Analyzing the Effect of School-level Sociodemographic Factors on State Math Assessment Results in NYC Public Schools

Applied Bayesian Analysis, Fall '23

Lara Karacasu

2023-12-23

I. Introduction

This project seeks to analyze the effect of school-level sociodemographic indicators on the percentage of NYC public school students passing the NY State Math assessment, accounting for variability at borough levels. The novelty in this project lies in its merging of two distinct NYC public school datasets to derive new insights into indicators for math exam performance via a multi-level model. While some basic summary statistics are publicly available, as published by the DOE, analyses on both datasets in tandem do not exist online to my knowledge, and neither do analyses that leverage this data to account the multi-level nature of the research question.

i. Context

Each year, New York public school students in grades 3-8 take a state-wide math exam known as the NY State Math Test. The assessment contains several different mathematics multiple choice and open-ended questions. Exams are specific to the grade and year, meaning that all NY public school students within the same grade, and within the same year, take the same exam. The exam results are aggregated at the school-level for each grade and year, then released by the Department of Education. There are 4 levels of achievement: 1, 2, 3, and 4. Achieving a 3 or 4 on the exam qualifies as 'passing' the exam.

Additionally, demographic data is collected yearly on these same public schools. This data will be described further in the 'Data' section, but it generally includes school data on socioeconomic variables, ethnicity, gender, English-language learners, and special education rates. I use the term 'sociodemographic' factors to encompass both economic demographics (like the percentage of student on free lunch program) social demographics (percent of students belonging to different races, percentage of male/female student, etc.). This term is used to concisely describe a broad range of social and economic variables.

ii. What it is used for

The exam data itself is used so that NY state can determine school-level mathematics performance. Schools with lower pass rates may receive additional state support, as lower pass rates indicate that the schools may not currently be meeting state-wide mathematics standards for grades 3 - 8. The data itself is released for public use, to inform the public of general school performance in the NY State Math Test. The school-level demographic information is used for similar reasons, as the state often makes financial and political decisions based on school demographics. I will use this data to understand how school-level socioeconomic factors are linked to school pass rates, while controlling for borough-level differences.

iii. Limitations

There are several limitations and assumptions involved. For the sake of brevity, I will list them below:

1. This project specifically uses exam result data from 2006 - 2012. Results cannot be generalized to time periods outside of this range without additional evidence.
2. The DOE data has been aggregated at the school-level. Thus, data from individual students is not tracked, and student-level factors have not been accounted for within the data. That is, the data does not track the individual data of each student, so conclusions can only be made starting at the school-level. Additionally, the data contains district-level data, which I do clean and transform, I only focus on school-within-borough nesting in the scope of this analysis.
3. The school-level accounts do not include features for many other socioeconomic factors that could be at play. There are many potentially significant variables that may have been omitted due to data accessibility, including school-level homelessness rates, parent income levels, single parent household rates, and more. Thus, it is possible that there are unaccounted-for variables that would change result conclusions if accounted for: such is the nature of the data provided by DOE.
4. The final dataset has over 24,000 observations. Additionally, some schools are 'incomplete' in the sense that they do not have values for all grades, all years, or all predictors. In order to both reduce the massive size of the dataset (so that my machine can actually create the models), and handle the missing values, I drop 'incomplete' schools. That is, I only keep schools that have values for all grades, all years, and all predictors. This is a naive approach and could introduce bias if there are patterns in the missing values. Additionally, because this only reduced my dataset to around 13,000 observations – still far too large for my laptop to handle – I limit my analysis to students in grade 8, in the year 2008. The reasoning is that subsetting the data in this way allowed me to control for a given year and grade, while still maintaining differences between schools across the boroughs.

II. Data

The data needed to accomplish this project includes school-level information about sociodemographic factors as well as information about each borough in which each school is located. Such features were not present in a singular dataset, so I decided to merge two different datasets over the 2006 - 2012 time period from NYC Open Data, Department of Education. The datasets are as follows: "2006 - 2012 School Demographics and Accountability Snapshot" and "2006 - 2012 Math Test Results - All Students."

i. Variables

Variables in Demographic Data

The first dataset tracks annual school accounts of NYC public school student populations served by grade, special programs, ethnicity, gender and Title I funded programs.

```
data1 <- read.csv("C:/Users/larak/OneDrive/Documents/Applied Bayesian Analysis/2006_-_2012_School_Demogr
head(data1)
```

```
##      DBN                      Name schoolyear fl_percent frl_percent
## 1 01M015 P.S. 015 ROBERTO CLEMENTE   20052006       89.4          NA
## 2 01M015 P.S. 015 ROBERTO CLEMENTE   20062007       89.4          NA
## 3 01M015 P.S. 015 ROBERTO CLEMENTE   20072008       89.4          NA
## 4 01M015 P.S. 015 ROBERTO CLEMENTE   20082009       89.4          NA
```

```
## 5 01M015 P.S. 015 ROBERTO CLEMENTE   20092010                  96.5
## 6 01M015 P.S. 015 ROBERTO CLEMENTE   20102011                  96.5
##   total_enrollment prek  k grade1 grade2 grade3 grade4 grade5 grade6 grade7
## 1              281   15 36     40     33     38     52     29     38     NA
## 2              243   15 29     39     38     34     42     46     NA     NA
## 3              261   18 43     39     36     38     47     40     NA     NA
## 4              252   17 37     44     32     34     39     49     NA     NA
## 5              208   16 40     28     32     30     24     38     NA     NA
## 6              203   13 37     35     33     30     30     25     NA     NA
##   grade8 grade9 grade10 grade11 grade12 ell_num ell_percent sped_num
## 1     NA     NA      NA      NA      NA      36        12.8       57
## 2     NA     NA      NA      NA      NA      38        15.6       55
## 3     NA     NA      NA      NA      NA      52        19.9       60
## 4     NA     NA      NA      NA      NA      48        19.0       62
## 5     NA     NA      NA      NA      NA      40        19.2       46
## 6     NA     NA      NA      NA      NA      30        14.8       46
##   sped_percent ctt_num selfcontained_num asian_num asian_per black_num
## 1         20.3      25                 9        10       3.6        74
## 2         22.6      19                15        18       7.4        68
## 3         23.0      20                14        16       6.1        77
## 4         24.6      21                17        16       6.3        75
## 5         22.1      14                14        16       7.7        67
## 6         22.7      21                 9        13       6.4        75
##   black_per hispanic_num hispanic_per white_num white_per male_num male_per
## 1      26.3          189         67.3         5       1.8      158     56.2
## 2      28.0          153         63.0         4       1.6      140     57.6
## 3      29.5          157         60.2         7       2.7      143     54.8
## 4      29.8          149         59.1         7       2.8      149     59.1
## 5      32.2          118         56.7         6       2.9      124     59.6
## 6      36.9          110         54.2         4       2.0      113     55.7
##   female_num female_per
## 1        123       43.8
## 2        103       42.4
## 3        118       45.2
## 4        103       40.9
## 5         84       40.4
## 6         90       44.3
```

```
dim(data1)
```

```
## [1] 10075    38
```

As shown above, there are many columns in the dataset, but only a few will be used. These variables are as follows:

1. DBN: a unique NYC school identifier that includes district (first two characters]), borough (third character), and school number (fourth through sixth characters). A DBN of 01M015 would represent District 01, in Manhattan, for School 15.
2. Name: school name.
3. Schoolyear: school year in which the exam was conducted.
4. frl_percent: percentage of students eligible for free or reduced cost lunch.

5. fl_percent: percentage of students eligible for free lunch.
6. grade: prek through grade12 columns are represented, but only grade3 through grade8 will be relevant.
7. ell_percent: percentage of English-language learners within the school grade during a given year.
8. white_per, asian_per, black_per, hispanic_per: percentage of students who identify as the specified race within the school grade during a given year.

Variables in Math Exam Data

The second dataset tracks NYC school performance on state math exams. The response variable will be 'Pct Level 3 and 4,' the percentage of students at the school, for a given grade level and year, who scored proficient or advanced proficient in the math assessment. This is essentially the percent of students who passed the state math exam.

```
data2 <- read.csv("C:/Users/larak/OneDrive/Documents/Applied Bayesian Analysis/2006_-_2012__Math_Test_R
head(data2)
```

```
##      DBN Grade Year  Demographic Number.Tested Mean.Scale.Score Num.Level.1
## 1 01M015     3 2006 All Students            39              667           2
## 2 01M015     3 2007 All Students            31              672           2
## 3 01M015     3 2008 All Students            37              668           0
## 4 01M015     3 2009 All Students            33              668           0
## 5 01M015     3 2010 All Students            26              677           6
## 6 01M015     3 2011 All Students            28              671          10
##   Pct.Level.1 Num.Level.2 Pct.Level.2 Num.Level.3 Pct.Level.3 Num.Level.4
## 1         5.1          11        28.2          20        51.3           6
## 2         6.5           3         9.7          22          71           4
## 3           0           6        16.2          29        78.4           2
## 4           0           4        12.1          28        84.8           1
## 5        23.1          12        46.2           6        23.1           2
## 6        35.7          13        46.4           5        17.9           0
##   Pct.Level.4 Num.Level.3.and.4 Pct.Level.3.and.4
## 1        15.4                26              66.7
## 2        12.9                26              83.9
## 3         5.4                31              83.8
## 4           3                29              87.9
## 5         7.7                 8              30.8
## 6           0                 5              17.9
```

```
dim(data2)
```

```
## [1] 33461    16
```

As shown above, there are many column in the dataset, but only a few will be used. The relevant variables are as follows:

1. DBN: Same as the other dataset.
2. Grade: Same as the other dataset.
3. Year: Same as the other dataset, but represented as the singular year during the spring in which the exam is taken.
4. Pct.Level.3.and.4: Percentage of students tested within a given grade, year, and school, who passed the state math exam.

ii. Research Question

The research question is as follows: How do school-level socioeconomic predictors, such as enrollment and demographics, influence the percentage of students passing the assessment (Pct Level 3 or 4) across different schools, accounting for variability at the borough level?

iii. Statistical Method

I use a hierarchical Beta regression models to analyze the data. The response variable is the percent of students passing the state math exam, and subsequently, the models have a Beta-distributed likelihood for the response variable. Additionally, a multi-level model is required to account for the nested levels o. There are three levels in my dataset: School > District > Borough, and I will include school-level random intercepts for districts and boroughs. I will include fixed effects for grade level and year. I will also include school-level fixed effects for the sociodemographic predictors, including the percentage of students on FRL (free and reduced lunch) or FL (free lunch) as well as racial demographics.

iv. Data preprocessing

A preprocessing pipeline is needed to ensure data quality and comparability prior to the analysis. I will explain and demonstrate the pipeline within this document for the sake of completeness. I recognize that this pipeline will lengthen the report, so please feel free to just skim over this section if necessary.

The steps are as follows:

1. Dropping columns

We will only keep columns that are relevant to the analysis – these are the columns mentioned in the previous section.

```
library(dplyr)
library(tidyverse)
```

Dropping columns from demographic data

```
## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3
```

```
library(rethinking)
```

```
## Warning: package 'StanHeaders' was built under R version 4.2.3
```

```
library(stringr)
```

```
clean1 <- data1[c("DBN", "Name", "schoolyear", "fl_percent", "frl_percent", "grade3", "grade4", "grade5
head(clean1)
```

```
##      DBN                     Name schoolyear fl_percent frl_percent grade3
## 1 01M015 P.S. 015 ROBERTO CLEMENTE   20052006       89.4          NA     38
## 2 01M015 P.S. 015 ROBERTO CLEMENTE   20062007       89.4          NA     34
## 3 01M015 P.S. 015 ROBERTO CLEMENTE   20072008       89.4          NA     38
## 4 01M015 P.S. 015 ROBERTO CLEMENTE   20082009       89.4          NA     34
## 5 01M015 P.S. 015 ROBERTO CLEMENTE   20092010                     96.5     30
## 6 01M015 P.S. 015 ROBERTO CLEMENTE   20102011                     96.5     30
##   grade4 grade5 grade6 grade7 grade8 ell_percent asian_per black_per
## 1     52     29     38     NA     NA        12.8       3.6      26.3
## 2     42     46     NA     NA     NA        15.6       7.4      28.0
## 3     47     40     NA     NA     NA        19.9       6.1      29.5
## 4     39     49     NA     NA     NA        19.0       6.3      29.8
## 5     24     38     NA     NA     NA        19.2       7.7      32.2
## 6     30     25     NA     NA     NA        14.8       6.4      36.9
##   hispanic_per white_per
## 1         67.3       1.8
## 2         63.0       1.6
## 3         60.2       2.7
## 4         59.1       2.8
## 5         56.7       2.9
## 6         54.2       2.0
```

```
clean2 <- data2[c("DBN", "Grade", "Year", "Pct.Level.3.and.4")]
head(clean2)
```

Dropping columns from exam data

```
##      DBN Grade Year Pct.Level.3.and.4
## 1 01M015     3 2006              66.7
## 2 01M015     3 2007              83.9
## 3 01M015     3 2008              83.8
## 4 01M015     3 2009              87.9
## 5 01M015     3 2010              30.8
## 6 01M015     3 2011              17.9
```

2. Data standardization

I will: 1. Standardize column names across the two datasets.

```
clean1 <- clean1 %>% rename('Year' = 'schoolyear')
```

2. Standardize school year representation. The same school year is represented differently in the two datasets (the first by the year in the spring, the second by range of the school year).

```
clean1 <- clean1 %>% mutate(Year = str_sub(Year, -4, -1))
```

3. Reformat the first dataset so that a row represents a combination of unique DBN, grade, and year.

```
clean1 <- clean1 %>%
  pivot_longer(cols = starts_with("grade"),
               names_to = "Grade",
               values_drop_na = FALSE) %>%
  mutate(Grade = readr::parse_number(Grade)) %>%
  rename('Number.Tested' = 'value') %>%
  relocate(Grade, .before = names(clean1)[3])
head(clean1)
```

```
## # A tibble: 6 x 12
##   DBN   Name  Grade Year  fl_percent frl_percent ell_percent asian_per black_per
##   <chr> <chr> <dbl> <chr> <chr>            <dbl>       <dbl>     <dbl>     <dbl>
## 1 01M0~ P.S.~     3 2006  89.4                NA        12.8       3.6      26.3
## 2 01M0~ P.S.~     4 2006  89.4                NA        12.8       3.6      26.3
## 3 01M0~ P.S.~     5 2006  89.4                NA        12.8       3.6      26.3
## 4 01M0~ P.S.~     6 2006  89.4                NA        12.8       3.6      26.3
## 5 01M0~ P.S.~     7 2006  89.4                NA        12.8       3.6      26.3
## 6 01M0~ P.S.~     8 2006  89.4                NA        12.8       3.6      26.3
## # i 3 more variables: hispanic_per <dbl>, white_per <dbl>, Number.Tested <int>
```

3. Merging data

We must perform an inner join on the DBN, school year, and grade level.

```
class(clean2$Grade) <- 'double'
```

```
## Warning in class(clean2$Grade) <- "double": NAs introduced by coercion
```

```
class(clean1$Year) <- 'integer'
class(clean1$fl_percent) <- 'double'
```

```
## Warning in class(clean1$fl_percent) <- "double": NAs introduced by coercion
```

```
df <- inner_join(clean1, clean2, by = c("DBN", "Grade", "Year"))
head(df)
```

```
## # A tibble: 6 x 13
##   DBN   Name  Grade  Year fl_percent frl_percent ell_percent asian_per black_per
##   <chr> <chr> <dbl> <int>      <dbl>       <dbl>       <dbl>     <dbl>     <dbl>
## 1 01M0~ P.S.~     3  2006       89.4          NA        12.8       3.6      26.3
## 2 01M0~ P.S.~     4  2006       89.4          NA        12.8       3.6      26.3
## 3 01M0~ P.S.~     5  2006       89.4          NA        12.8       3.6      26.3
## 4 01M0~ P.S.~     6  2006       89.4          NA        12.8       3.6      26.3
## 5 01M0~ P.S.~     3  2007       89.4          NA        15.6       7.4      28
## 6 01M0~ P.S.~     4  2007       89.4          NA        15.6       7.4      28
## # i 4 more variables: hispanic_per <dbl>, white_per <dbl>, Number.Tested <int>,
## #   Pct.Level.3.and.4 <chr>
```

```
dim(df)
```

```
## [1] 24281    13
```

4. Splitting DBN

Splitting the DBN column into three separate columns: district, borough, and school name. Doing so will enable me to conduct a multilevel analysis, as one row will correspond to a given school, which are nested within boroughs.

```
df$District <- substr(df$DBN, 1, 2)
df$Borough <- substr(df$DBN, 3, 3)
df$School.Number <- substr(df$DBN, 4, 6)
df <- df %>%
  relocate(District, .before = names(df)[2]) %>%
  relocate(Borough, .before = names(df)[2]) %>%
  relocate(School.Number, .before = names(df)[2]) %>%
  relocate(Pct.Level.3.and.4, .before = 8)
class(df$Pct.Level.3.and.4) <- 'double'
```

```
## Warning in class(df$Pct.Level.3.and.4) <- "double": NAs introduced by coercion
```

```
head(df, 20)
```

```
## # A tibble: 20 x 16
##    DBN    District Borough School.Number Name      Grade  Year Pct.Level.3.and.4
##    <chr>  <chr>    <chr>   <chr>         <chr>     <dbl> <int>              <dbl>
##  1 01M015 01       M       015           "P.S. 01~     3  2006               66.7
##  2 01M015 01       M       015           "P.S. 01~     4  2006               22.4
##  3 01M015 01       M       015           "P.S. 01~     5  2006               32.3
##  4 01M015 01       M       015           "P.S. 01~     6  2006               43.6
##  5 01M015 01       M       015           "P.S. 01~     3  2007               83.9
##  6 01M015 01       M       015           "P.S. 01~     4  2007               57.5
##  7 01M015 01       M       015           "P.S. 01~     5  2007               32
##  8 01M015 01       M       015           "P.S. 01~     3  2008               83.8
##  9 01M015 01       M       015           "P.S. 01~     4  2008               51.2
## 10 01M015 01       M       015           "P.S. 01~     5  2008               61.1
## 11 01M015 01       M       015           "P.S. 01~     3  2009               87.9
## 12 01M015 01       M       015           "P.S. 01~     4  2009               51.3
## 13 01M015 01       M       015           "P.S. 01~     5  2009               59
## 14 01M015 01       M       015           "P.S. 01~     3  2010               30.8
## 15 01M015 01       M       015           "P.S. 01~     4  2010               34.5
## 16 01M015 01       M       015           "P.S. 01~     5  2010               28.6
## 17 01M015 01       M       015           "P.S. 01~     3  2011               17.9
## 18 01M015 01       M       015           "P.S. 01~     4  2011               39.3
## 19 01M015 01       M       015           "P.S. 01~     5  2011               48
## 20 01M015 01       M       015           "P.S. 01~     3  2012               33.3
## # i 8 more variables: fl_percent <dbl>, frl_percent <dbl>, ell_percent <dbl>,
## #   asian_per <dbl>, black_per <dbl>, hispanic_per <dbl>, white_per <dbl>,
## #   Number.Tested <int>
```

```
dim(df)
```

```
## [1] 24281    16
```

5. Handling missing data and downsizing

As we can see, I currently have about 24,000 rows (each of which represent a different school/grade/year combination). This is far too large to run any models efficiently. I will need to reduce the length of the data substantially, to around 1000 rows or so. Additionally, we can see that there are some NA cells in the data – each school in the data only has a value for either frl_lunch (free or reduced lunch) OR free lunch. These variables are also likely to be multicollinear due to fl_lunch being a subset of frl_lunch, so we should only keep one of them.

Let's check which has higher proportion in the dataset.

```
sum(!is.na(df['frl_percent']))
```

```
## [1] 10768
```

```
sum(!is.na(df['fl_percent']))
```

```
## [1] 13500
```

As we can see, more rows use the 'free lunch' designation over 'free or reduced lunch'. So, we shall drop the frl_percent column.

```
df <- df %>% select(-frl_percent, -Name, -Number.Tested)
final_df <- drop_na(df)
head(final_df)
```

```
## # A tibble: 6 x 13
##   DBN    District Borough School.Number Grade  Year Pct.Level.3.and.4 fl_percent
##   <chr>  <chr>    <chr>   <chr>         <dbl> <int>             <dbl>      <dbl>
## 1 01M015 01       M       015               3  2006              66.7       89.4
## 2 01M015 01       M       015               4  2006              22.4       89.4
## 3 01M015 01       M       015               5  2006              32.3       89.4
## 4 01M015 01       M       015               6  2006              43.6       89.4
## 5 01M015 01       M       015               3  2007              83.9       89.4
## 6 01M015 01       M       015               4  2007              57.5       89.4
## # i 5 more variables: ell_percent <dbl>, asian_per <dbl>, black_per <dbl>,
## #   hispanic_per <dbl>, white_per <dbl>
```

```
dim(final_df)
```

```
## [1] 13384    13
```

Even after dropping all rows with any NA values, the dataframe has over 13,000 tuples.

Note: I initially intended to use this full dataset, but it was still far too large for the model to run properly on my laptop. Thus, I will limit my analysis to a single year: 2009. Doing this will also help control for some variability in the model that might be introduced by the temporal element. I chose the year 2009 simply because it had the most observations out of the years within the dataset.

```r
final_df <- final_df[final_df$Year == 2008, ]
final_df <- final_df %>% select(-Year)
print(final_df)
```

```
## # A tibble: 3,390 x 12
##    DBN    District Borough School.Number Grade Pct.Level.3.and.4 fl_percent
##    <chr>  <chr>    <chr>   <chr>         <dbl>             <dbl>      <dbl>
##  1 01M015 01       M       015               3              83.8       89.4
##  2 01M015 01       M       015               4              51.2       89.4
##  3 01M015 01       M       015               5              61.1       89.4
##  4 01M019 01       M       019               3              80.4       61.5
##  5 01M019 01       M       019               4              75.5       61.5
##  6 01M019 01       M       019               5              76.6       61.5
##  7 01M020 01       M       020               3              95.1       92.5
##  8 01M020 01       M       020               4              84.7       92.5
##  9 01M020 01       M       020               5              87.1       92.5
## 10 01M034 01       M       034               3              82.4       76.8
## # i 3,380 more rows
## # i 5 more variables: ell_percent <dbl>, asian_per <dbl>, black_per <dbl>,
## #   hispanic_per <dbl>, white_per <dbl>
```

The cleaned dataset has about 3200 rows. Unfortunately, this is still far too many for our purposes, so we will need to refine our analysis further. I will additionally

```r
final_df <- final_df[final_df$Grade == 8, ]
final_df <- final_df %>% select(-Grade)
head(final_df)
```

```
## # A tibble: 6 x 11
##   DBN    District Borough School.Number Pct.Level.3.and.4 fl_percent ell_percent
##   <chr>  <chr>    <chr>   <chr>                     <dbl>      <dbl>       <dbl>
## 1 01M034 01       M       034                        46.3       76.8         8.3
## 2 01M140 01       M       140                        61.9       79.9        14.7
## 3 01M184 01       M       184                        98.2       72.2        12
## 4 01M188 01       M       188                        69.6       89.6        14.7
## 5 01M292 01       M       292                        44.3       99.5        10.1
## 6 01M301 01       M       301                        41.5       62.8         7.1
## # i 4 more variables: asian_per <dbl>, black_per <dbl>, hispanic_per <dbl>,
## #   white_per <dbl>
```

~400 rows is a reasonable number of rows to ensure fast processing time on my device. Although this will limit the generalizability of the conclusions, it will also control for additional variables preemptively, which could benefit model convergence.

III. Priors

I chose a beta regression model due to the response variable being a proportion (percentage of students who passed the exam). Thus, the priors will be those of beta regression, and we also need to account for the hierarchical structure of schools within districts within boroughs.

Note: after attempting the analysis portion, I initially ran into many strange RStan model compilation errors. After much debugging, I realized that one of them was due to the model breaking due to the presence

of some 1.000 values in the dataset – quite problematic for my beta regression. To deal with this challenge without sacrificing my beta regression (which is appropriate for my percentage response variable), I apply a data transformation to adjust the boundaries for the response variable to be strictly within (0, 1). Such a transformation only marginally changes the actual pass rate values, as I choose a very small epsilon value for the transformation.

Additionally, I had some trouble with rogue values from the 'out from the wild' data, so I employed some of the preprocessing steps from class. I use a unique integer identifier for each school, district, and borough. I additionally scale the percentage columns to range from [0, 1] for consistency.

```
data3 <- final_df

# For Beta regression
data3$Pass_Rate <- data3$Pct.Level.3.and.4 / 100
epsilon <- 1e-4
data3$Pass_Rate <- pmin(pmax(data3$Pass_Rate, epsilon), 1 - epsilon)

# Encoding factor variables as integers so that each has a unique numerical identifier
data3$School_Number <- as.integer(as.factor(data3$School.Number))
data3$District <- as.integer(as.factor(data3$District))
data3$Borough <- as.integer(as.factor(data3$Borough))
data3$School <- 1:nrow(data3)
data3$DBN<- as.integer(as.factor(data3$DBN))

# No need for these because they're redundant with existing columns
data3 <- data3 %>% select(-School.Number, -School_Number, -Pct.Level.3.and.4, -DBN)

# Scale the percent columns between 0 and 1
divide_perc_columns_by_100 <- function(df) {
    perc_cols <- grep("per", names(df))
    df[, perc_cols] <- df[, perc_cols] / 100
    return(df)
}
data3 <- divide_perc_columns_by_100(data3) %>% relocate(School, .before = 1) %>% relocate(Pass_Rate, .b
head(data3, 10)
```

```
## # A tibble: 10 x 10
##    School District Borough Pass_Rate fl_percent ell_percent asian_per black_per
##     <int>    <int>   <int>     <dbl>      <dbl>       <dbl>     <dbl>     <dbl>
##  1      1        1       2     0.463      0.768       0.083     0.05      0.21
##  2      2        1       2     0.619      0.799       0.147     0.042     0.176
##  3      3        1       2     0.982      0.722       0.12      0.829     0.056
##  4      4        1       2     0.696      0.896       0.147     0.024     0.319
##  5      5        1       2     0.443      0.995       0.101     0.109     0.307
##  6      6        1       2     0.415      0.628       0.071     0.109     0.209
##  7      7        1       2     0.343      0.738       0.061     0.061     0.294
##  8      8        1       2     0.717      0.866       0.029     0.154     0.172
##  9      9        1       2     0.584      0.706       0.063     0.056     0.269
## 10     10        1       2     1.00       0.118       0.005     0.224     0.097
## # i 2 more variables: hispanic_per <dbl>, white_per <dbl>
```

```
dim(data3)
```

```
## [1] 398  10
```

Finally, it looks suitable for the multi-level models! Note that I used unique integers for the levels, so the integers do not have direct intrinsic meanings: they only encode my factors.

Analysis 1: Uninformative Priors

Since EOD has not released any formal analyses using these datasets, it makes sense to use uninformative priors for one choice. We can break down the priors by choice of fixed vs. random effects for the model.

Fixed effects variables are as follows: School-level sociodemographic factors (fl_percent, ell_percent, white_per, black_per, hispanic_per, asian_per). These variables are assumed to have a consistent impact across all schools and boroughs. For example, the influence of a specific grade level on the pass rate is considered uniform across the dataset.

Random effects variables are as follows: School and Borough. These variables represent different hierarchical levels, capturing the nested nature of the data (schools within boroughs). The random effects account for the variability at different levels of the model, acknowledging that different boroughs might have unique characteristics influencing the outcome.

Model 1.1: Nested Hierarchical Model, Uninformative Priors

For the fixed effects, I will choose normally distributed priors with a mean of zero and a large standard deviation (e.g., dnorm(0, 1)). This choice indicates that before seeing the data, we have no strong expectations about the magnitude of the effects.

Our priors at the school level will be normal distribution centered around the borough mean, with an unknown standard deviation (e.g., a_school[school] ~ dnorm(a_borough[borough], sigma_school)). This reflects that each school's effect is a deviation from its borough's average effect. At the base borough level, we simply use another uninformative prior (e.g., a_borough[borough] ~ dnorm(0, 1)).

My rationale for the use of broad normal distributions as uninformative priors allows the data to primarily inform the posterior estimates. This is particularly useful because there's a lack of strong prior knowledge or assumptions about the parameters, as the data hasn't been publicly analyzed by EOD.

```
set.seed(10)
m_uninformative_1 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),
    logit(mu) <- a_school[School] + a_borough[Borough] +
                 b_fl*fl_percent + b_ell*ell_percent +
                 b_asian*asian_per + b_black*black_per + b_hispanic*hispanic_per + b_white*white_per,

    # Priors for fixed effects
    b_fl ~ dnorm(0, 1),
    b_ell ~ dnorm(0, 1),
    b_asian ~ dnorm(0, 1),
    b_black ~ dnorm(0, 1),
    b_hispanic ~ dnorm(0, 1),
    b_white ~ dnorm(0, 1),

    # Random effects
    a_school[School] ~ dnorm(a_borough[Borough], sigma_school),
    a_borough[Borough] ~ dnorm(0, 1),

    # Hyperpriors
```

```
    sigma_school ~ dexp(1),
    sigma_borough ~ dexp(1),
    phi ~ dexp(1)
  ),
  data = data3,
  chains = 4,
  cores = 4,
  log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 24.2 seconds.
```

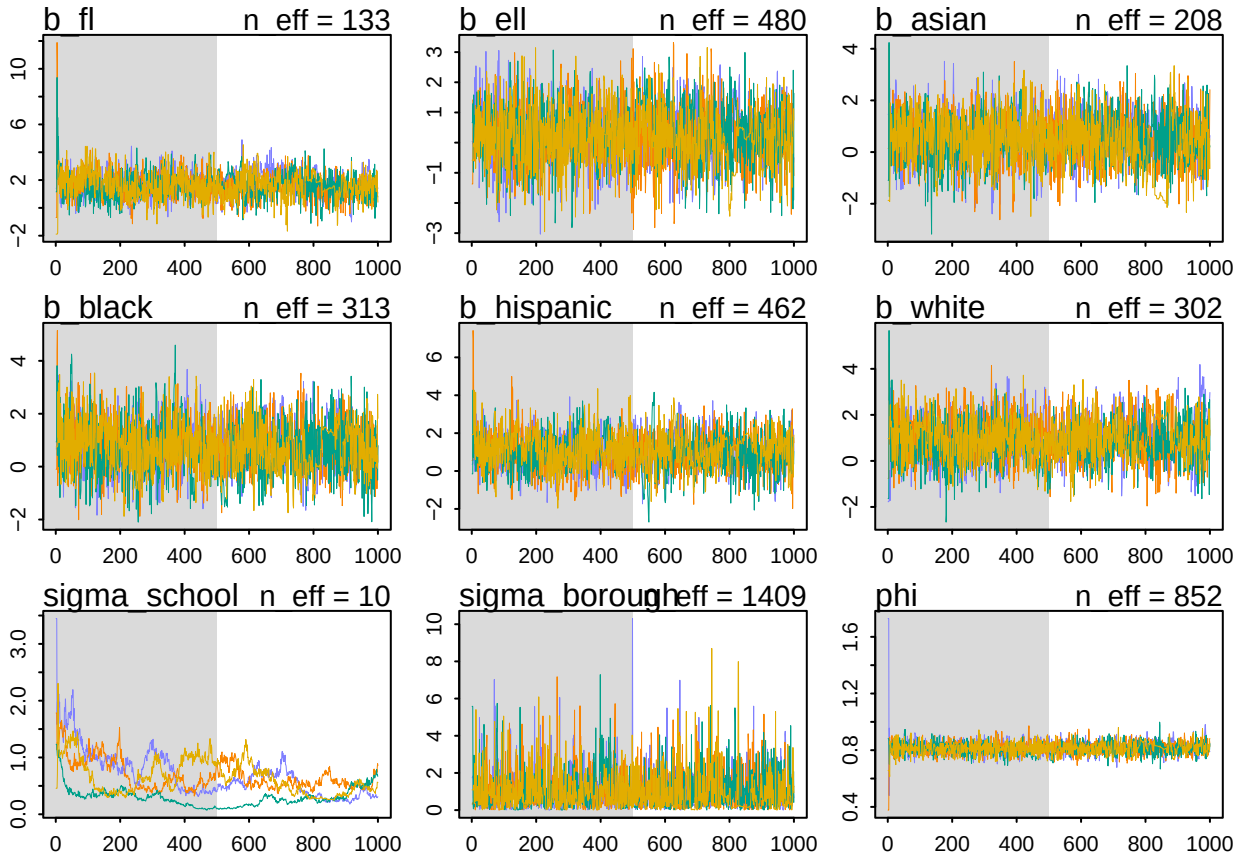
```
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 27.1 seconds.
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 27.2 seconds.
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 40.9 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 29.8 seconds.
## Total execution time: 41.2 seconds.
```

```
precis(m_uninformative_1)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                    mean        sd        5.5%     94.5%      n_eff    Rhat4
## b_fl          1.5938502 0.8668098  0.15954767 2.9398168  133.29258 1.029817
## b_ell         0.1992459 0.9838172 -1.34672640 1.7437288  480.07067 1.014234
## b_asian       0.4773450 0.9677921 -1.08906285 2.0824969  208.14719 1.024835
## b_black       0.7226701 0.9326808 -0.76439739 2.2209592  313.37220 1.008311
## b_hispanic    1.0042553 0.9582708 -0.56753402 2.4928710  461.60965 1.015025
## b_white       0.9446083 0.9180490 -0.49900127 2.4451620  302.38385 1.009621
## sigma_school  0.4759434 0.2220469  0.16511435 0.8854724   10.10297 1.684106
## sigma_borough 0.9835593 1.0151811  0.05782783 2.8880179 1409.39291 1.002379
## phi           0.8161756 0.0424961  0.74810833 0.8846406  851.96690 1.004495
```

```
traceplot(m_uninformative_1, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white", "
```



Model 1.1.2: Attempting to Fix Convergence

It seems that the nested model had some convergence issues, specifically with sigma_school having a high Rhat value and low n_eff. I conjecture that it could be due to the mu parameter becoming exactly 0 or 1.

This can happen due to the values of predictors and random effects leading to extreme values on the logit scale, which, when transformed back to the probability scale, become 0 or 1. I attempt to fix it with a direct logistic regression transformation: I'll transform the predictor directly using the inv_logit function to ensure that mu stays between 0 and 1, exclusive, because the logistic function inherently bounds the output between 0 and 1.

```
set.seed(1)
epsilon <- 1e-4
m_uninformative_12 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),

    # Applying logistic transformation with a safeguard for mu
    mu <- inv_logit(a_school[School] + a_borough[Borough] +
                    b_fl*fl_percent + b_ell*ell_percent +
                    b_asian*asian_per + b_black*black_per +
                    b_hispanic*hispanic_per + b_white*white_per),

    # Priors for fixed effects
    b_fl ~ dnorm(0, 1),
    b_ell ~ dnorm(0, 1),
```

```
    b_asian ~ dnorm(0, 1),
    b_black ~ dnorm(0, 1),
    b_hispanic ~ dnorm(0, 1),
    b_white ~ dnorm(0, 1),

    # Random effects
    a_school[School] ~ dnorm(a_borough[Borough], sigma_school),
    a_borough[Borough] ~ dnorm(0, 1),

    # Hyperpriors
    sigma_school ~ dexp(1),
    sigma_borough ~ dexp(1),
    phi ~ dexp(1)
  ),
  data = data3, chains = 4, cores = 4, log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
```

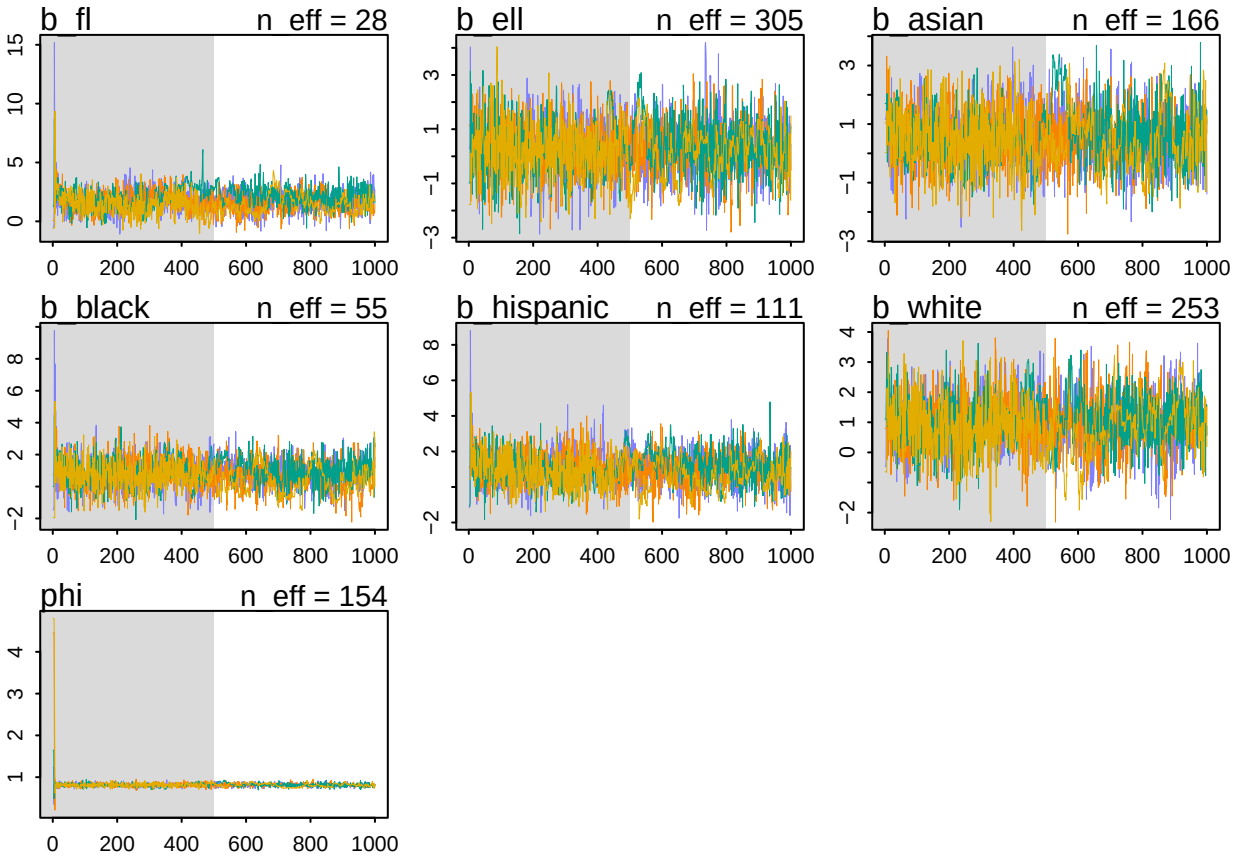
```
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 26.7 seconds.
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 29.5 seconds.
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 39.3 seconds.
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 64.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 39.9 seconds.
## Total execution time: 64.3 seconds.
```

```
precis(m_uninformative_12)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                   mean         sd        5.5%     94.5%       n_eff     Rhat4
## b_fl          1.7123293 0.87768613  0.38377157 3.1413567   27.946974 1.0756354
## b_ell         0.2470829 0.98817212 -1.31959235 1.8637633  304.811732 1.0132366
## b_asian       0.5762884 0.99499763 -0.96640943 2.3087165  166.169438 1.0186943
## b_black       0.7136855 0.89819497 -0.67841509 2.1332132   55.068757 1.0606193
## b_hispanic    0.9648204 0.91777938 -0.49005541 2.3869430  110.618645 1.0474395
## b_white       0.9516911 0.91060487 -0.56162277 2.4066346  252.691517 1.0196859
## sigma_school  0.4955386 0.38404421  0.10354795 1.2371259    5.048583 2.2279309
## sigma_borough 0.9832838 0.95561108  0.06636331 2.8037583 1111.906529 0.9999925
## phi           0.8140577 0.04069132  0.74660784 0.8799028  154.172966 1.0092930
```

```
traceplot(m_uninformative_12, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white",
```



Unfortunately, the technique did not fix the convergence issues in the nested hierarchical model. Interestingly, it did eliminate the warnings that initially came up when running the previous model. We shall attempt to try another version to see if we can achieve model convergence: a non-nested hierarchical model. Despite schools logically being nested within boroughs, it is possible that the variability within schools might not differ significantly across different boroughs, or the borough-level effects might be minimal compared to the school-level effects. This could lead to difficulties in estimating higher-level (borough) parameters accurately. Thus, we shall try a final non-nested alternative.

Model 1.2: Non-Nested Hierarchical Model, Uninformative Priors

Now, I will try another version of the model: a non-nested hierarchical model, again with uninformative priors. Based on the nature of the underlying nature – eg. one school belongs to exactly one borough – it seems that a nested hierarchical beta regression model would be most fitting for the data. However, I will also build a non-nested hierarchical model to see if this would help with convergence issues.

```
set.seed(1)
m_uninformative_2 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),
    logit(mu) <- a_school[School] + a_borough[Borough] +
                 b_fl*fl_percent + b_ell*ell_percent +
                 b_asian*asian_per + b_black*black_per +
                 b_hispanic*hispanic_per + b_white*white_per,
    a_school[School] ~ dnorm(0, 1),
    a_borough[Borough] ~ dnorm(0, 1),
    b_fl ~ dnorm(0, 1),
```

```
    b_ell ~ dnorm(0, 1),
    b_asian ~ dnorm(0, 1),
    b_black ~ dnorm(0, 1),
    b_hispanic ~ dnorm(0, 1),
    b_white ~ dnorm(0, 1),
    phi ~ dexp(1)
  ),
  data = data3, chains = 4, cores = 4, log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
```

```
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 15.4 seconds.
## Chain 2 finished in 15.5 seconds.
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 16.3 seconds.
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 17.0 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 16.1 seconds.
## Total execution time: 17.1 seconds.
```

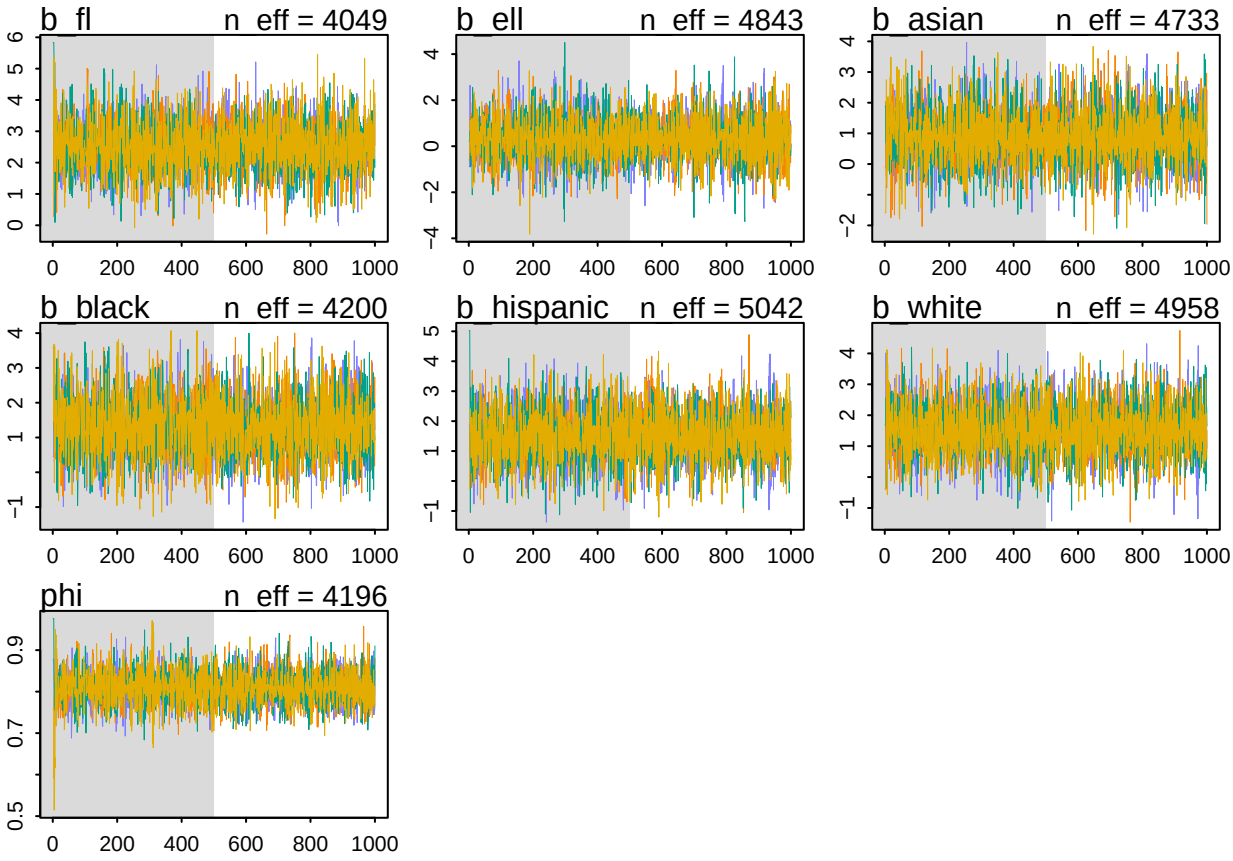
```
precis(m_uninformative_2)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                 mean        sd        5.5%     94.5%    n_eff     Rhat4
## b_fl       2.5002771 0.8147455  1.23578205 3.8404749 4049.254 1.0014483
## b_ell      0.3526137 0.9826021 -1.23790465 1.9216897 4842.808 0.9990182
## b_asian    0.9003727 0.9208430 -0.54317403 2.3990207 4733.188 0.9986044
## b_black    1.2987762 0.8556116 -0.06509884 2.7240076 4200.373 0.9988278
## b_hispanic 1.5206952 0.8646241  0.16485444 2.9291215 5041.679 1.0004024
## b_white    1.5135407 0.8814585  0.12260334 2.9713757 4958.072 0.9984090
## phi        0.8091592 0.0408646  0.74248740 0.8739219 4195.989 0.9984858
```

The Rhat values for b_fl, b_ell, b_asian, b_black, b_white, and phi are all 1. This seems to indicate that the model_uninformative_2 properly converged. Let's also check the traceplot to ensure that this is the case.

```
traceplot(m_uninformative_2, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white", "
```



They do seem to be "fuzzy," normal-looking traceplot that converge. There aren't any clear abnormalities within the traceplot. Thus, we can say that the m_uninformative_2, the model with uninformative and non-nested priors, converged.

Analysis 2: Weakly Informative Priors

Now, let's find some more informative priors. We can look to past research to come up with some ideas for better priors.

Although I haven't seen any work done with these specific datasets (or specific math exam), there have certainly been previous works that study the influence of different predictors on math performance in general.

This study by the Economic Policy Institute examined the impact of socioeconomics and race on standardized academic test scores across different educational levels. They found that, by high school, socioeconomic status (for which free lunch is commonly used as a proxy) is a strong predictor of academic performance on Language Arts and Math exams. They also found that ELL, to a lesser degree than SES, is a significant predictor of performance. and racial factors. Finally, they found that race is also a significant predictor. Thus, based on these findings, I will adjust my own priors by: increase the mean for the prior on fl_percent and decrease the standard deviation, increase the mean for the prior on ell_district, and keep the standard deviations high for the racial predictors.

Let's use the same model set-ups as before, with the same logic, but change the priors to reflect the previous literature better.

Model 2.1: Nested Hierarchical Model, Informative Priors

```
set.seed(1)
m_informative_1 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),
    logit(mu) <- a_school[School] + a_borough[Borough] +
                 b_fl*fl_percent + b_ell*ell_percent +
                 b_asian*asian_per + b_black*black_per +
                 b_hispanic*hispanic_per + b_white*white_per,

    a_school[School] ~ dnorm(a_borough[Borough], sigma_school),
    a_borough[Borough] ~ dnorm(1, 2),
    b_fl ~ dnorm(1, 0.5),
    b_ell ~ dnorm(0.5, 1),
    b_asian ~ dnorm(0.2, 1),
    b_black ~ dnorm(0.2, 1),
    b_hispanic ~ dnorm(0.2, 1.),
    b_white ~ dnorm(0.2, 1),
    sigma_school ~ dexp(1),
    phi ~ dexp(1)
  ),
  data = data3,
  chains = 4,
  cores = 4,
  log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
```

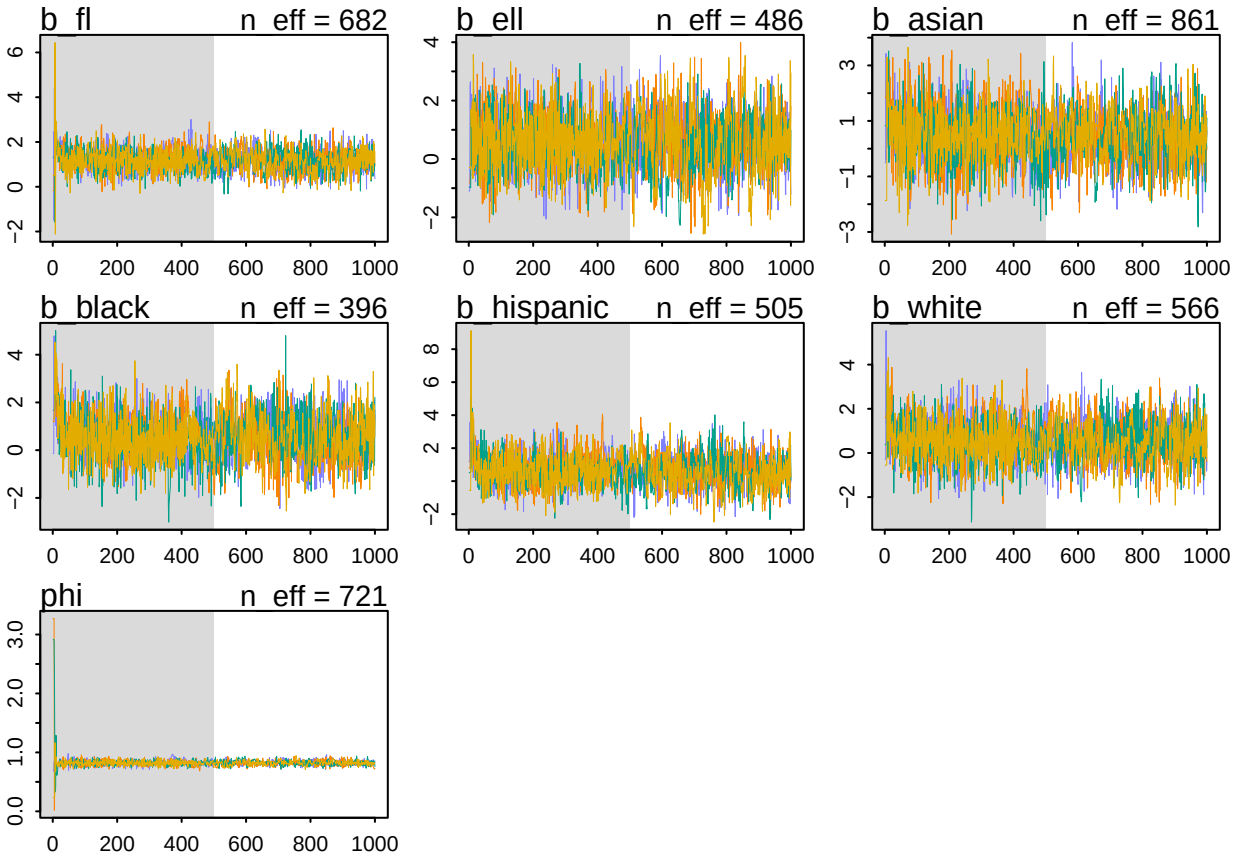
```
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 33.0 seconds.
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 34.1 seconds.
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 35.3 seconds.
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 47.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 37.5 seconds.
## Total execution time: 47.5 seconds.
```

```
precis(m_informative_1)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.

##                   mean         sd       5.5%     94.5%     n_eff    Rhat4
## b_fl         1.1545151 0.48625553  0.3428932 1.9522166 682.49355 1.004043
## b_ell        0.5480875 1.02115360 -1.1196991 2.1447375 486.41284 1.007179
## b_asian      0.4074774 0.92920010 -1.1021802 1.8745272 861.28246 1.008117
## b_black      0.4977277 1.00375069 -1.0653165 2.1118950 395.51448 1.010857
## b_hispanic   0.5783428 0.96957190 -0.9605561 2.1213169 504.96849 1.006474
## b_white      0.5295438 0.94296558 -0.9671003 2.0206066 566.43342 1.000379
## sigma_school 0.5718120 0.27202580  0.2364055 1.1278459  16.74734 1.183772
## phi          0.8216252 0.04007169  0.7571331 0.8876007 720.62077 1.004240
```

```
traceplot(m_informative_1, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white", "phi
```



Unfortunately, the model does not converge.

Model 2.1.2: Attempting to Fix Convergence

Again, we run into convergence issues with the initial nested model specification. I will use the same trick as before: applying inv_logit to keep the bounds on mu between 0 and 1, since this was what caused the initial errors.

```
set.seed(1)
m_informative_12 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),

    # Applying logistic transformation with a safeguard for mu
    mu <- inv_logit(a_school[School] + a_borough[Borough] +
                b_fl*fl_percent + b_ell*ell_percent +
                b_asian*asian_per + b_black*black_per +
                b_hispanic*hispanic_per + b_white*white_per),

    a_school[School] ~ dnorm(a_borough[Borough], sigma_school),
    sigma_school ~ dexp(1),

    a_borough[Borough] ~ dnorm(1, 2),
    b_fl ~ dnorm(1, 0.5),
    b_ell ~ dnorm(0.5, 1),
    b_asian ~ dnorm(0.2, 1),
```

```
    b_black ~ dnorm(0.2, 1),
    b_hispanic ~ dnorm(0.2, 1.),
    b_white ~ dnorm(0.2, 1),
    phi ~ dexp(1)
  ),
  data = data3, chains = 4, cores = 4, log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2 finished in 32.4 seconds.
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
```

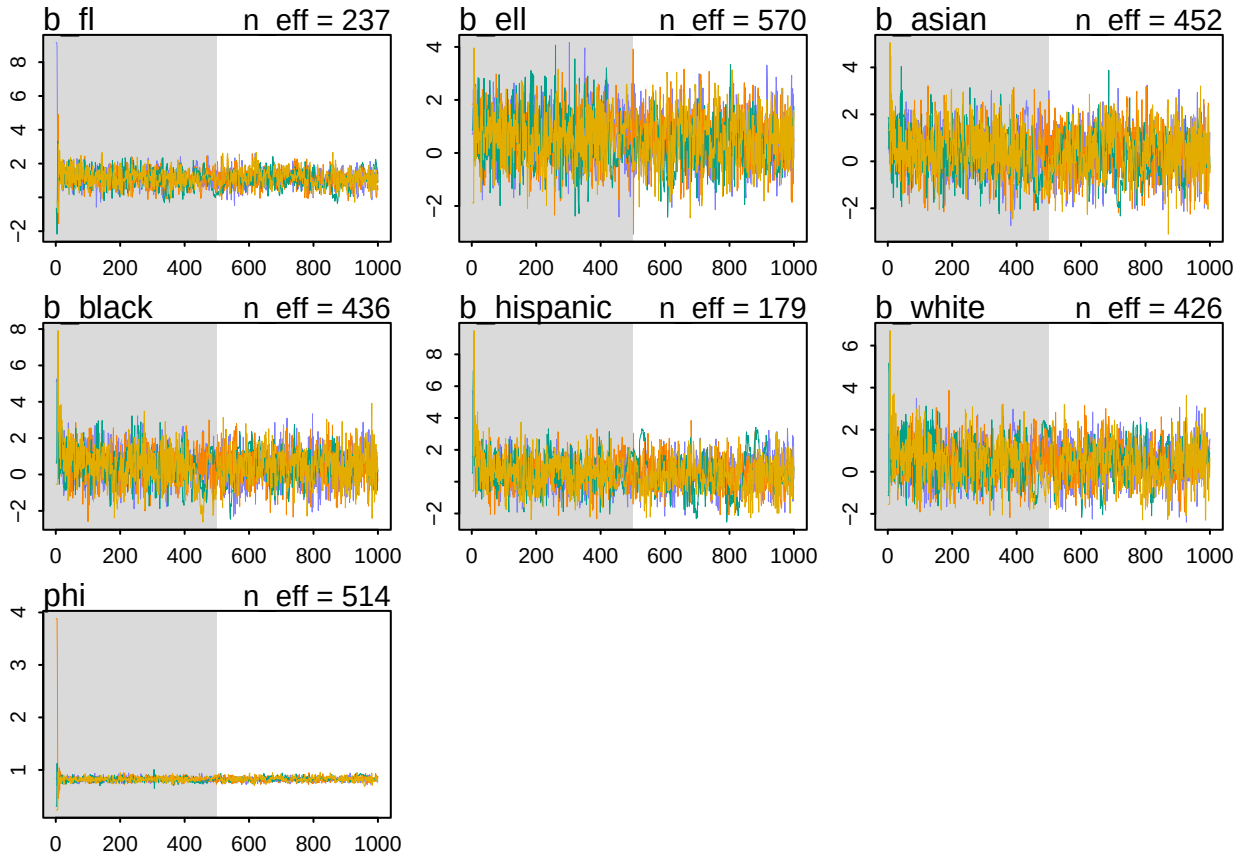
```
## Chain 1 finished in 37.3 seconds.
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 46.7 seconds.
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 52.9 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 42.3 seconds.
## Total execution time: 53.1 seconds.
```

```
precis(m_informative_12)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                   mean        sd        5.5%     94.5%      n_eff    Rhat4
## sigma_school 0.8254434 0.6099289  0.09489528 1.9498815   4.714329 1.909048
## b_fl         1.1083831 0.4959001  0.28807181 1.8709297 237.103409 1.016993
## b_ell        0.4585314 0.9599834 -1.03450990 1.9628207 570.061603 1.007886
## b_asian      0.3235302 0.9828213 -1.28286700 1.8807739 451.990230 1.001652
## b_black      0.4221749 0.9350690 -1.07580585 1.9015720 435.797631 1.000751
## b_hispanic   0.4766526 1.0632986 -1.28105190 2.2315044 179.451359 1.005310
## b_white      0.4215116 0.9501197 -1.11966890 1.9035336 426.226861 1.012828
## phi          0.8265385 0.0412554  0.76013573 0.8963835 513.578949 1.005493
```

```
traceplot(m_informative_12, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white", "pl
```



Again, the model's warnings are eliminated when applying the trick, but the model does not converge. We will attempt a non-nested hierarchical model with the informative models and determine if anything changes.

Model 2.2: Non-Nested Hierarchical Model, Informative Priors

Again, I will try another version of the Model 2.1: a non-nested hierarchical prior, again with weakly information priors. I will also build a non-nested hierarchical model to see if this would help with convergence issues.

```
set.seed(1)
m_informative_2 <- ulam(
  alist(
    Pass_Rate ~ dbeta(mu, phi),
    logit(mu) <- a_school[School] + a_borough[Borough] +
                 b_fl*fl_percent + b_ell*ell_percent +
                 b_asian*asian_per + b_black*black_per +
                 b_hispanic*hispanic_per + b_white*white_per,

    a_school[School] ~ dnorm(0, 1),
    a_borough[Borough] ~ dnorm(1, 2),
    b_fl ~ dnorm(1, 0.5),
    b_ell ~ dnorm(0.5, 1),
    b_asian ~ dnorm(0.2, 1),
    b_black ~ dnorm(0.2, 1),
    b_hispanic ~ dnorm(0.2, 1.),
```

```
    b_white ~ dnorm(0.2, 1),
    phi ~ dexp(1)
  ),
  data = data3, chains = 4, cores = 4, log_lik = TRUE
)
```

```
## Running MCMC with 4 parallel chains, with 1 thread(s) per chain...
##
## Chain 1 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 4 Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 4 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1 Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 4 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1 Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 4 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1 Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 4 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1 Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 4 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1 Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 4 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1 Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 4 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1 Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 4 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1 Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3 finished in 13.4 seconds.
## Chain 4 Iteration: 900 / 1000 [ 90%]  (Sampling)
```

```
## Chain 2 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2 finished in 13.7 seconds.
## Chain 4 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 4 finished in 14.5 seconds.
## Chain 1 Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1 finished in 14.7 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 14.1 seconds.
## Total execution time: 14.8 seconds.
```

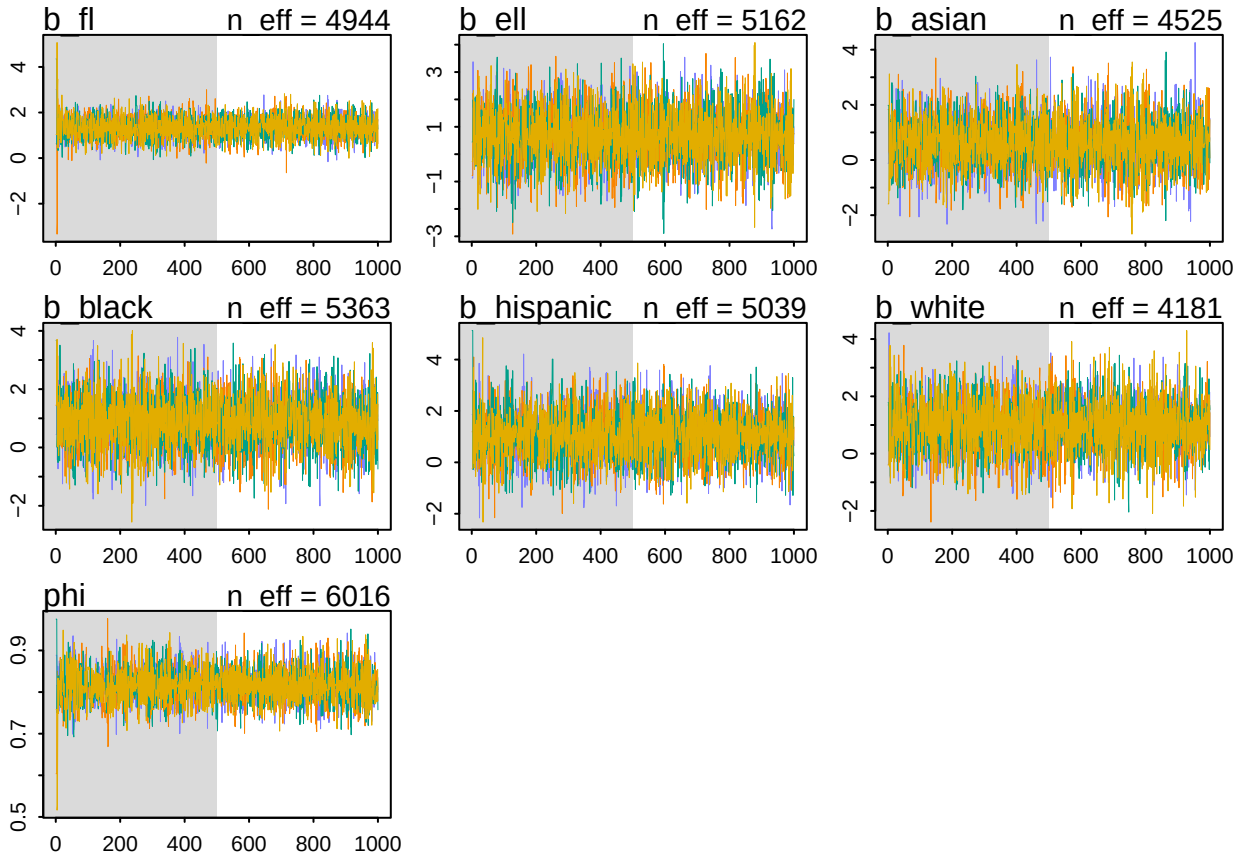
```
precis(m_informative_2)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```

```
##                 mean         sd       5.5%     94.5%    n_eff     Rhat4
## b_fl       1.3486039 0.47854466  0.5884430 2.1138978 4944.214 0.9983576
## b_ell      0.6874410 0.99872527 -0.9035058 2.2675113 5161.736 0.9986200
## b_asian    0.6194216 0.96085473 -0.9110869 2.1599498 4525.026 0.9993835
## b_black    0.8476084 0.91049242 -0.6205722 2.2740704 5362.755 0.9994560
## b_hispanic 1.0547236 0.91713279 -0.4883769 2.5552889 5038.798 0.9986528
## b_white    0.9381828 0.90919180 -0.4567469 2.3827854 4181.157 0.9987063
## phi        0.8158668 0.04014517  0.7530206 0.8797805 6015.556 0.9982250
```

Again, the Rhat values for b_fl, b_ell, b_asian, b_black, b_white, and phi are all 1. This seems to indicate that the model_uninformative_2 properly converged. Let's also check the traceplot to ensure that this is the case.

```
traceplot(m_informative_2, pars = c("b_fl", "b_ell", "b_asian", "b_black", "b_hispanic", "b_white", "phi
```



They do seem to be "fuzzy," normal-looking traceplot with no clear abnormalities within the traceplot. Thus, we can say that the m_informative_2, the model with informative priors with a non-hierarchical prior set-up, likely converged.

V. Conclusions

i. Comparisons

I will compare all 6 models.

```
compare(m_uninformative_1, m_uninformative_12, m_uninformative_2, m_informative_1, m_informative_12, m_
```

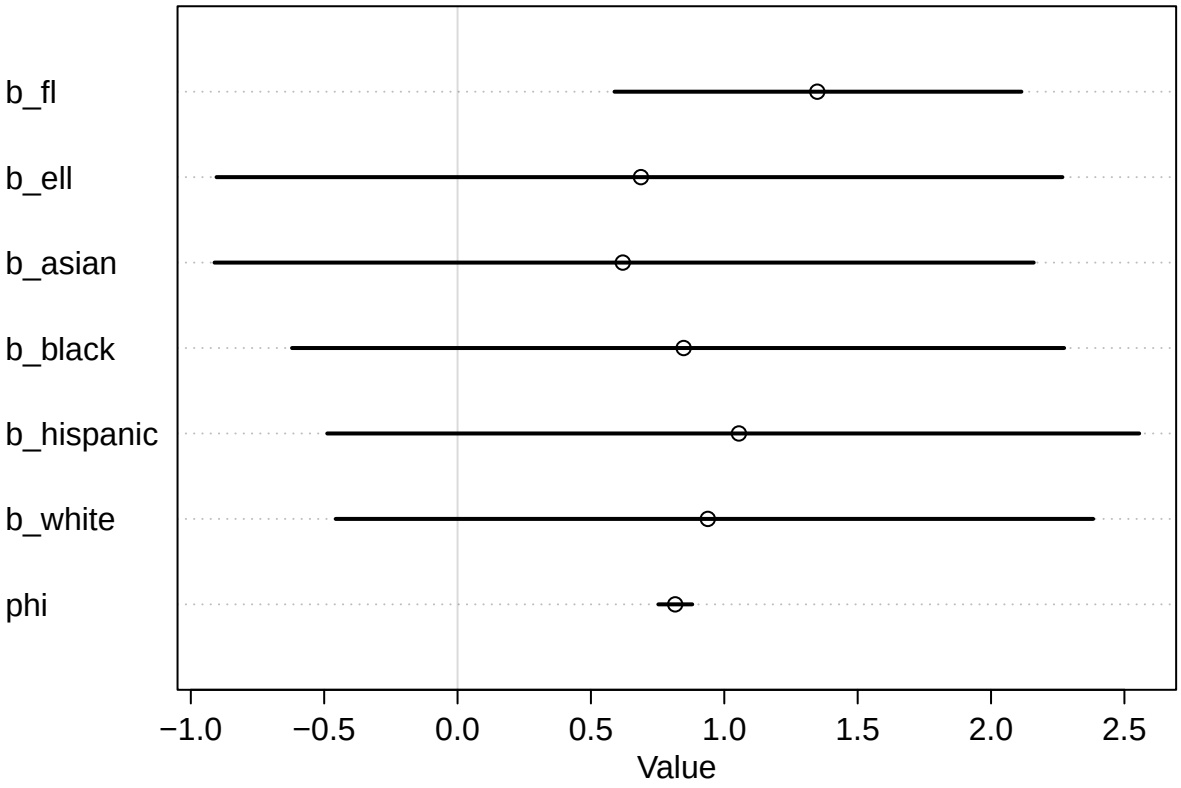
```
##                          WAIC       SE    dWAIC       dSE     pWAIC     weight
## m_informative_12   -12.195100 7.987372 0.000000        NA 1.0571512 0.48252161
## m_informative_1    -11.643735 8.196465 0.551365 0.2226019 0.9635173 0.36625990
## m_informative_2     -8.449423 8.335763 3.745677 0.4154976 1.1203398 0.07415720
## m_uninformative_1   -7.292220 8.235035 4.902880 0.3876731 1.1739158 0.04157860
## m_uninformative_12  -6.769571 8.325188 5.425529 0.4614668 1.1505040 0.03201681
## m_uninformative_2   -2.322976 8.346729 9.872124 0.6430894 1.4548051 0.00346587
```

Out of the six models, the model with the uninformative priors and the nesting structure has the lowest WAIC. This would suggest that model_uninformative_12 fits the data best. However, we should keep in mind that only the two non-nested model properly converged. Out of those two converged models, m_informative_2 performed the best. Thus, we should keep this as our "final model."

ii. Visualizations

```
model_summary <- plot(m_informative_2, depth=1)
```

```
## 403 vector or matrix parameters hidden. Use depth=2 to show them.
```



iii. Summary and Implications

In summary, we examined the effect of various sociodemographic factors on NY State Math Test results. We constrained the analysis to 8th graders in 2009 for practical purposes. Within those parameters, we used a Beta regression model

We can look at the overall distribution of our variables and summary statistics to see if the final model's results are in line with what we'd expect based on the distribution.

```
summary(data3)
```

```
##      School         District        Borough        Pass_Rate
##  Min.   :  1.0   Min.   : 1.00   Min.   :1.000   Min.   :0.1150
##  1st Qu.:100.2   1st Qu.: 8.00   1st Qu.:1.000   1st Qu.:0.4475
##  Median :199.5   Median :15.00   Median :2.000   Median :0.5980
##  Mean   :199.5   Mean   :15.37   Mean   :2.663   Mean   :0.5999
##  3rd Qu.:298.8   3rd Qu.:23.00   3rd Qu.:4.000   3rd Qu.:0.7600
##  Max.   :398.0   Max.   :32.00   Max.   :5.000   Max.   :0.9999
##    fl_percent      ell_percent       asian_per         black_per
##  Min.   :0.0990   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
```

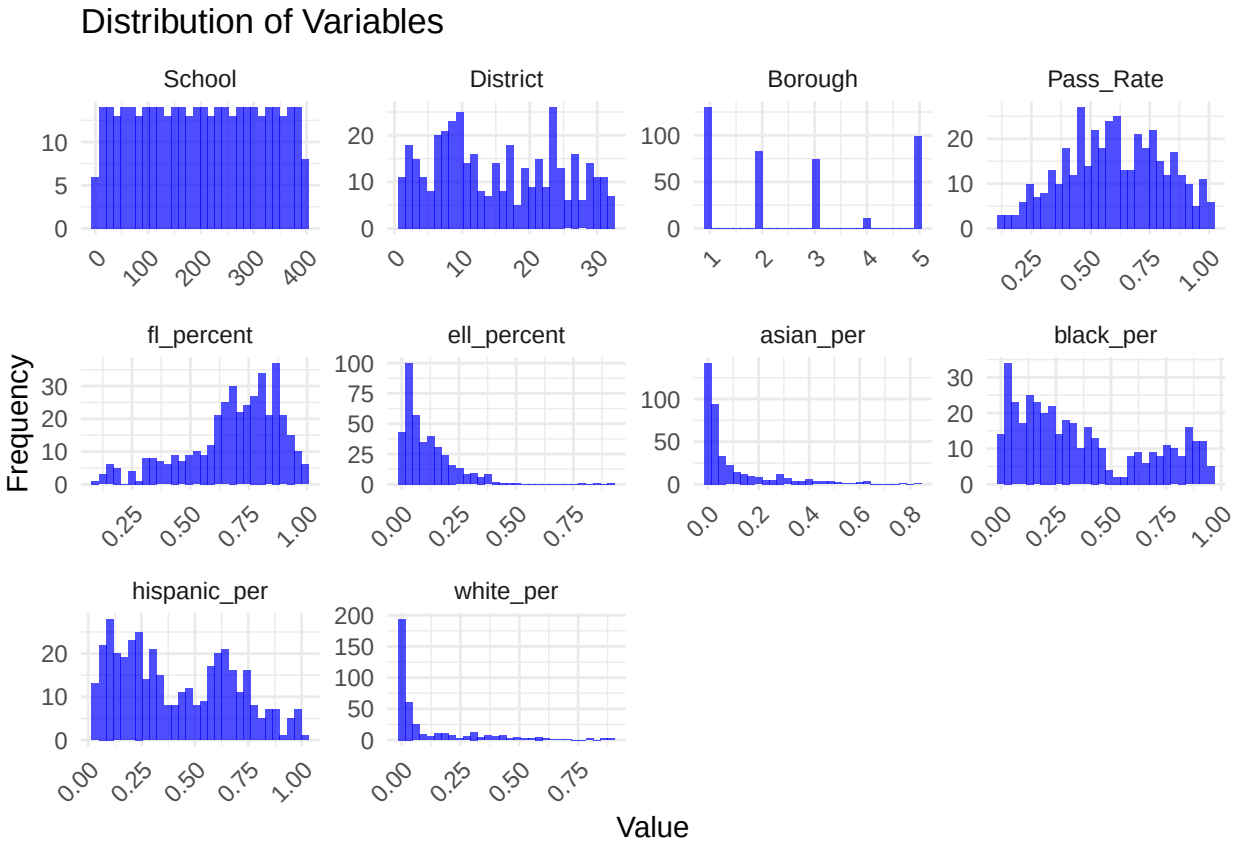
```
##  1st Qu.:0.5960   1st Qu.:0.0360   1st Qu.:0.01000   1st Qu.:0.1330
##  Median :0.7215   Median :0.0775   Median :0.02700   Median :0.2935
##  Mean   :0.6839   Mean   :0.1172   Mean   :0.09496   Mean   :0.3732
##  3rd Qu.:0.8357   3rd Qu.:0.1638   3rd Qu.:0.10900   3rd Qu.:0.6368
##  Max.   :0.9970   Max.   :0.9150   Max.   :0.82900   Max.   :0.9520
##   hispanic_per      white_per
##  Min.   :0.0180   Min.   :0.0000
##  1st Qu.:0.1805   1st Qu.:0.0070
##  Median :0.3490   Median :0.0160
##  Mean   :0.4125   Mean   :0.1119
##  3rd Qu.:0.6375   3rd Qu.:0.1547
##  Max.   :1.0000   Max.   :0.8910
```

Let's consider the distributions of the response variable and some of our predictors. Since all of them are standardized between 0 and 1 as percentages, we can visualize their distributions accordingly.

```
df_melted <- reshape2::melt(data3)
```

```
## No id variables; using all as measure variables
```

```
ggplot(df_melted, aes(x = value)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.7) +
  facet_wrap(~ variable, scales = "free") +
  theme_minimal() +
  labs(title = "Distribution of Variables", x = "Value", y = "Frequency") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



It seems that some of the predictors quite widely in their distributions. For example, looking at the third quantile racial composition predictors, it seems like Asian and white students seem to have lower enrollment rates at the schools within the dataset than Black and Hispanic students. Additionally, the percent of students eligible for free lunch is also skewed toward larger percentages, while the opposite is true for the proportion of English-language learners. Finally, inspecting the response variable, we see that pass_rates are better than chance, as the median pass_rate is around 0.6, and the distribution also seems to be bimodal.

To summarize the models that were created, there were six in total: one for each different combinations of model structure and prior choice (uninformative vs. weakly informative and nested vs. non-nested). Additionally, I created another two models for the nesting because the initial ones did not converge, as demonstrated by high Rhat values, low n_eff values, and irregular traceplots for the sigma_school parameter. However, the additional transformation to try to keep mu values between 0 and 1 did not resolve the convergence issue with sigma_school. Only upon creating a non-nested model, were the models able to converge.

This suggests to me that the nested model's complexity, with multiple levels of hierarchy, may have introduced challenges in estimating the model parameters effectively. This could be due to over-parameterization or correlations within the data that the model struggled to capture. Another hypothesis would be that it's possible that the variation within schools did not significantly differ across boroughs: if schools don't exhibit clear grouping patterns within boroughs, the nested model may not be the most appropriate representation.

While the improved convergence and performance of the non-nested model did not align with my initial hypothesis, the data might inherently have high variability that the nested models could not adequately capture. As I mentioned before, even if schools are logically nested within boroughs, the actual data might not reflect strong hierarchical patterns. For example, the variability within schools might not differ significantly across different boroughs, or the borough-level effects might be minimal compared to the school-level effects. This could lead to difficulties in estimating higher-level (borough) parameters accurately.

Turning to the "final model," the one that fit the data best AND converged was m_informative_2. This is the non-nested model with priors informed by previous literature. Interestingly, all models with informed priors outperformed all models with flat priors in terms of WAIC.

For the final model, the only significant sociodemographic predictor that was significantly different from zero, was b_fl. B_fl is the coefficient for the percentage of students within a school who are eligible for free lunch.

```
post = extract.samples(m_informative_2)
b_fl_samples <- post$b_fl
HPDI(b_fl_samples)
```

```
##    |0.89    0.89|
## 0.627485 2.150040
```

This aligns with the previous literature, which found the economic indicators were the primary sociodemographic factor involved in determining perform on academic assessments. The findings indicate that race and status as an English-language learner, the other school-level features accounted for, were not significant predictors of pass rates for the NY State Math Test. Even when accounting for variations at the borough level, the results suggest that students' eligibility for free lunch (a direct proxy for socioeconomic status) was the most important factor of those tested within this project. While keeping in mind that the data is limited to 2008, the broader implications of these findings are that the state should likely devote more attention toward bridging gaps in income inequity if they aim to improve eighth graders' performance on the NY State Math Exam.