000 3rd Qu.:2.000    3rd Qu.:2007    3rd Qu.:1.0000  
 Max.   :1.0000  Max.   :56.00    Max.   :2015    Max.   :1.0000  
```

By grouping people with specific variables, such as state and income, we can also calculate their labor force participation rate by the total number of people in the group dividing the number of people who work and generate distinct datasets as follows:

```
head(state)

##   year state lfpr
## 1 1977     1 0.55
## 2 1978     1 0.54
## 3 1979     1 0.54
## 4 1980     1 0.55
## 5 1981     1 0.56
## 6 1982     1 0.56

ggplot(state, aes(x=state,y=lfpr)) + geom_smooth() + labs(y='lfp rate')

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

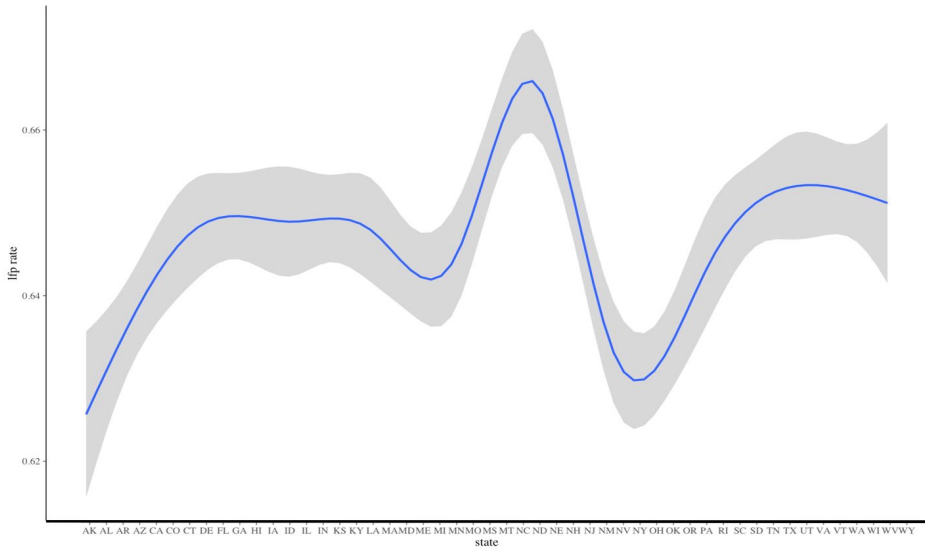

Fig.1.1.1: Labor Force Participation Rate of States

From above grouping by state, we can see that Minnesota has the greatest labor force participation rate. On the other hand, Alabama has the lowest labor force participation rate.

```
head(income)
##   year  income lfpr
## 1 1976      0 0.37
## 2 1977      0 0.39
## 3 1978      0 0.38
## 4 1979      0 0.38
## 5 1980      0 0.38
## 6 1981      0 0.37

ggplot(income, aes(x=income,y=lfpr)) + geom_smooth() + labs(y='lfp rate')
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

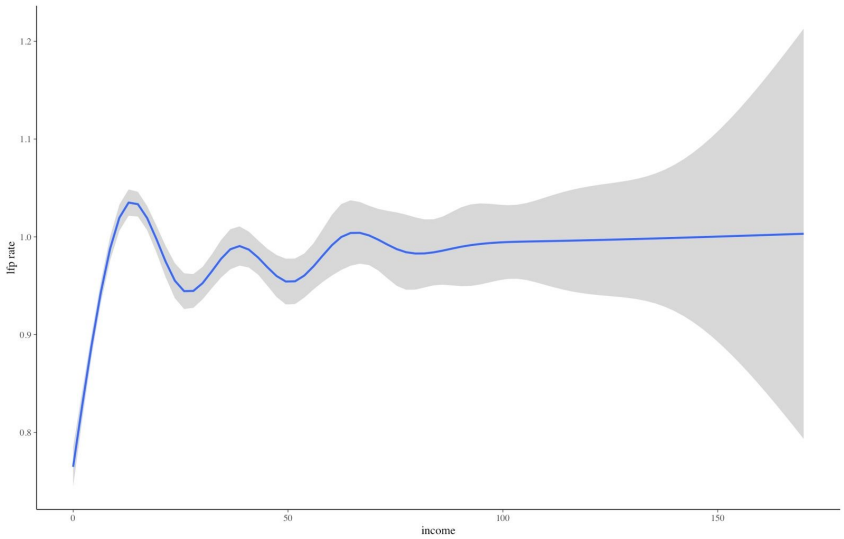

Fig.1.1.2: Labor Force Participation Rate of Income Group

From the above grouping by income, we can see that annual income has a positive relationship with the labor force participation rate before it reaches $100,000. Afterwards, the labor force participation rate keeps fluctuating horizontally with the increase of annual income. In addition, when people's annual income is larger than 1,500k, their labor force participation rate fluctuates a lot.

2. Multilevel Logistic Regression Model

2.1 Fitting Logit Regression On Multiple Predictors

2.1.1 Model Formulation

When attempting to fit models to the labor force participation, the logistic regression model seems to be a suitable choice considered that the labor participation is primarily a binary data filled with 0 and 1. In the discussion of the Regression model in Stan User's Guidebook, it provides a hint on fitting simple logistic models on one predictor. This model is formulated by a logit-parameterized version of the Bernoulli distribution which is defined as:

$$Bernoulli\ Logit\ (y|\alpha) \ = \ Bernoulli(y|logit^{-1}(\alpha))$$

From the Stan book, we learned that the error is built into the Stan, therefore we do not bother to account for that specifically. When attempting to fit the model to multiple predictors, we build our model in a format similar to the single predictor logit regression model:

$$P(Y_i = 1) \ = \ logit^{-1}(\alpha \ + \ \sum_{i=1}^{k} \beta_i \times x_i)$$

To explain this model, we consider k = 7 predictors which include sex, age, race, skill, annual income, state and year as we have discussed in the introduction section where we have explained the grouping criterion in details. This is by far the most comprehensive models that specifically examine each coefficient on the predictors to avoid identifiability issue when using the matrix as predictor inputs (vectorization). However, this model may cause scalability issue and sacrifice efficiency due to unreasonably long run-time in computation. So, the following stan model is not suggested if used in a large dataset. In our case, we also realize this problem and draw random samples of 15,000 from the dataset to make our model run faster. We separate the sample data into 5 fold with 3,000 in each fold. Then we use random 4 folds as training set and the last one as a test set to compute the prediction rate of the model. This will yield potentials for out-of-sample predictions when it comes to the model evaluation. We will come back to this after the results.

But on the good side, this model provides us a better insight on the intercept and the coefficients. By manipulating the model from the snippet of the simple logit regression, we add a transformed model block to see the normalized coefficients and the error scale for each predictor. We impose a weak prior to each of the parameters because no particular prior knowledge of these parameters is assumed.

It is worthwhile to mention that, in this model, we generate the prediction in the stan model rather than make predictions in R. Also, since we want the out-of-sample prediction, the model reads from the test data set for later prediction. This fit-and-predict method is slow yet fairly accurate, which is highly suggested in the Stan book, so we borrow this idea and incorporate this part in our model. However, for faster computation, we suggest predicting

outside of Stan, such as making a direct prediction by using the estimation results from Stan. Quite a trade-off between efficiency and accuracy!

```
################################################Stan Code#########################################################
data {
  int N;
  int Nt;
  int x1_N;                     //  number of categories in x1
  int x2_N;                     //  number of categories in x2
  int x3_N;                     //  number of categories in x3
  int x4_N;                     //  number of categories in x4
  int x5_N;                     //  number of categories in x5
  int x6_N;                     //  number of categories in x6
  int x7_N;                     //  number of categories in x7

  int<lower=1, upper=x1_N> x1[N];
  int<lower=1, upper=x2_N> x2[N];
  int<lower=1, upper=x3_N> x3[N];
  int<lower=1, upper=x4_N> x4[N];
  int<lower=1, upper=x5_N> x5[N];
  int<lower=1, upper=x6_N> x6[N];
  int<lower=1, upper=x7_N> x7[N];

  int<lower=1, upper=x1_N> x1t[Nt];
  int<lower=1, upper=x2_N> x2t[Nt];
  int<lower=1, upper=x3_N> x3t[Nt];
  int<lower=1, upper=x4_N> x4t[Nt];
  int<lower=1, upper=x5_N> x5t[Nt];
  int<lower=1, upper=x6_N> x6t[Nt];
  int<lower=1, upper=x7_N> x7t[Nt];
  int<lower=0, upper=1> y[N];
}
parameters {
  real intercept;
  vector[x1_N] x1_coeff;
  vector[x2_N] x2_coeff;
  vector[x3_N] x3_coeff;
  vector[x4_N] x4_coeff;
  vector[x5_N] x5_coeff;
  vector[x6_N] x6_coeff;
  vector[x7_N] x7_coeff;
  real<lower=0> sigma_x1;
  real<lower=0> sigma_x2;
  real<lower=0> sigma_x3;
  real<lower=0> sigma_x4;
  real<lower=0> sigma_x5;
  real<lower=0> sigma_x6;
  real<lower=0> sigma_x7;
}
transformed parameters {
  vector[x1_N] normalized_x1_coeff;
  vector[x2_N] normalized_x2_coeff;
  vector[x3_N] normalized_x3_coeff;
  vector[x4_N] normalized_x4_coeff;
  vector[x5_N] normalized_x5_coeff;
  vector[x6_N] normalized_x6_coeff;
  vector[x7_N] normalized_x7_coeff;
  normalized_x1_coeff = x1_coeff - x1_coeff[1];
  normalized_x2_coeff = x2_coeff - x2_coeff[1];
  normalized_x3_coeff = x3_coeff - x3_coeff[1];
  normalized_x4_coeff = x4_coeff - x4_coeff[1];
  normalized_x5_coeff = x5_coeff - x5_coeff[1];
```

```
  normalized_x6_coeff = x6_coeff - x6_coeff[1];
  normalized_x7_coeff = x7_coeff - x7_coeff[1];
}
model {
  vector[N] p;                  // probabilities
  // priors
  intercept ~ normal(0, 20);
  sigma_x1 ~ cauchy(0, 5);
  sigma_x2 ~ cauchy(0, 5);
  sigma_x3 ~ cauchy(0, 5);
  sigma_x4 ~ cauchy(0, 5);
  sigma_x5 ~ cauchy(0, 5);
  sigma_x6 ~ cauchy(0, 5);
  sigma_x7 ~ cauchy(0, 5);
  // level 1
  x1_coeff ~ normal(0, sigma_x1);
  x2_coeff ~ normal(0, sigma_x2);
  x3_coeff ~ normal(0, sigma_x3);
  x4_coeff ~ normal(0, sigma_x4);
  x5_coeff ~ normal(0, sigma_x5);
  x6_coeff ~ normal(0, sigma_x6);
  x7_coeff ~ normal(0, sigma_x7);
  // level 2
  for (i in 1:N) {
    p[i] <- x1_coeff[x1[i]] + x2_coeff[x2[i]] + x3_coeff[x3[i]] + x4_coeff[x4[i]]+  x5_coeff[x5[i]]+
x6_coeff[x6[i]]+ x7_coeff[x7[i]];
}
  y ~ bernoulli_logit(intercept + p);
}
generated quantities{
  vector[Nt] y_test;
  vector[Nt] pt;
  for (i in 1:Nt) {
    pt[i] <- x1_coeff[x1t[i]] + x2_coeff[x2t[i]] + x3_coeff[x3t[i]] + x4_coeff[x4t[i]]+  x5_coeff[x5t[i]]+
x6_coeff[x6t[i]]+  x7_coeff[x7t[i]];
    y_test[i] = bernoulli_rng(inv_logit(intercept + pt[i]));
  }}


################################################Implement Stan#################################################

rstan_options(auto_write=TRUE)
options(mc.cores = parallel::detectCores())
wage <- read.csv("lfp.csv")
wage<- as.data.frame(wage)

wage$income[which(wage$income<=4)] <- 1
wage$income[which(wage$income>4 & wage$income<=10)] <- 2
wage$income[which(wage$income>10 & wage$income<=20)] <- 3
wage$income[which(wage$income>20)] <- 4

wage$state[which(wage$state<=15)] <- 1
wage$state[which(wage$state>15 & wage$state<=30)] <- 2
wage$state[which(wage$state>30 & wage$state<=45)] <- 3
wage$state[which(wage$state>45 & wage$state<=60)] <- 4

wage$year[which(wage$year<=1985)] <- 1
wage$year[which(wage$year>1985 & wage$year<=1995)] <- 2
wage$year[which(wage$year>1995 & wage$year<=2005)] <- 3
wage$year[which(wage$year>2005 & wage$year<=2015)] <- 4

wage_new <- wage
ind =sample(1:dim(wage_new)[1], 15000)
wage_test = wage_new[ind[1:3000],]
wage_train1 = wage_new[ind[3001:6000],]
wage_train2 = wage_new[ind[6001:9000],]
```

```
wage_train3 = wage_new[ind[9001:12000],]
wage_train4 = wage_new[ind[12001:15000],]


# fit training data1
wage_train = wage_train1
x1 = wage_train$sex
x2 = wage_train$white +1
x3 = wage_train$age +1
x4 = wage_train$skilled +1
x5 = wage_train$income
x6 = wage_train$state
x7 = wage_train$year
y = wage_train$lfp

# prepare the test data
x1t = wage_test$sex
x2t = wage_test$white +1
x3t = wage_test$age +1
x4t = wage_test$skilled +1
x5t = wage_test$income
x6t = wage_test$state
x7t = wage_test$year
yt = wage_test$lfp
Nt = length(yt)


N = length(x1) ## // number of x
x1_N = length(unique(x1))## // number of categories in x
x2_N = length(unique(x2)) ##// number of categories in x2
x3_N = length(unique(x3))
x4_N = length(unique(x4))
x5_N = length(unique(x5))
x6_N = length(unique(x6))
x7_N = length(unique(x7))

data <- list(x1_N = x1_N, x2_N = x2_N,x3_N = x3_N,x4_N = x4_N,x5_N = x5_N,x6_N = x6_N,x7_N = x7_N,
             N = N,Nt = Nt, x1 = x1,x2 = x2,x3 = x3,x4 = x4,x5 = x5,x6 = x6,x7 = x7,x1t=x1t,
             x2t=x2t,x3t=x3t,x4t=x4t,x5t=x5t,x6t=x6t,x7t=x7t,
             y = y)

fit_multi1<-stan(file = "multi_logit4.stan",data=data, iter = 500, chains=1, control = list(max_treedepth
= 15))

fit_multi1_sum <- extract(fit_multi1)


# Now let's see how did we do about prediction on test
y_pred = apply(fit_multi1_sum$y_test, MARGIN = 2,FUN = mean)
y_pred_bin = ifelse(y_pred >=0.5,1,0)
y_pred_bin1 = y_pred_bin

# The success prediction rate
pre <- c(pre, mean(y_pred_bin==yt))
pre

## [1] 0.726

MSE <- c(MSE, var(y_pred_bin==yt)/length(y_pred_bin) )
MSE

## [1] 0.0001991231

fit_summary <- as.data.frame(summary(fit_multi1)$summary)

# Make vector of wanted parameter names
```

```
wanted_pars <- c(paste0("intercept"),paste0("x1_coeff[", 1:x1_N,"]"),
paste0("x2_coeff[",1:x2_N,"]"),paste0("x3_coeff[",1:x3_N,"]"),

paste0("x4_coeff[",1:x4_N,"]"),paste0("x5_coeff[",1:x5_N,"]"),paste0("x6_coeff[",1:x6_N,"]"),paste0("x7_co
eff[",1:x7_N,"]"),c("sigma_x1","sigma_x2","sigma_x3","sigma_x4","sigma_x5","sigma_x6","sigma_x7"))

# Get estimated and generating values for wanted parameters
estimated_values <- fit_summary[wanted_pars, c("mean", "2.5%", "97.5%")]

# Assemble a data frame to pass to ggplot()
sim_df <- data.frame(parameter = factor(wanted_pars, rev(wanted_pars)), row.names = NULL)
sim_df$middle <- estimated_values[, "mean"]
sim_df$lower <- estimated_values[, "2.5%"]
sim_df$upper <- estimated_values[, "97.5%"]

# Plot the posterior estimation of the parameters
ggplot(sim_df) + aes(x = parameter, y = middle, ymin = lower, ymax = upper) +
  scale_x_discrete() + geom_abline(intercept = 0, slope = 0, color = "white") +
  geom_linerange() + geom_point(size = 2) + labs(y = "Estimation", x = NULL) +
  theme(panel.grid = element_blank()) + coord_flip()
```

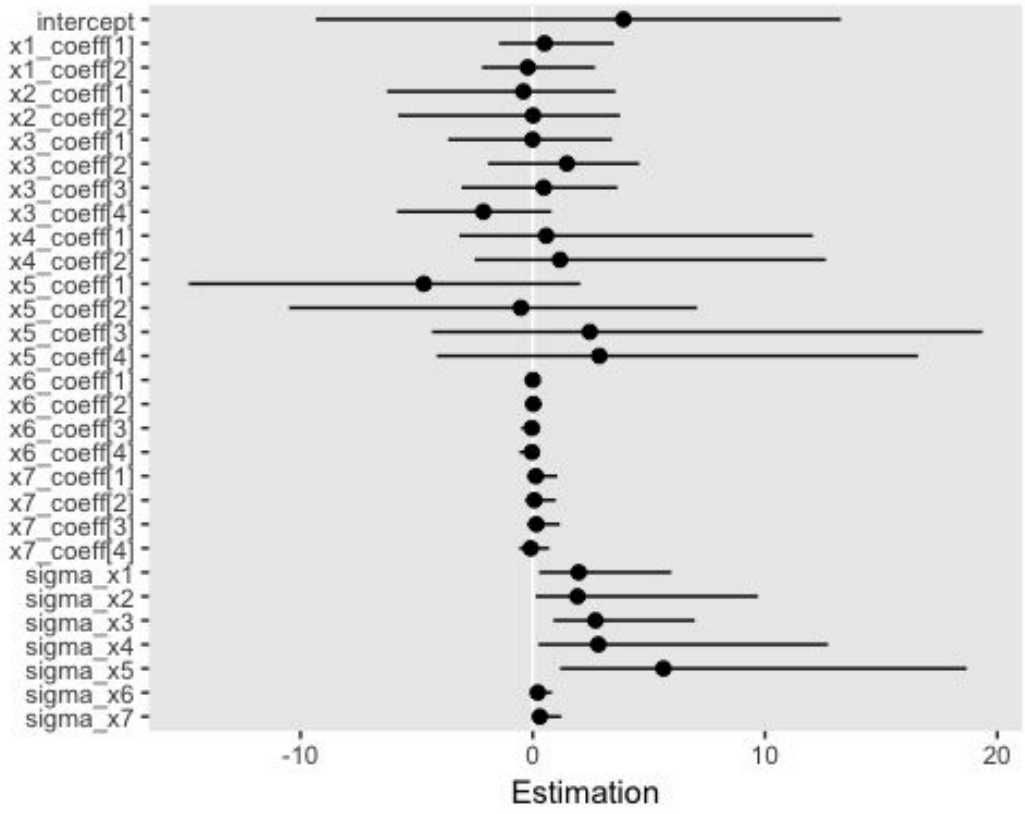

Fig 2.1.1: Posterior Estimation of the Parameters

2.1.2 Model Checking

We summarize our model by graphing out the posterior estimation of the parameters, with its mean and the 95% credible region. By a quick look at the Fig 2.1.1, we can see that this model performs poorly on two predictors: year and state. There are several potential reasons we need to figure out. For the year predictor, performing regression on the year is not potentially a

good idea as we realized later because a time series model is better suited to estimate the potential periodic behavior in this data. As for the state predictor, the criterion we have chosen to group these 51 states may be too vague to capture any useful insights from our model. In this project, the temporal analysis on the labor force participation rate is not the main objective. However, we can solve that later by looking at the state individually.

Further, we want to make sure the parameters are converged by looking at the Fig 2.1.2. We plot out the R-hat statistics for all of the posterior draws of the parameters and we could clearly observe that the R-hat statistics are all lower than 1.1. So this model yields convergent estimation of the parameters. However, when examining the trace plot of the parameters, we see that some of the coefficients for the age and income predictors perform not that good, which also can be observed in Fig 2.1.1. But the good news is that there is no clear indication that these coefficients are strongly biased. So one theory for the variation for these coefficients is caused by relatively small iteration of the draws, which will be solved if we have increased the iteration to significantly larger. But due to the limitation of computational power, we leave this matter as trivial.

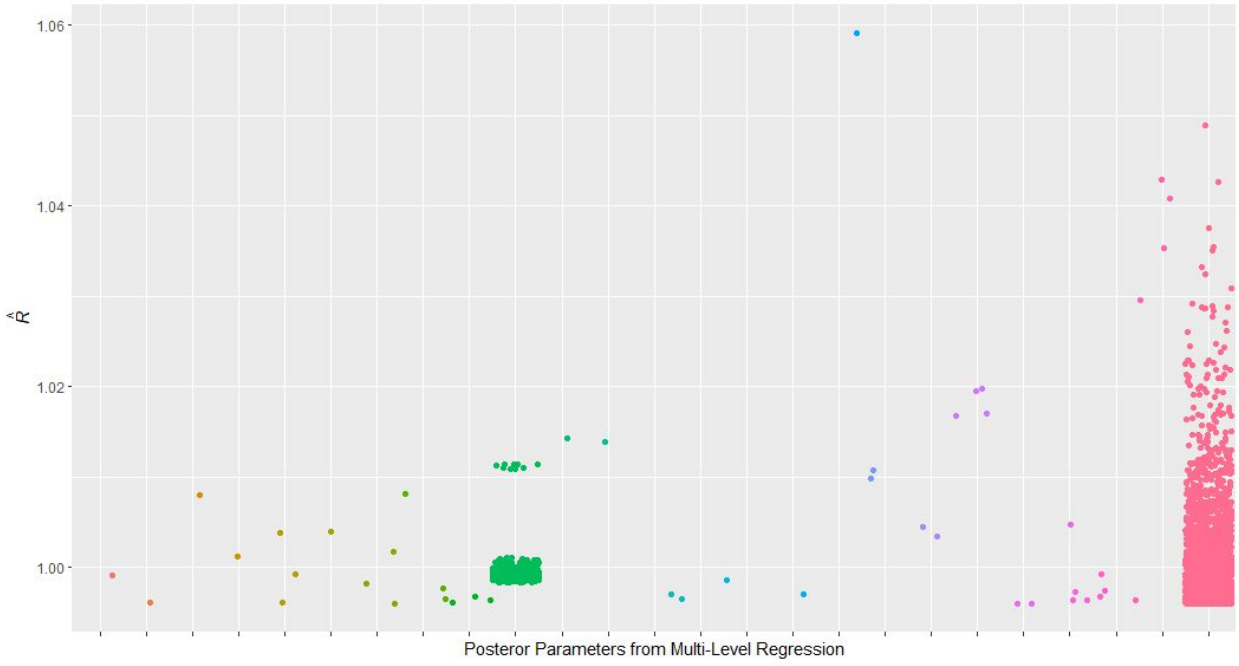

Fig 2.1.2: R-hat Statistics for All Parameters in the Model

```
library(ggplot2)
coef <-
cbind(fit_multi1_sum$x1_coeff[,1],fit_multi1_sum$x1_coeff[,2],fit_multi1_sum$x2_coeff[,1],fit_multi1_sum$x
2_coeff[,2],fit_multi1_sum$x3_coeff[,1],fit_multi1_sum$x3_coeff[,2],fit_multi1_sum$x3_coeff[,3],fit_multi1
_sum$x3_coeff[,4],fit_multi1_sum$x4_coeff[,1],fit_multi1_sum$x4_coeff[,2],fit_multi1_sum$x5_coeff[,1],fit_
multi1_sum$x5_coeff[,2],fit_multi1_sum$x5_coeff[,3],fit_multi1_sum$x5_coeff[,4],fit_multi1_sum$x6_coeff[,1
],fit_multi1_sum$x6_coeff[,2],fit_multi1_sum$x6_coeff[,3],fit_multi1_sum$x6_coeff[,4],fit_multi1_sum$x7_co
eff[,1],fit_multi1_sum$x7_coeff[,2],fit_multi1_sum$x7_coeff[,3],fit_multi1_sum$x7_coeff[,4])
```

```
index = seq(1:nrow(coef))
coef = as.data.frame(cbind(index,coef))

ggplot(coef, aes(index)) +
  geom_line(aes(y = coef[,2], colour = "sex1")) +
  geom_line(aes(y = coef[,3], colour = "sex2")) +
  geom_line(aes(y = coef[,4], colour = "white1")) +
  geom_line(aes(y = coef[,5], colour = "white2")) +
  geom_line(aes(y = coef[,6], colour = "age1")) +
  geom_line(aes(y = coef[,7], colour = "age2")) +
  geom_line(aes(y = coef[,8], colour = "age3")) +
  geom_line(aes(y = coef[,9], colour = "age4")) +
  geom_line(aes(y = coef[,10], colour = "skill1")) +
  geom_line(aes(y = coef[,11], colour = "skill2")) +
  geom_line(aes(y = coef[,12], colour = "income1")) +
  geom_line(aes(y = coef[,13], colour = "income2")) +
  geom_line(aes(y = coef[,14], colour = "income3")) +
  geom_line(aes(y = coef[,15], colour = "income4")) +
  geom_line(aes(y = coef[,16], colour = "state1")) +
  geom_line(aes(y = coef[,17], colour = "state2")) +
  geom_line(aes(y = coef[,18], colour = "state3")) +
  geom_line(aes(y = coef[,19], colour = "state4")) +
  geom_line(aes(y = coef[,20], colour = "year1")) +
  geom_line(aes(y = coef[,21], colour = "year2")) +
  geom_line(aes(y = coef[,22], colour = "year3")) +
  geom_line(aes(y = coef[,23], colour = "year4"))  + labs(y='Coefficient')
```

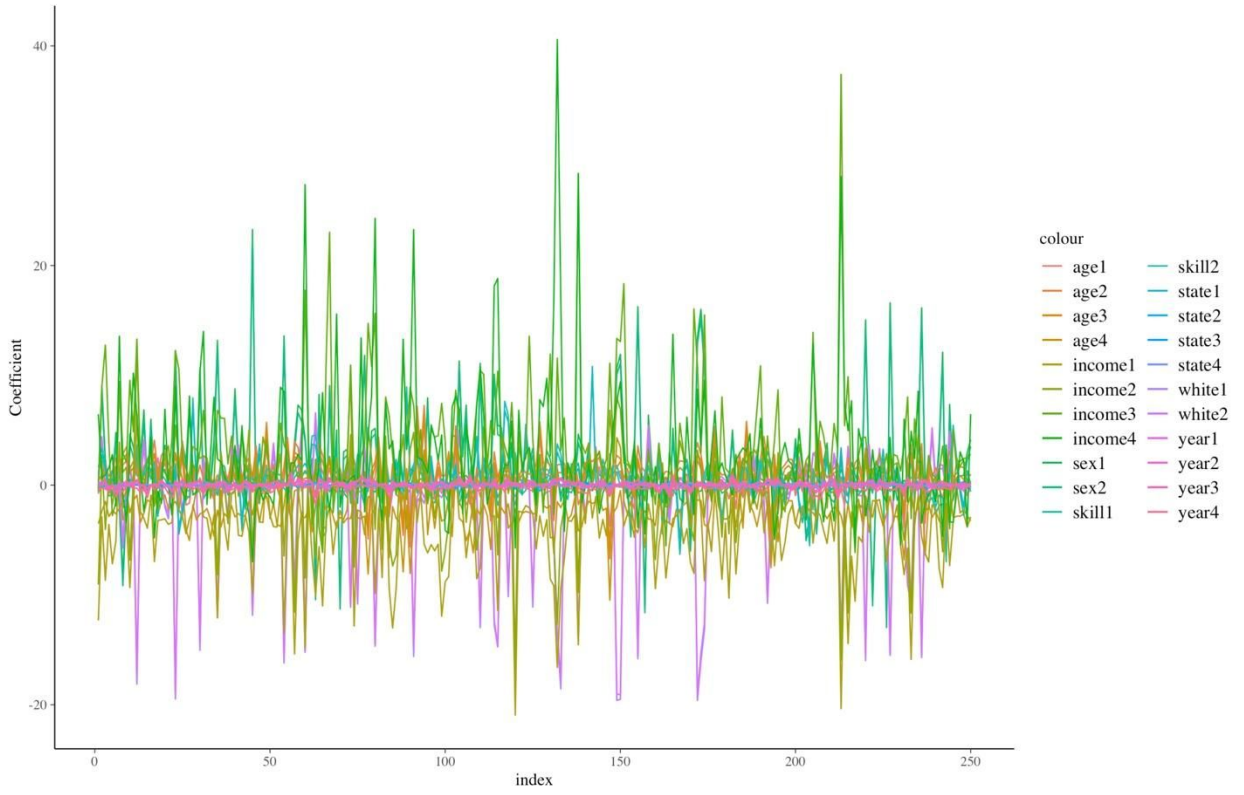

Fig 2.1.3: Trace-plots of Coefficients of Predictors

2.2 Model Fitting for Each State

2.2.1 Model Checking and Summary

Based on the inference from the previous model, we observe that the model does not capture the impact from the state to the labor force participation very well. Therefore, we decided to examine the remaining predictors while fixing the states. To do this, we simply subset the data set into 51 datasets by states. Since the dataset is enormous, these subsets are large and will cause runtime error as we have discussed previously. So, we apply the same trick to draw random samples from each state subset. The following analysis is based on New York state subset.

```
#Taking New York as an example:

state = 36 # New York
wage_new = wage[which(wage$state == state),]
ind =sample(1:dim(wage_new)[1], 15000)
wage_test = wage_new[ind[1:3000],]
wage_train1 = wage_new[ind[3001:6000],]
wage_train2 = wage_new[ind[6001:9000],]
wage_train3 = wage_new[ind[9001:12000],]
wage_train4 = wage_new[ind[12001:15000],]

# fit training data1
wage_train = wage_train1
x1 = wage_train$sex
x2 = wage_train$white +1
x3 = wage_train$age +1
x4 = wage_train$skilled +1
x5 = wage_train$income
y = wage_train$lfp

# prepare the test data
x1t = wage_test$sex
x2t = wage_test$white +1
x3t = wage_test$age +1
x4t = wage_test$skilled +1
x5t = wage_test$income
yt = wage_test$lfp
Nt = length(yt)

N = length(x1) ## // number of x
x1_N = length(unique(x1)) # number of categories in x
x2_N = length(unique(x2))
x3_N = length(unique(x3))
x4_N = length(unique(x4))
x5_N = length(unique(x5))

data <- list(x1_N = x1_N, x2_N = x2_N,x3_N = x3_N,x4_N = x4_N,x5_N = x5_N,x6_N = x6_N,
             N = N,Nt = Nt, x1 = x1,x2 = x2,x3 = x3,x4 = x4,x5 = x5,x6 = x6,x1t=x1t,
             x2t=x2t,x3t=x3t,x4t=x4t,x5t=x5t,x6t=x6t,
             y = y)

fit_multi1<-stan(file = "multi_logit3.stan",data=data, iter = 500, chains=1, control = list(max_treedepth
= 15))

fit_multi1_sum <- extract(fit_multi1)
```

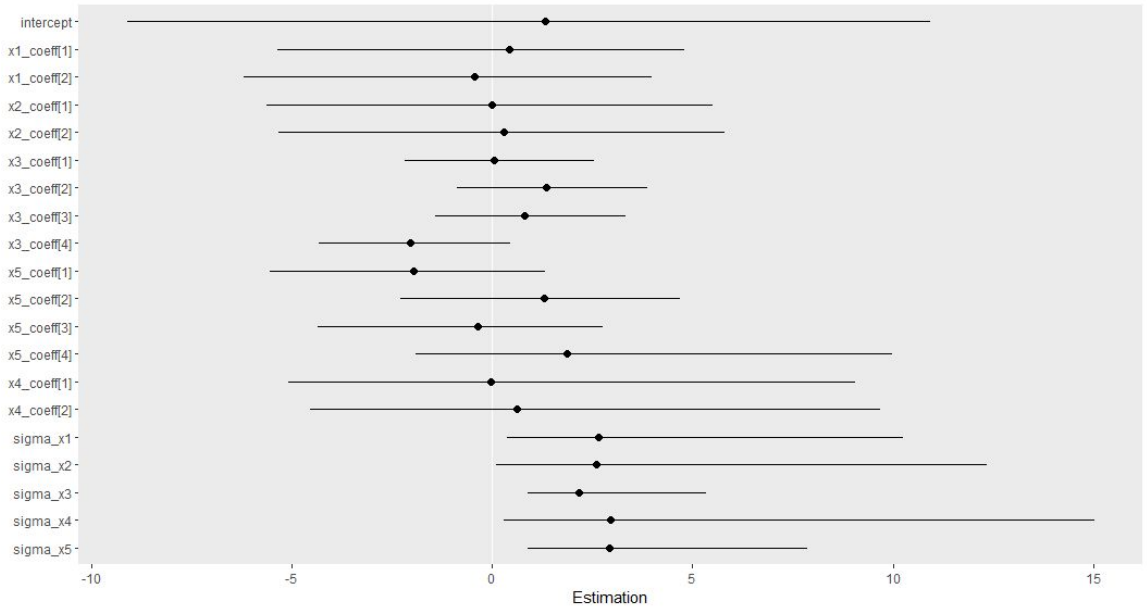

Fig 2.2.1: Posterior Estimation of the Parameters

To check the parameters easier, we examine the posterior estimation of the parameters by visualizing the simulation from the Stan model output. Fig 2.2.1 provides a comprehensive outlook that captures the coefficients of the predictors and the error scale. We also examine the convergence of the parameters by using R-hat statistics and conclude that all of the parameters do converge. We also have the trace-plot available for coefficients as a reference.

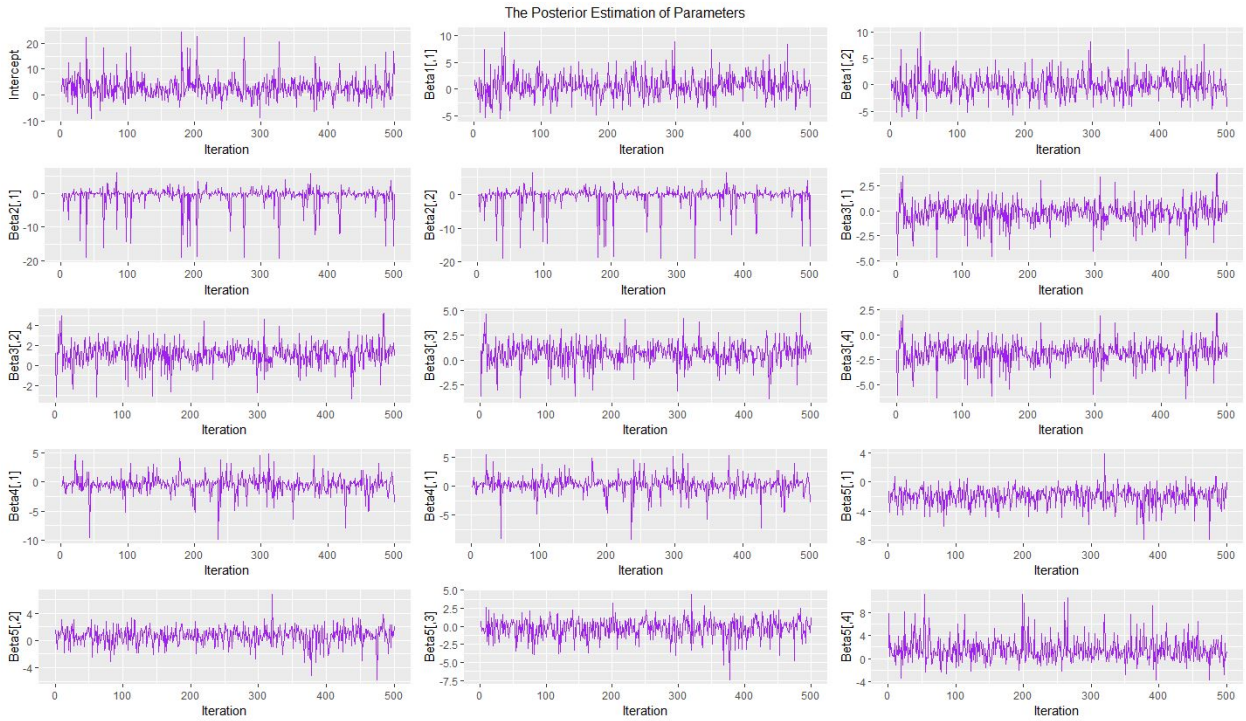

Fig 2.2.2: Trace-plot of Posterior Estimation for Coefficients

2.2.2 Model Evaluation

Fortunate for us, we have a luxurious large dataset which enables us to perform out-of-sample predictive accuracy using the available data. In section 2.2.1, we have mentioned the possible prediction using the posterior estimated parameters in Stan. In the data preparation, we have separated the 15,000 entries of data into 5 folds, where one fold with 3,000 entries is left as a test set. Therefore, including this toy model which only considered NY state, we will perform the model on each of the 4 training set and compare their results to the test set. The prediction rates are obtained using a simplified version of a confusion matrix where we calculate the rate of the number of successful predictions. The following code snippet shows the results from the first training set which is 0.739.

```
y_pred = apply(fit_multi1_sum$y_test, MARGIN = 2,FUN = mean)
y_pred_bin = ifelse(y_pred >=0.5,1,0)
y_pred_bin1 = y_pred_bin

# The success prediction rate
mean(y_pred_bin==yt)

## [1] 0.739
```

In order to avoid overfitting, we also replicate the process by fitting the model to the remaining training sets and then evaluating the predictive accuracy of the holdout test set. The results for the process are shown in Fig 2.2.3. We visualize the replicated results in the Fig 2.2.3. The performance of the model remains relatively stable across the training set. The overall prediction is adequately appropriate and the results run on the training set4 provides nearly perfect prediction on the test set, showing that the model generalizes the data decently well and we are on the right track in selecting the model.

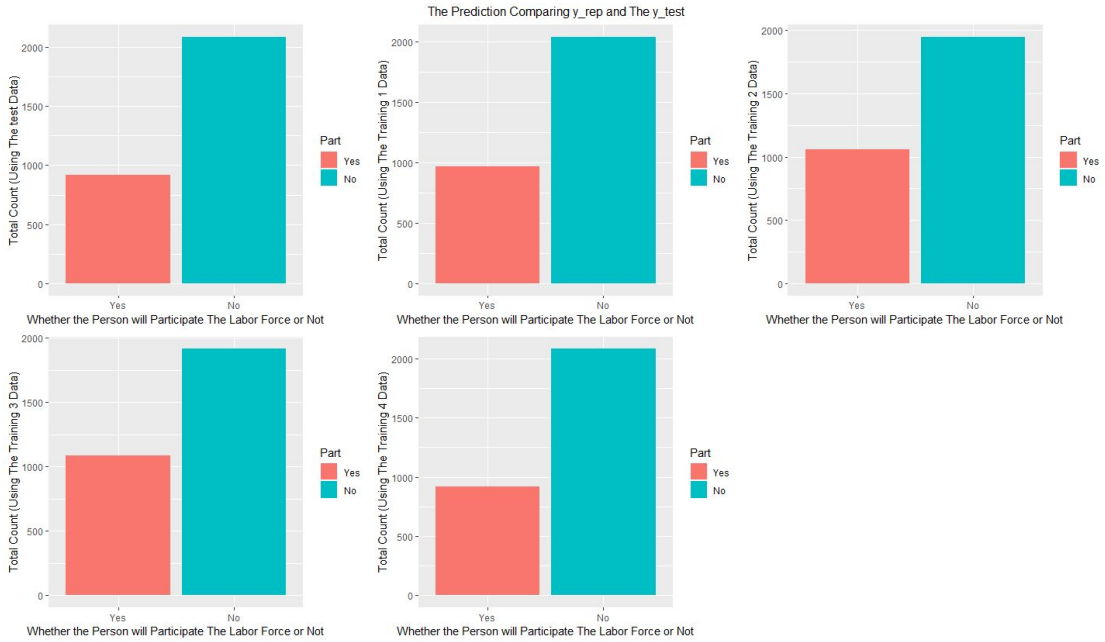

Fig 2.2.3: Labor Force Participation Prediction (NY state)

Therefore, we have confidence in expanding the model to the remaining 50 states and retrieve the prediction rate for each state. The summary of the results is shown in Figure 2.2.4. The prediction rate on the states are fairly stable, ranging from 0.7 to 0.8. However, the variation of the prediction rates may be better explained in a detailed study, which we will leave out this part to the potential model expansion.

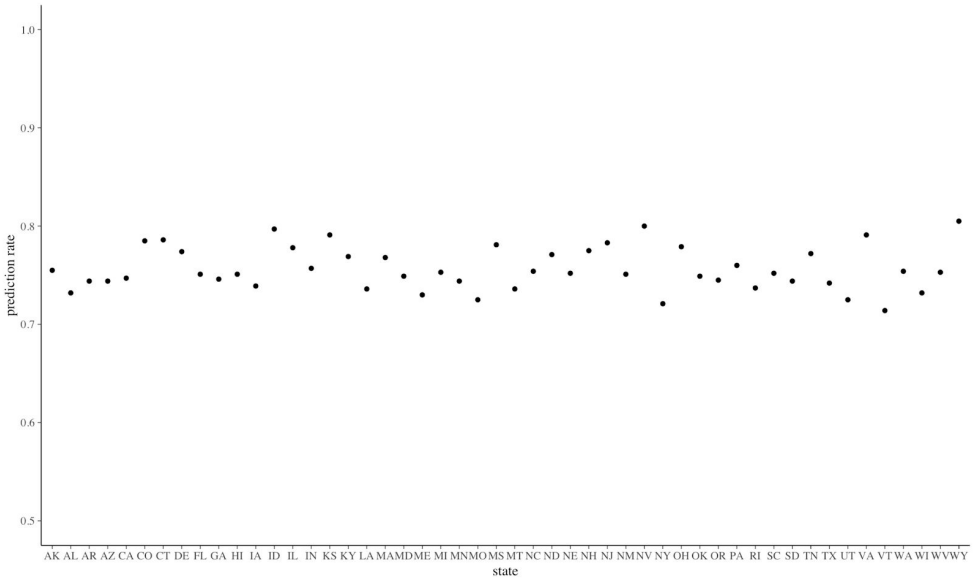

Fig 2.2.4: Prediction Rate for Each State (50 States)

2.3 A Vectorization Version of the Multi-Logit Regression Model

Since we have examine the application of multi-logit regression model to our data, we are very confident about the model and the next thing we consider is to speed up the computation by apply the vectorization form of the predictors. In the stan book, it provides a snippet of stan code to allows us to build the model with a valid start. In our case, we modify the Stan model as the following:

```
##############################Stan Code###################################
# Multilogit Stan
data {
  int<lower=2> K;             // number of categories
  int<lower=2> D;             // number of predictors
  int<lower=0> N;             // number of observations
  matrix[N, D] x;             // predictors
  int<lower=1,upper=K> y[N];  // observations
}
parameters {
matrix[K, D] beta;      // slopes
}
model {
  matrix[N, K] gamma;
  gamma = x * beta';

  // prior
  to_vector(beta) ~ normal(0, 5);
```

```
  // likelihood
  for (n in 1:N)
    y[n] ~ categorical_logit(gamma[n]');
}
```

The above stan model will provide similar results to the previous data while speeding up the computation. One caveat of using this stan codes is that we generally should incorporate a column of ones when using the vectorization form of the predictors, otherwise, we would not be able to examine the intercepts of the regression. The following output from stan provides sample outcomes from this snippet:

```
Inference for Stan model: multi_logit.
1 chains, each with iter=200; warmup=100; thin=1;
post-warmup draws per chain=100, total post-warmup draws=100.

              mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff Rhat
beta[1,1]    -0.51    0.43 3.50    -7.18    -2.94    -0.63     2.10     5.44    68 0.99
beta[1,2]    -0.40    0.39 3.57    -6.93    -2.91    -0.35     2.25     6.50    84 1.00
beta[1,3]    -0.04    0.45 3.79    -7.85    -2.61     0.24     2.37     7.22    72 0.99
beta[1,4]    -0.25    0.47 4.14    -7.43    -3.50    -0.45     3.00     7.07    77 0.99
beta[1,5]    -1.08    0.40 3.48    -7.71    -3.51    -0.90     0.83     6.12    76 1.02
beta[1,6]    -0.91    0.42 3.23    -7.20    -3.46    -0.60     1.65     4.16    58 1.01
beta[2,1]     0.23    0.43 3.51    -6.36    -2.27     0.10     2.79     6.18    67 0.99
beta[2,2]    -0.71    0.39 3.57    -7.26    -3.19    -0.68     1.95     6.17    83 1.00
beta[2,3]     0.03    0.45 3.79    -7.80    -2.56     0.32     2.48     7.25    72 0.99
beta[2,4]    -0.67    0.47 4.14    -7.86    -3.94    -0.87     2.60     6.66    77 0.99
beta[2,5]    -0.78    0.40 3.48    -7.41    -3.20    -0.59     1.16     6.46    76 1.02
beta[2,6]     0.60    0.43 3.25    -5.64    -1.76     0.87     3.07     5.86    59 1.00
lp__      -1756.15    0.26 2.21 -1760.14 -1757.84 -1756.12 -1754.57 -1752.15    71 1.03

# Samples were drawn using NUTS(diag_e) at Sun Dec 09 16:18:05 2018.
# 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).
```

3. Potential Model Expansion

3.1 Varying Intercept Model with No Predictors

To analyze the relationship between each variable and labor force participation more specifically, we group people with specific variable and then compute their labor force participation rate by the total number of people in the group dividing the number of people who work.

Considering the simplest multilevel model for the labor force participation rate of the group to which every respondent belongs nested within other 7 variables, we can write a two-level varying intercept model with no predictors using the usual two-stage formulation as

$$y_{ij} = \alpha_j + \varepsilon_{ij}, where\ \varepsilon_{ij} \sim N(0, \sigma_y^2)$$

$$\alpha_j = \mu_\alpha + u_j, where\ u_j \sim N(0, \sigma_\alpha^2)$$

where we will insert y_{ij} as the labor force participation rate and α_j as different variables. After attempting all 7 variables, we find that income is the one with the smallest Akaike information criterion(AIC). Below, we can see the comparison of this model based on state and income:

3.1.1 Labor Force Participation Rate based on State

```
M1_stanlmer <- stan_lmer(formula = lfpr ~ 1 + (1 | state),
                         data = state,
                         seed = 349)

## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 0.000313 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 3.13 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 23.1431 seconds (Warm-up)
##                5.40667 seconds (Sampling)
##                28.5497 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 0.000159 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.59 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 24.1687 seconds (Warm-up)
##                7.48834 seconds (Sampling)
##                31.657 seconds (Total)
##
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
##
## Gradient evaluation took 0.000159 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.59 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 19.2271 seconds (Warm-up)
##                5.50527 seconds (Sampling)
##                24.7324 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
##
## Gradient evaluation took 0.000167 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.67 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 26.9976 seconds (Warm-up)
##                5.83871 seconds (Sampling)
##                32.8363 seconds (Total)
```

After fitting a model using stan_lmer, we can check the priors by prior_summary function.

```
# Obtain a summary of priors used
prior_summary(object = M1_stanlmer)

## Priors for model 'M1_stanlmer'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 0.46
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
```

```
##      **adjusted scale = 0.046 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details

# Obtain SD of outcome
sd(state$lfpr, na.rm = TRUE)

## [1] 0.04555567
```

As seen above, the scales of the priors for μ_α and σ_y are set to 0.46 and 0.046 respectively after rescaling.

Now we can display a quick summary of the fit from Model 1 using the print method.

```
#Posterior medians and posterior median absolute deviations
print(M1_stanlmer, digits = 2)

## stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ 1 + (1 | state)
##  observations: 2002
## ------
##             Median MAD_SD
## (Intercept) 0.65   0.00
## sigma       0.03   0.00
##
## Error terms:
##  Groups   Name        Std.Dev.
##  state    (Intercept) 0.0379
##  Residual             0.0269
## Num. levels: state 51
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.65   0.00
##
## ------
## For info on the priors used see help('prior_summary.stanreg').

#Posterior means, posterior standard deviations, 95% credible intervals and Monte Carlo errors
summary(M1_stanlmer,
        pars = c("(Intercept)", "sigma", "Sigma[state:(Intercept),(Intercept)]"),
        probs = c(0.025, 0.975),
        digits = 2)

##
## Model Info:
##
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ 1 + (1 | state)
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 2002
##  groups:       state (51)
##
## Estimates:
##                                        mean   sd   2.5%   97.5%
## (Intercept)                           0.65   0.00 0.64   0.66
## sigma                                 0.03   0.00 0.03   0.03
## Sigma[state:(Intercept),(Intercept)] 0.00   0.00 0.00   0.00
```

```
## Diagnostics:
##                                     mcse Rhat n_eff
## (Intercept)                         0.00 1.05  101
## sigma                               0.00 1.00 1944
## Sigma[state:(Intercept),(Intercept)] 0.00 1.02  201
##
## For each parameter, mcse is Monte Carlo standard error, 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).
```

```
# Extract the posterior draws for all parameters
sims <- as.matrix(M1_stanlmer)
dim(sims)
```

```
## [1] 4000   54
```

```
para_name <- colnames(sims)
# para_name

##Obtaining means, standard deviations, medians and 95% credible intervals.

# Obtain state-level varying intercept a_j
# draws for overall mean
mu_a_sims <- as.matrix(M1_stanlmer,
                       pars = "(Intercept)")
# draws for statel-level error
u_sims <- as.matrix(M1_stanlmer,
                    regex_pars = "b\\[\\(Intercept\\) state\\:")
# draws for states' varying intercepts
a_sims <- as.numeric(mu_a_sims) + u_sims
# Obtain sigma_y and sigma_alpha^2
# draws for sigma_y
s_y_sims <- as.matrix(M1_stanlmer,
                      pars = "sigma")
# draws for sigma_alpha^2
s__alpha_sims <- as.matrix(M1_stanlmer,
                           pars = "Sigma[state:(Intercept),(Intercept)]")


# Compute mean, SD, median, and 95% credible interval of varying intercepts
# Posterior mean and SD of each alpha
a_mean <- apply(X = a_sims,
                MARGIN = 2,
                FUN = mean)
a_sd <- apply(X = a_sims,
              MARGIN = 2,
              FUN = sd)
# posterior mean
# posterior SD
# Posterior median and 95% credible interval
a_quant <- apply(X = a_sims,
                 MARGIN = 2,
                 FUN = quantile,
                 probs = c(0.025, 0.50, 0.975))
a_quant <- data.frame(t(a_quant))
names(a_quant) <- c("Q2.5", "Q50", "Q97.5")
# Combine summary statistics of posterior simulation draws
a_df <- data.frame(a_mean, a_sd, a_quant)
round(head(a_df), 2)
```

```
##                        a_mean a_sd Q2.5  Q50 Q97.5
## b[(Intercept) state:1]   0.59    0 0.58 0.59  0.60
## b[(Intercept) state:2]   0.70    0 0.69 0.70  0.70
## b[(Intercept) state:4]   0.62    0 0.61 0.62  0.63
## b[(Intercept) state:5]   0.60    0 0.60 0.60  0.61
```

```
## b[(Intercept) state:6]   0.64    0 0.63 0.64  0.65
## b[(Intercept) state:8]   0.69    0 0.68 0.69  0.70

# Sort dataframe containing an estimated alpha's mean and sd for every school
a_df <- a_df[order(a_df$a_mean), ]
a_df$a_rank <- c(1 : dim(a_df)[1])  # a vector of school rank
# Plot state-level alphas's posterior mean and 95% credible interval
ggplot(data = a_df,
       aes(x = a_rank,
           y = a_mean)) +
  geom_pointrange(aes(ymin = Q2.5,
                      ymax = Q97.5),
                  position = position_jitter(width = 0.1,
                                             height = 0)) +
  geom_hline(yintercept = mean(a_df$a_mean),
             size = 0.5,
             col = "red") +
  scale_x_continuous("Rank",
                     breaks = seq(from = 0,
                                  to = 80,
                                  by = 5)) +
  scale_y_continuous(expression(paste("varying intercept, ", alpha[j]))) +
  theme_bw( base_family = "serif")
```

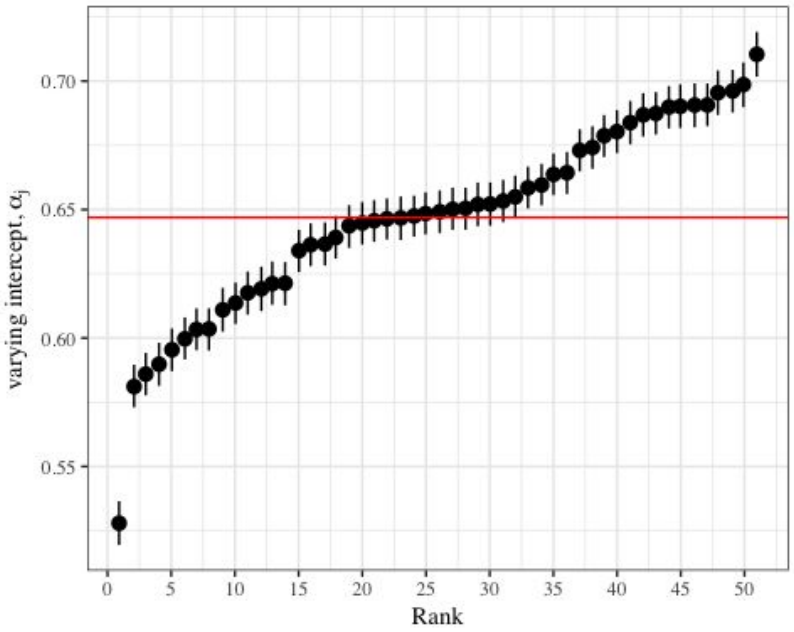

Fig 3.1.1.1: State-level alphas's Posterior Mean and 95% Credible Interval

```
##Making comparisons between individual states
# The difference between the two state averages (state #6 California and #36 New York)
state_diff <- a_sims[, 6] - a_sims[, 36]

# Investigate differences of two distributions
mean <- mean(state_diff)
sd <- sd(state_diff)
quantile <- quantile(state_diff, probs = c(0.025, 0.50, 0.975))
quantile <- data.frame(t(quantile))
names(quantile) <- c("Q2.5", "Q50", "Q97.5")
diff_df <- data.frame(mean, sd, quantile)
round(diff_df, 6)
```

```
##       mean       sd     Q2.5      Q50    Q97.5
## 1 0.053897 0.006047 0.041949 0.053896 0.065881

state_diff <- state_diff*100
# Histogram of the differences
ggplot(data = data.frame(state_diff),
       aes(x = state_diff)) +
  geom_histogram(color = "black",
                 fill = "gray",
                 binwidth = 0.25) +
  scale_x_continuous("Labor Force Participation Rate diffence between two states: California & New York
(*100)",
                     breaks = seq(from = 3,
                                  to = 8,
                                  by = 0.5)) +
  geom_vline(xintercept = c(mean(state_diff),
                            quantile(state_diff,
                                     probs = c(0.025, 0.975))),
             colour = "red",
             linetype = "longdash") +
  geom_text(aes(5.39, 30, label = "mean = 0.0539"),
            color = "red",
            size = 4) +
  geom_text(aes(7, 50, label = "SD = 0.0060"),
            color = "blue",
            size = 4) +
  theme_bw( base_family = "serif")
```

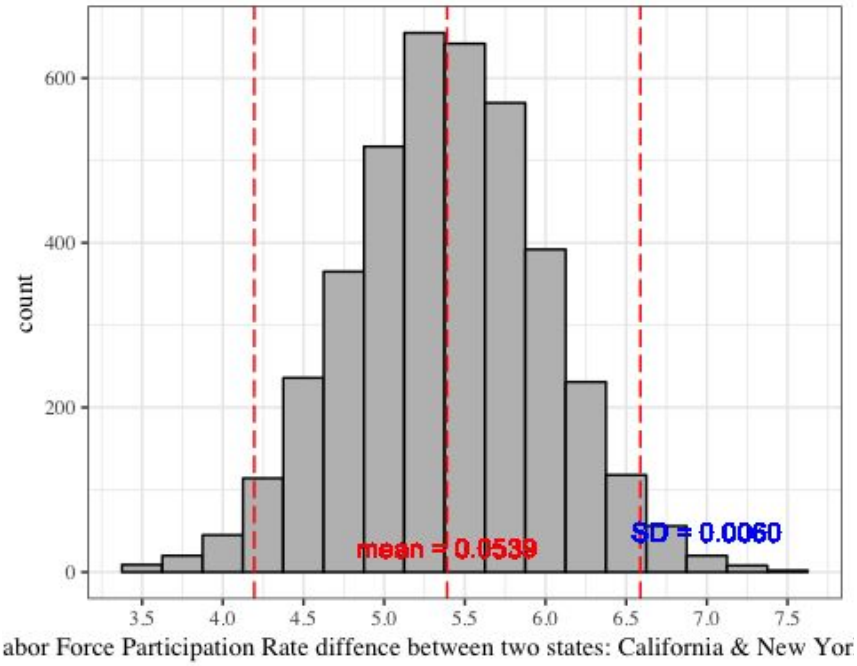

Fig 3.1.1.2: Histogram of the differences

```
plot(M1_stanlmer, "rhat")
```

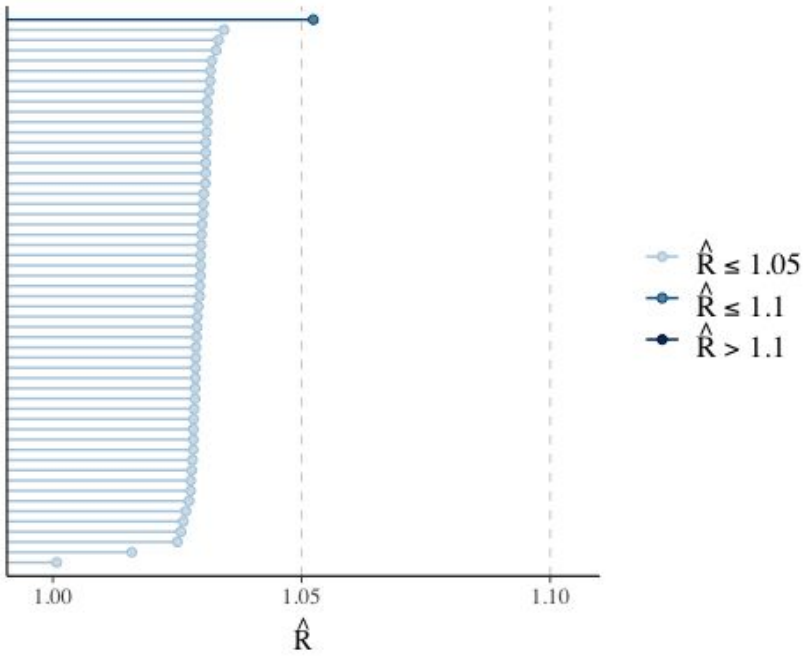

Fig 3.1.1.3

```
plot(M1_stanlmer, "ess")
```

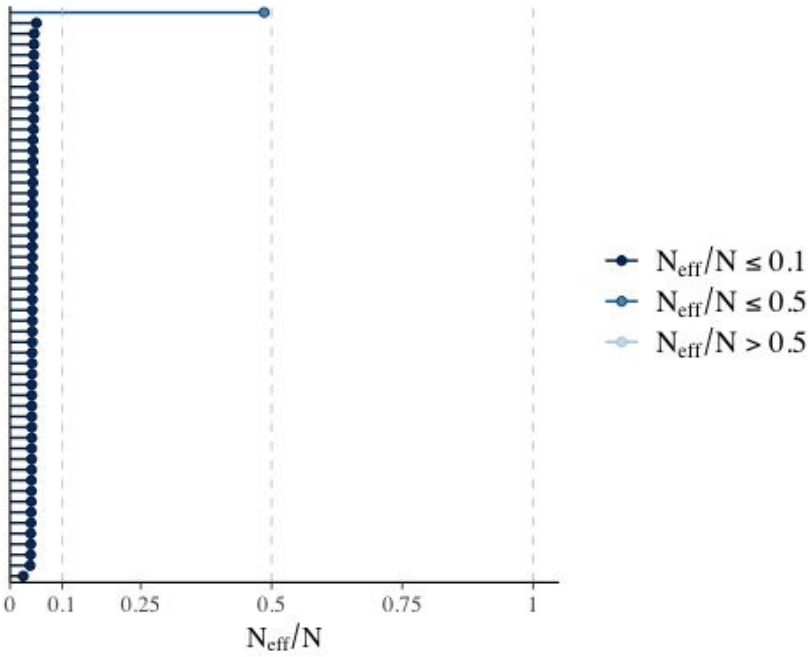

Fig 3.1.1.4

```
y_pre <- rep(NA, nrow(state))
for (i in 1: nrow(state)){
  a <- rnorm(1,0.65,0.03)
  y_pre[i] <- rnorm(1,a,0)
}

library(Metrics)

MSE <- round(mse(y_pre,state$lfpr),4)
MSE

## [1] 0.0029
```

From Fig. 3.1.1, we can see that the caterpillar plot shows the fully Bayes estimates for the state-varying intercepts in rank order together with their 95% credible intervals. By inserting the fitted parameter back to the model, the model gets 0.0029 MSE, which is quite small.

Furthermore, as California (#6) and New York(#36) are two states with great GDP in the U.S., we are interested in the comparison between them. We take the difference between the two vectors of draws $\alpha_6 - \alpha_{36}$ and investigate the posterior distribution of the difference with descriptive statistics and a histogram.

The expected difference comes to 0.0539 with a standard deviation of 0.006 and a wide range of uncertainty (Fig. 3.1.1.2). The 95% credible interval is [0.0419, 0.0659], so we are 95% certain that the true value of the difference between the two states lies within the range given the data. We then evaluate model convergence. The diagnostics which we use to access whether the chains have converged to the posterior distribution are the statistics $\hat{R}$ and N_{eff} . The $\hat{R}$ is the ratio of between-chain variance to within-chain variance. Since the $\hat{R}$ is always less than 1.1, all the chains have converged.

3.1.2 Labor Force Participation Rate based on Income

```
M1_stanlmer <- stan_lmer(formula = lfpr ~ 1 + (1 | income),
                         data = income,
                         seed = 349)
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 0.000111 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.11 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 16.3059 seconds (Warm-up)
##                3.40846 seconds (Sampling)
##                19.7144 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 0.000107 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.07 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 16.0588 seconds (Warm-up)
##                3.40529 seconds (Sampling)
##                19.4641 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
##
## Gradient evaluation took 0.000105 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.05 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 12.991 seconds (Warm-up)
##                3.44292 seconds (Sampling)
##                16.4339 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
##
## Gradient evaluation took 0.000107 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.07 seconds.
## Adjust your expectations accordingly!
##
##
```

```
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 15.9128 seconds (Warm-up)
##                3.39316 seconds (Sampling)
##                19.306 seconds (Total)
```

```
# Obtain a summary of priors used
prior_summary(object = M1_stanlmer)
```

```
## Priors for model 'M1_stanlmer'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 1.38
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 0.14 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details
```

```
# Obtain SD of outcome
sd(income$lfpr, na.rm = TRUE)
```

```
## [1] 0.1379202
```

```
#Posterior medians and posterior median absolute deviations
print(M1_stanlmer, digits = 2)
```

```
## stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ 1 + (1 | income)
##  observations: 1234
## ------
##             Median MAD_SD
## (Intercept) 0.98   0.01
## sigma       0.07   0.00
##
## Error terms:
##  Groups   Name        Std.Dev.
##  income   (Intercept) 0.0763
##  Residual             0.0669
## Num. levels: income 114
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.96   0.00
##
## ------
## For info on the priors used see help('prior_summary.stanreg').
```

```r
#Posterior means, posterior standard deviations, 95% credible intervals and Monte Carlo errors
summary(M1_stanlmer,
        pars = c("(Intercept)", "sigma", "Sigma[income:(Intercept),(Intercept)]"),
        probs = c(0.025, 0.975),
        digits = 2)
```

```
## Model Info:
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ 1 + (1 | income)
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 1234
##  groups:       income (114)
## Estimates:
##                                       mean   sd   2.5%   97.5%
## (Intercept)                          0.98   0.01 0.96   0.99
## sigma                                0.07   0.00 0.06   0.07
## Sigma[income:(Intercept),(Intercept)] 0.01   0.00 0.00   0.01
##
## Diagnostics:
##                                       mcse Rhat n_eff
## (Intercept)                          0.00 1.01  337
## sigma                                0.00 1.00 4000
## Sigma[income:(Intercept),(Intercept)] 0.00 1.00 1102
##
## For each parameter, mcse is Monte Carlo standard error, 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).
```

```r
# Extract the posterior draws for all parameters
sims <- as.matrix(M1_stanlmer)
dim(sims)
```

```
## [1] 4000  117
```

```r
para_name <- colnames(sims)
# para_name

##Obtaining means, standard deviations, medians and 95% credible intervals.
# Obtain income-level varying intercept a_j
# draws for overall mean
mu_a_sims <- as.matrix(M1_stanlmer,
                       pars = "(Intercept)")
# draws for income-level error
u_sims <- as.matrix(M1_stanlmer,
                    regex_pars = "b\\[\\(Intercept\\) income\\:")
# draws for income's varying intercepts
a_sims <- as.numeric(mu_a_sims) + u_sims
# Obtain sigma_y and sigma_alpha^2
# draws for sigma_y
s_y_sims <- as.matrix(M1_stanlmer,
                      pars = "sigma")
# draws for sigma_alpha^2
s__alpha_sims <- as.matrix(M1_stanlmer,
                           pars = "Sigma[income:(Intercept),(Intercept)]")

# Compute mean, SD, median, and 95% credible interval of varying intercepts
# Posterior mean and SD of each alpha
a_mean <- apply(X = a_sims,
                MARGIN = 2,
                FUN = mean)
a_sd <- apply(X = a_sims,
              MARGIN = 2,
              FUN = sd)
# posterior mean / SD
```

```r
# Posterior median and 95% credible interval
a_quant <- apply(X = a_sims,
                 MARGIN = 2,
                 FUN = quantile,
                 probs = c(0.025, 0.50, 0.975))
a_quant <- data.frame(t(a_quant))
names(a_quant) <- c("Q2.5", "Q50", "Q97.5")
# Combine summary statistics of posterior simulation draws
a_df <- data.frame(a_mean, a_sd, a_quant)
round(head(a_df), 2)
```

```
##                          a_mean a_sd Q2.5  Q50 Q97.5
## b[(Intercept) income:0]   0.32 0.01 0.30 0.32  0.34
## b[(Intercept) income:1]   0.92 0.01 0.90 0.92  0.94
## b[(Intercept) income:2]   0.97 0.01 0.95 0.97  0.99
## b[(Intercept) income:3]   0.98 0.01 0.96 0.98  1.00
## b[(Intercept) income:4]   0.98 0.01 0.96 0.98  1.00
## b[(Intercept) income:5]   0.98 0.01 0.96 0.98  1.00
```

```r
# Sort dataframe containing an estimated alpha's mean and sd for every school
a_df <- a_df[order(a_df$a_mean), ]
a_df$a_rank <- c(1 : dim(a_df)[1])  # a vector of school rank
# Plot income-level alphas's posterior mean and 95% credible interval
ggplot(data = a_df,
       aes(x = a_rank,
           y = a_mean)) +
  geom_pointrange(aes(ymin = Q2.5,
                      ymax = Q97.5),
                  position = position_jitter(width = 0.1,
                                             height = 0)) +
  geom_hline(yintercept = mean(a_df$a_mean),
             size = 0.5,
             col = "red") +
  scale_x_continuous("Rank",
                     breaks = seq(from = 0,
                                  to = 80,
                                  by = 5)) +
  scale_y_continuous(expression(paste("varying intercept, ", alpha[j]))) +
  theme_bw( base_family = "serif")
```

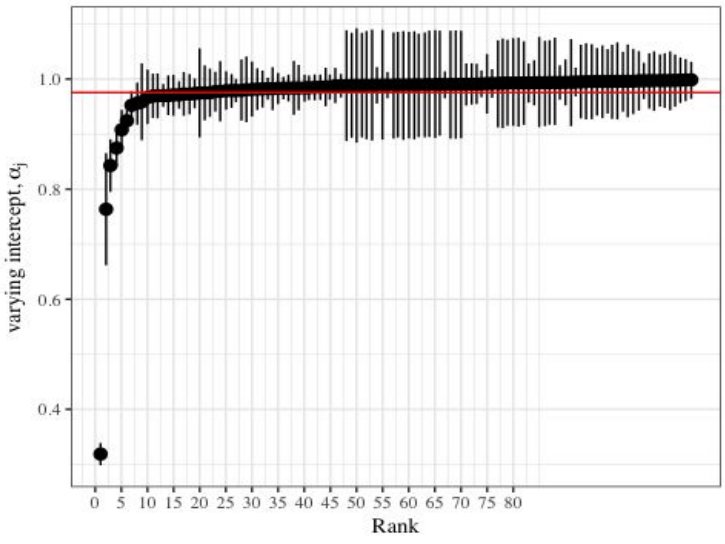

Fig 3.1.2.1: Income-level Alphas's Posterior Mean and 95% Credible Interval

```r
##Making comparisons between individual income level
# The difference between the two income averages (income #10,000 and #100,000)
income_diff <- a_sims[, 1] - a_sims[, 10]

# Investigate differences of two distributions
mean <- mean(income_diff)
sd <- sd(income_diff)
quantile <- quantile(income_diff, probs = c(0.025, 0.50, 0.975))
quantile <- data.frame(t(quantile))
names(quantile) <- c("Q2.5", "Q50", "Q97.5")
diff_df <- data.frame(mean, sd, quantile)
round(diff_df, 2)

##    mean   sd Q2.5   Q50 Q97.5
## 1 -0.67 0.02 -0.7 -0.67 -0.64

#Labor Force Participation Rate diffence between two group of people with yearly income between 10,000 to
20,000 and 100,000 to 110,000
ggplot(data = data.frame(income_diff),
       aes(x = income_diff)) +
  geom_histogram(color = "black",
                 fill = "gray",
                 binwidth = 0.01) +
  scale_x_continuous("Labor Force Participation Rate diffence via income",
                     breaks = seq(from = -0.8,
                                  to = -0.5,
                                  by = 0.05)) +
  geom_vline(xintercept = c(mean(income_diff),
                            quantile(income_diff,
                                     probs = c(0.025, 0.975))),
             colour = "red",
             linetype = "longdash") +
  geom_text(aes(-0.6665, 50, label = "mean = -0.6665"),
            color = "red",
            size = 4) +
  geom_text(aes(-0.62, 100, label = "SD = 0.0159"),
            color = "blue",
            size = 4) +
  theme_bw( base_family = "serif")
```

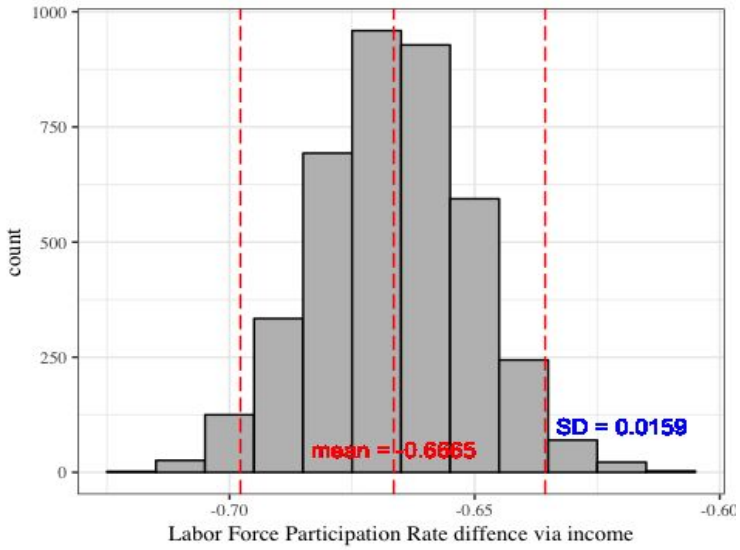

Fig 3.1.2.2: Histogram of the differences

```
plot(M1_stanlmer, "rhat")
```

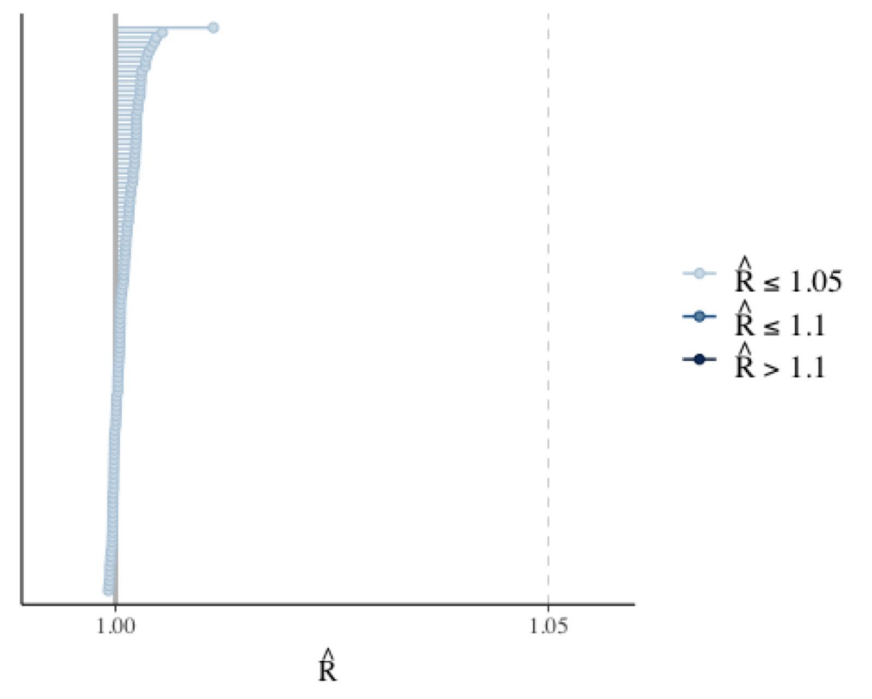

Fig. 3.1.2.3

```
plot(M1_stanlmer, "ess")
```

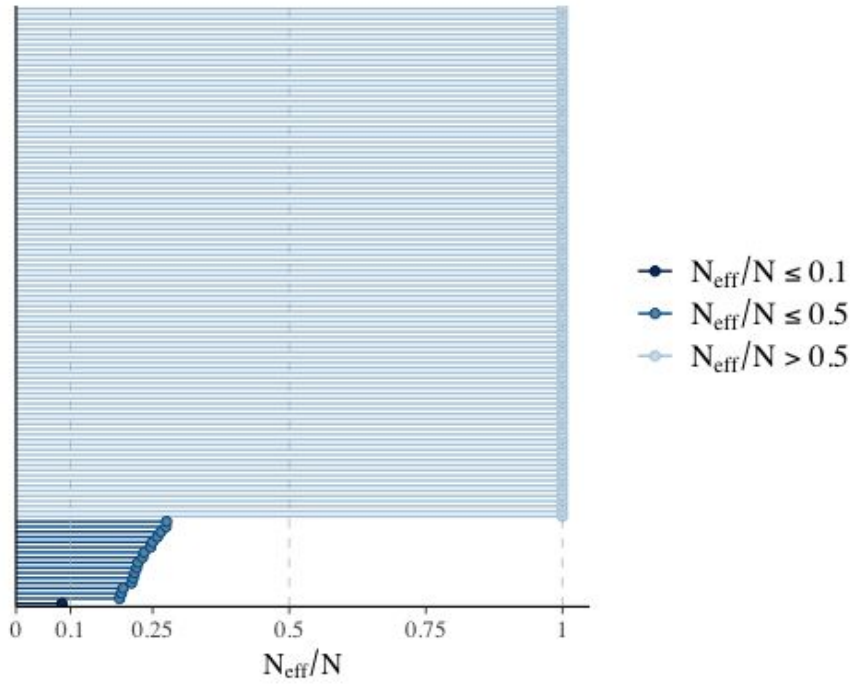

Fig. 3.1.2.4

```
y_pre <- rep(NA, nrow(income))
for (i in 1: nrow(income)){
  a <- rnorm(1,0.98,0.07)
  y_pre[i] <- rnorm(1,a,0.01)
}
```

We can also see the plot in Fig 3.1.2.1 shows the fully Bayes estimates for the income varying intercepts in rank together with their 95% credible intervals. In addition, Fig 3.1.2.2 shows that the expected difference between income group from 10,000 to 20,000 and group from 100,000 to 110,000 comes to -0.67 with a standard deviation of 0.02 and a wide range of uncertainty. The 95% credible interval is [-0.7, -0.64], so we are 95% certain that the true value of the difference between the two levels of income lies within the range given the data. From the $\hat{R}$, we can also see that the income model has converged since the distribution of $\hat{R}$ is less than 1.1.

Above two simple models both show that state and income fit the model for labor force participation rate pretty well, but income offers more concrete prediction results.

3.2 Varying intercept Model with a Single Predictor

To study the relationship between both variables (state & income) with the labor force participation rate, we are going to extend the varying-intercept models by including an indicator variable.

The varying intercept model with one predictor at the state level can be written as $y_{ij} \sim N(\alpha_j + \beta x_{ij}, \sigma_y^2)$ and $\alpha_j \sim N(\mu_\alpha, \sigma_\alpha^2)$. The equation of the average regression line across incomes is $\mu_{ij} = \mu_\alpha + \beta x_{ij}$. The regression lines for specific incomes will be parallel to the average regression line(having the same slope β), but differ in terms of its intercept α_j. We use noninformative prior distributions for the hyperparameters (μ_α and σ_α) as specified in the varying intercept model with no predictors. Additionally, the regression coefficient β is given normal prior distributions with mean 0 and standard deviation of 100. This states, roughly, that we expect this coefficient to be in the range $(-100,100)$, and if the ML estimate is in this range, the prior distribution is providing very little information for the inference.

```
M2_stanlmer <- stan_lmer(formula = lfpr ~ state + (1 | income),
                         data = new_state_income ,
                         prior = normal(location = 0,
                                        scale = 100,
                                        autoscale = FALSE),prior_intercept = normal(location = 0,scale =
100,autoscale = FALSE),seed = 349)

prior_summary(object = M2_stanlmer)

## Priors for model 'M2_stanlmer'
## ------
```

```
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 100)
##
## Coefficients
##  ~ normal(location = 0, scale = 100)
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 0.17 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details

M2_stanlmer

## stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ state + (1 | income)
##  observations: 1000
## ------
##             Median MAD_SD
## (Intercept) 1.0    0.0
## state       0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.1    0.0
##
## Error terms:
##  Groups   Name        Std.Dev.
##  income   (Intercept) 0.095
##  Residual             0.065
## Num. levels: income 68
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

3.3 Varying Intercept and Slope Model with a Single Predictor

We now extend the varying intercept model with a single predictor to allow both the intercept and the slope to vary randomly across incomes using the following model:

$$y_{ij} \sim N(\alpha_j + \beta_j x_{ij}, \sigma_y^2),$$

$$\begin{pmatrix} \alpha_j \\ \beta_j \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_\alpha \\ \mu_\beta \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{pmatrix} \right)$$

We also use stan_lmer to fit Model 3.3. We use the default priors which are mostly similar to what was done in Model 1. Additionally, we are also required to specify a prior for the covariance matrix for α_j and β_j in this model.

```
M3_stanlmer <- stan_lmer(formula = lfpr ~ state + (1 + state | income),
                         data = new_state_income,
                         seed = 349)
prior_summary(object = M3_stanlmer)

## Priors for model 'M3_stanlmer'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 1.67
##
## Coefficients
##  ~ normal(location = 0, scale = 2.5)
##      **adjusted scale = 0.026
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 0.17 (adjusted rate = 1/adjusted scale)
##
## Covariance
##  ~ decov(reg. = 1, conc. = 1, shape = 1, scale = 1)
## ------
## See help('prior_summary.stanreg') for more details

M3_stanlmer

## stan_lmer
##  family:       gaussian [identity]
##  formula:      lfpr ~ state + (1 + state | income)
##  observations: 1000
## ------
##             Median MAD_SD
## (Intercept) 1.0    0.0
## state       0.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 0.1    0.0
##
## Error terms:
##  Groups   Name        Std.Dev. Corr
##  income   (Intercept) 0.0998
##           state       0.0012   -0.29
##  Residual             0.0633
## Num. levels: income 68
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.9    0.0
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

4. Conclusion

Generally, using Multilevel Logistic Regression Model to predict whether a U.S. citizen participates in the job market or not is based on 7 variables, including sex, race, age, skill, annual income, state, and year, which could reach above 70% accuracy.

Based on the inference results, we observe that the model does not capture the impact from the state to the labor force participation very well. Hence, by fixing the state variable, we examine the remaining predictors and notice that the income variable indicates a better fit.

To analyze more about the effect of interactions among variables, we adopt Varying Intercept and Slope Model. However, this does not turn out to a better result. One reason might be that our grouping of regions into 4 parts is too board, which captures less information. The other reason might be that there is little correlation between state and income variable.

Therefore, considering more about the grouping of variables and the relationships among them might better improve our model to predict the U.S. labor force participation rate.