Bayesian Multivariate Regression on Income of University Alumni in the U.S.

Niko Miller, Akseli Manninen, Santeri Löppönen

Contents

1. Project Introduction **2**

2. Data Description and Analysis Problem **2**

3. Feature Selection **2**

3. Model Descriptions **9**

4. Choice of Priors **10**

5. Stan Code **11**

6. Stan Specifications **18**

7. Convergence Diagnostics **18**

8. Posterior Predictive Checks and Model Comparison **20**

9. Predictive Performance Assessment **24**

10. Prior Sensitivity Analysis **34**

11. Discussion of Issues and Potential Improvements **34**

12. Conclusions **34**

13. Self-reflection **35**

14. References **36**

15. Appendix **36**

1. Project Introduction

Studying the relationship between income and education has been the focus of many studies. The studies have concluded that a strong connection exists between higher education and income (Card, 1999). In general, individuals with stronger education are more likely to be employed and earn a big salary compared to less educated people (Card, 1999). For that reason, education is described as an investment in human capital (Wolla & Sullivan, 2017).

This study examines this phenomenon from the perspective of people that have acquired their education from colleges in the United States. As the connection between education and income has been shown in the existing literature, this study strives to further examine the associations between college related features and income level years after graduation. This project is not limited to only considering educational aspects but expands it to family backgrounds.

In this study, a Bayesian approach is taken to observe the bond between the educational and family related features and earnings in a multivariate linear regression setting. It is in our interest to find out how accurately the selected features can predict future income for the students. Aim of our project is to find a way to predict average income after studying at a university for any non-specialized university in the United States. We build three different multivariate models – separate, pooled and hierarchical – to predict income after school.

As this study is conducted in a university environment by university students and presented mainly to other students and faculty, the findings of this study could be especially meaningful for the members of the group and peers on the course.

2. Data Description and Analysis Problem

The used dataset is the most recent institutional-level college scorecard data from the US Department of Education[1]. The institutional-level dataset contains aggregate data for each educational institution and includes data on institutional characteristics, enrollment, student aid, costs and student outcomes. The dataset has 6681 observations on 2989 variables.

This dataset was chosen because it seemed to provide information that could answer interesting education-related questions in the project, and also because the dataset has a large number of observations and a large number of variables. The large amount of variables permits for a lot of flexibility when it comes to modelling with the data and also having enough data is important in order to be able to make valid inferences.

After investigating the dataset, the most intriguing analysis problem was to study how college related factors affect the median earnings of alumni 10 years after entry, which is our dependent variable:

Name	Data type	Description
MD_EARN_WNE_P10	double	Median earnings of students 10 years after entry

To answer the analysis problem, a subset of college related factors is extracted and generated from the dataset and after that weigths for each factor are calculated based on three models that are introduced later in this document.

In the next section, we outline how the independent variables, hereafter features, were selected

3. Feature Selection

The initial data set has almost 3000 features, which is a large number compared to the total number of observations of 6681. Moreover, there are many missing values for certain variables in the dataset and not all variables are interesting when trying to predict future earnings. For these reasons, there was a clear need to prune down features.

[1] Available at: https://collegescorecard.ed.gov/data

We conducted feature selection in three phases: in the first phase, a subset of features was selected based on features commonly used in previous studies that have examined the relationship between educational factors and earnings. In addition, we tried to come up with additional interesting features that could predict future earnings. In the second phase, we assessed the relationship of the features with our dependent variable, and accounted for any multicollinearity arising from mutually correlated features. Furthermore, we visualized categorical variables and engineered new features to be added to the model. In the last phase, we utilized stepwise regression in streamlining our model to include only the most significant features in terms of the Akaike Information Criterion (AIC).

Phase 1 - Feature selection based on literature

The initial set of features chosen based on past literature and general intuition is outlined in the table below. (Dominic et. Al, 1996), (David, Krueger, 1996)

	Name	Data type	Description
1	SATVRMID	integer	Midpoint of SAT scores at the institution (critical reading)
2	SATMTMID	integer	Midpoint of SAT scores at the institution (math)
3	SATWRMID	integer	Midpoint of SAT scores at the institution (writing)
4	MD_FAMINC	double	Median family income
5	AGE_ENTRY	double	Average age of entry
6	FEMALE	double	Share of female students
7	FIRST_GEN	double	Share of first-generation students
8	PCT_WHITE	double	Percent of the population from students' zip codes that is White
9	DEBT_MDN_SUPP	integer	Median debt, suppressed for n=30
10	C150_4	double	Completion rate for first-time, full-tim students
11	COSTT4_A	integer	Average cost of attendance (academic year institutions)
12	POVERTY_RATE	double	Poverty rate
13	UNEMP_RATE	double	Unemployment rate
14	MARRIED	double	Share of married students
15	VETERAN	double	Share of veteran students
16	LOCALE	categorical	Locale of institution
17	CCBASIC	categorical	Carnegie Classification – basic
18	CONTROL	categorical	Control of institution

SAT scores were aggregated into one composite variable called SAT_ALL, because SAT components were highly correlated and could easily be viewed as one entity. However, writing SAT scores had too few observations due to discontinued tracking. Therefore, SAT_ALL was formed by summing the math and critical thinking SAT scores. After this, we ended up with a total of 793 observations for the feature subset.

Descriptive statistics for the resulting numerical variables can be found from the table below. Min is the minimum, 1st Qu. is the 1st quartile, Median is the median, Mean is the arithmetic mean, 3rd Qu. is the 3rd quartile, Max is the maximum and St.dev is the sample standard deviation.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max	St.dev
MD_EARN_WNE_P10	24209.00	43093.00	49240.50	51246.87	56516.75	103246.00	11587.58
MD_FAMINC	10702.00	35386.12	44858.50	48915.83	61402.38	121852.50	18732.71
AGE_ENTRY	19.55	20.92	22.04	22.53	23.51	33.82	2.18
FEMALE	0.12	0.54	0.59	0.59	0.64	0.98	0.10
FIRST_GEN	0.10	0.27	0.34	0.33	0.39	0.62	0.09
PCT_WHITE	24.24	71.45	80.24	77.62	87.62	95.96	13.09
DEBT_MDN_SUPP	3688.00	13971.00	16000.00	16190.37	19000.00	26800.00	3595.32
C150_4	0.09	0.46	0.56	0.57	0.68	0.98	0.16
COSTT4_A	11704.00	23250.25	31904.00	35719.42	45932.00	79750.00	15267.87
POVERTY_RATE	3.58	6.44	7.74	8.68	9.87	45.73	3.81
UNEMP_RATE	2.12	3.01	3.34	3.48	3.78	7.92	0.74
MARRIED	0.01	0.03	0.06	0.08	0.10	0.38	0.07
SAT_ALL	395.50	513.25	545.00	553.57	584.62	760.00	59.93

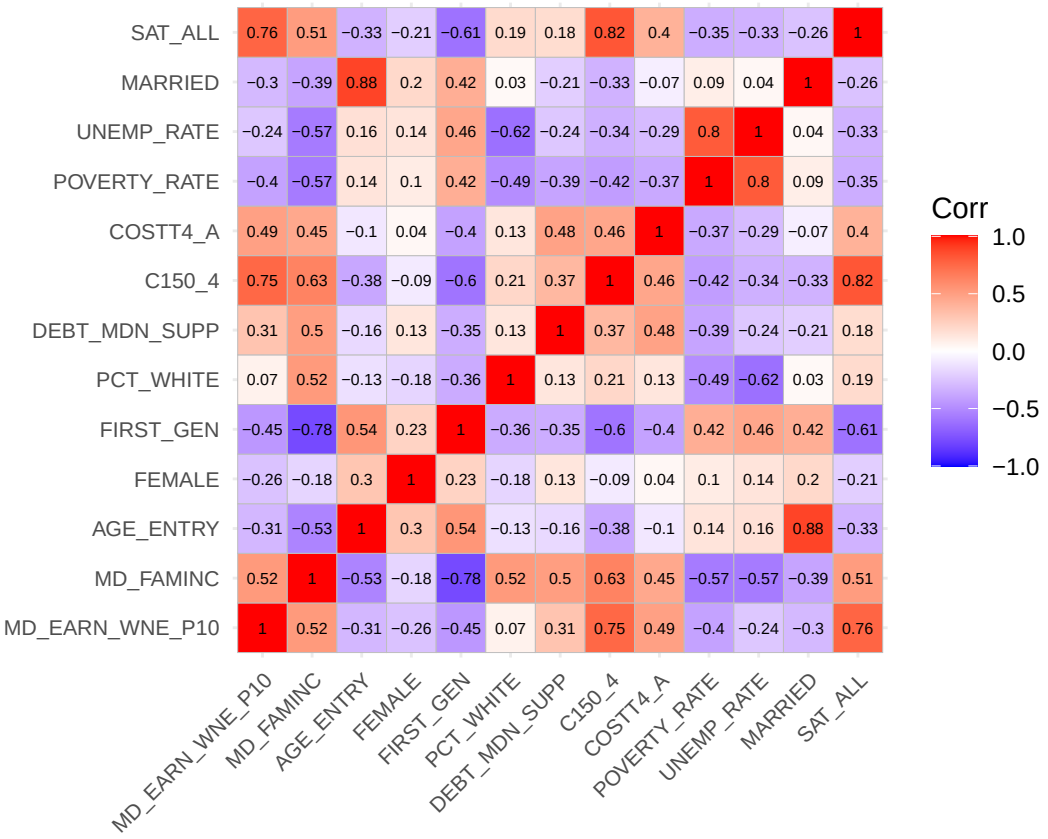
The table shows that our dependent variable - median earnings 10 years after entry - range from \$24.2k to \$103.2k. Overall one can see that there is great variation in almost all features as well. For example, age of entry varies from 19.6 to 33.8, average composite SAT scores range from 395 to 760, and cost of attendance varies from \$11.7k to \$79.8k, to name a few interesting details.

Phase 2 - Feature Selection with Correlation and Linear Association

In the second phase of feature selection, a subset of features was selected from the 18 variables resulting from the first phase. We examined the correlations between the dependent variable and the features as well as between-features correlations. Moreover, we examined the linear relationships between the features and the dependent variable. Lastly, we visually examined categorical variables and engineered new features that we believed could have predictive ability over the dependent variable.

Numerical Variables

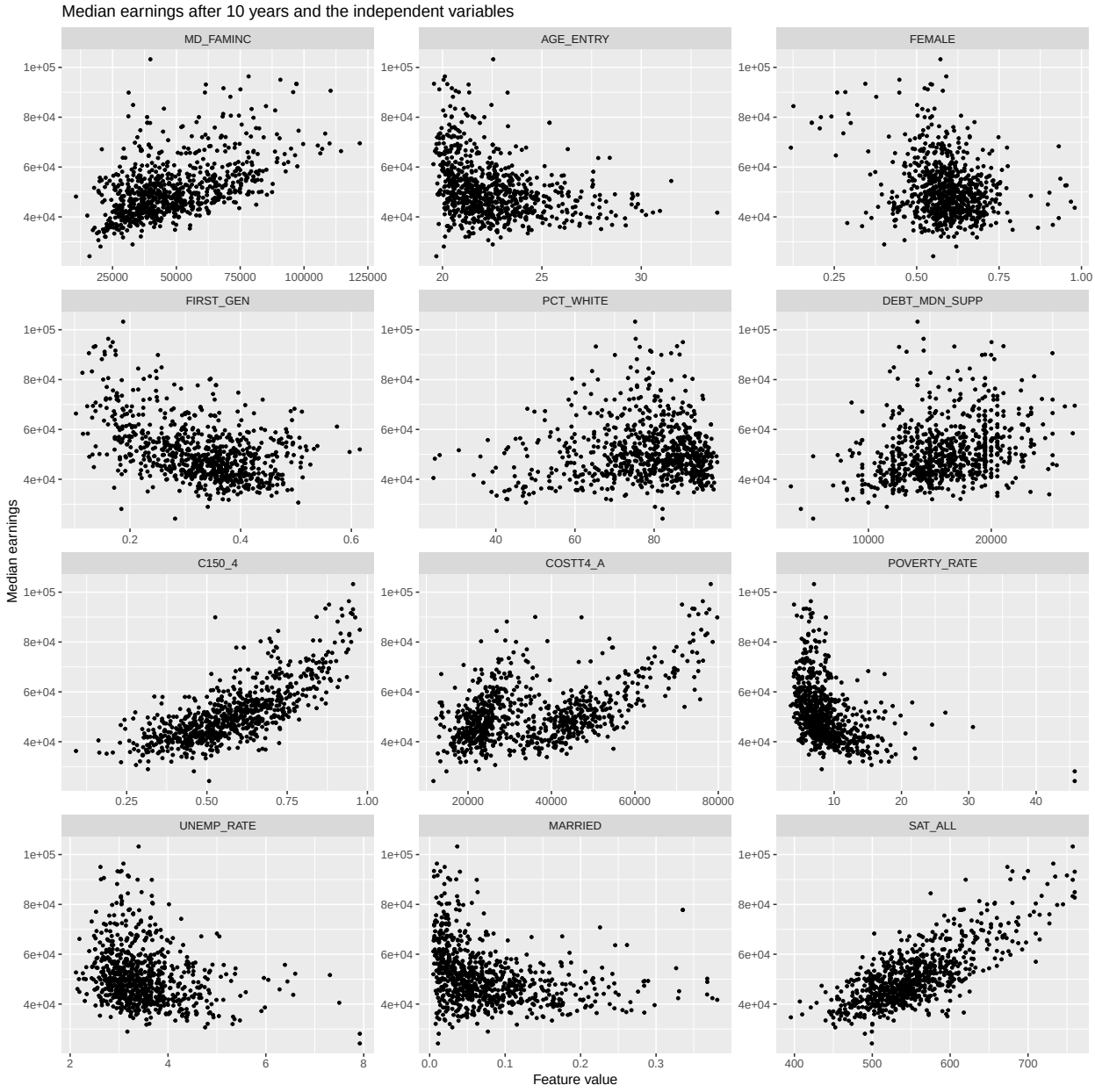
Correlation Analysis The figure below shows the (Pearson's) correlation matrix for the dependent variable and the features.




```
## Saving 6.5 x 4.5 in image
```

The correlation matrix shows on the bottom row that SAT scores, cost of attendance, and family income are the most positively correlated features with our dependent variable. The respective correlations are 0.76, 0.75, and 0.52. Most negatively correlated features are the percentage of first generation students, poverty rate, and average age of entry with respective correlations of -0.45, -0.4, and -0.31. These are interesting observations and make intuitive sense as well.

Bivariate Plotting The panel of figures below show scatter plots between the dependent variable and all numerical features. The x-axis represents the value of the feature and the y-axis represents the value of the dependent variable.




```
## Saving 12 x 12 in image
```

The figures show that many of the features exhibit a linear like relationship with the dependent variable. Especially, SAT scores (SAT_ALL) show a clear linear association. Other features that show a linear trend are e.g., family income (MD_FAMINC), completion rate (C140_4), and the percentage of first generation students (FIRST_GEN). There are hints of some non-linear relationships as well, e.g., poverty rate appears to have a convex non linear relationship with the dependent variable. Cost of attendance (COSTT4_A), on the other hand, suggests that there could be two groups, perhaps cheaper public school and private school.

Summary of Numerical Feature Selection Based on the analysis presented thus far, we selected the following numerical variables: SAT_ALL (median sum of math and critical thinking SAT scores), MD_FAMINIC (median family income of the student), AGE_ENTRY (median age of starting at the college), COSTT4_A (median cost of college), and POVERTY_RATE (poverty rate in the area the college is located).

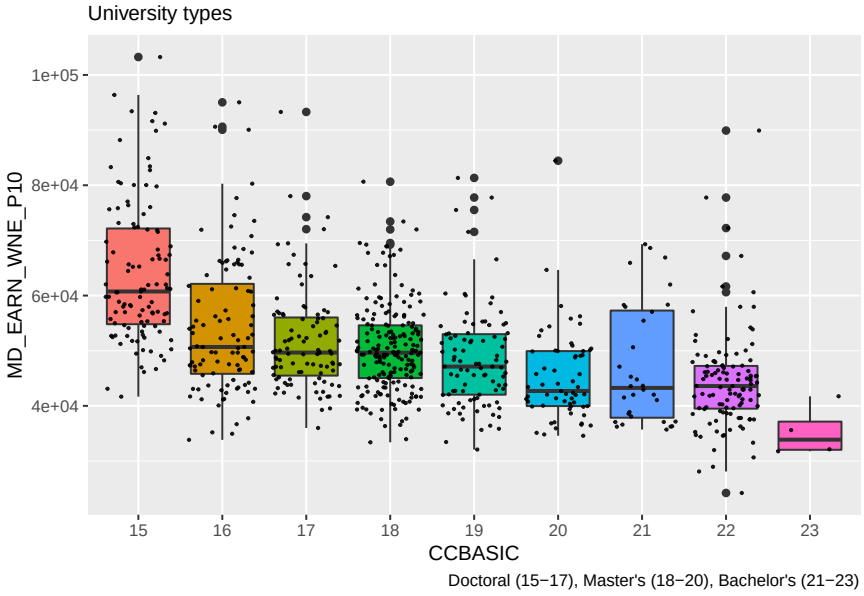
After including the obvious features, such as SAT scores, we included features with the following reasoning: if

the correlation between a feature and the dependent variable was low and there was no observable dependency between the two in the scatter plot, the feature was excluded. Furthermore, to avoid multicollinearity, we removed features that were highly correlated with other independent variables, especially if they were not clearly correlated with the dependent variable and we couldn't form a believable hypothesis for the mechanism through which that variable affected income after college.

Categorical Variables

Box Plots The categorical variables were visualized with box plots to assess if income after college differs among the categories.

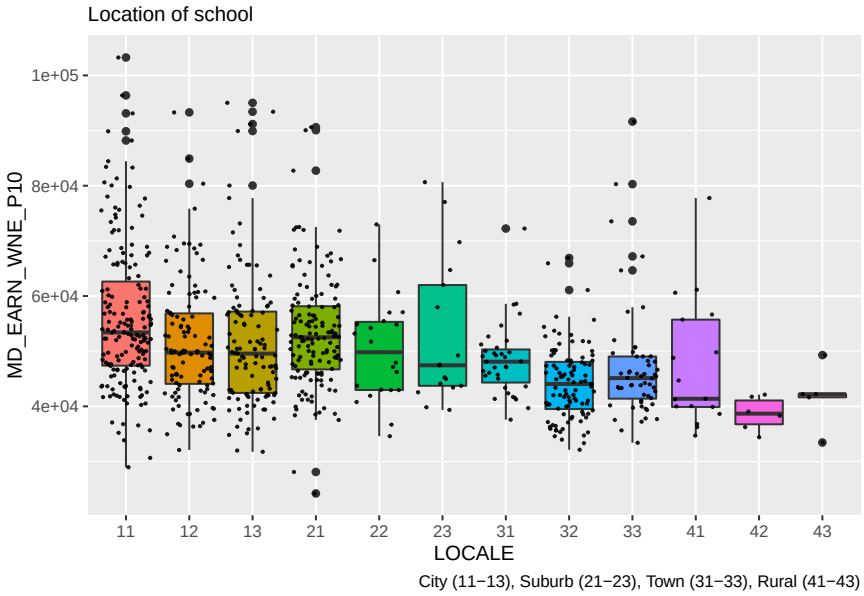
The figure below shows the median earnings for different university types. CCBASIC stands for the Carnegie Classification for the university.




```
## Saving 6.5 x 4.5 in image
```

Based on the plot, CCBASIC was divided into two dummy variables: MASTER and DOCTORAL. These variables represent if the college is classified as Master's college and university or Doctoral's college and university. If a college does not belong to either of those, it is a Bachelor's college and university. Other special focus colleges and universities were discarded from the dataset to keep the model simpler and to avoid unnecessary outliers due to unconventional nature of some very specialized colleges. Our model does not attempt to provide accurate predictions for specialized colleges.

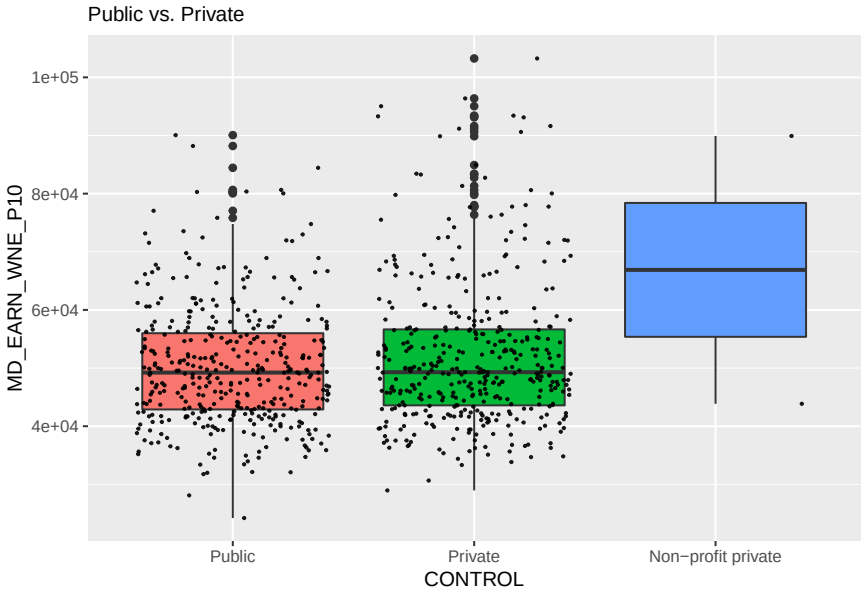
The figure below shows the median earnings for different university locations. LOCALE specifies whether the university locates in a metropolis, city, suburb, town, or a rural area.




```
## Saving 6.5 x 4.5 in image
```

Based on figure, we engineered a new dummy variable URBAN, which takes the value 1 if the university LOCALE is not classified as Rural.

The categorical variable CONTROL had three classes: public, for-profit private and nonprofit private. The figure below shows the median earnings depending on the control status.




```
## Saving 6.5 x 4.5 in image
```

Based on the figure, CONTROL was modified into a dummy variable PRIVATE, which takes the value 1 if the school is privately controlled (either for profit or non profit).

The selected categorical variables were: URBAN, DOCTORAL, MASTER, PRIVATE.

In the third and final phase, all remaining variables were used with stepwise regression to test which subset of features perform the best, whether the received coefficients are reasonable, and if there are signs of overfitting.

The stepwise regression suggested using all of the variables, except AGE ENTRY and DOCTORAL. The output of the stepwise regression can be found in the Appendix. Because stepwise regression suggested leaving out DOCTORAL, for the sake of consistency we decided to also leave out MASTER as it was a the middle level in our institutional classifications and wouldn't have made much sense on its own. To test whether AGE ENTRY would be included by stepwise after removing MASTER and DOCTORAL, stepwise was ran again. Even then, AGE ENTRY was discarded from the model and the other variables remained the same.

The final features are listed in the table below:

	Name	Data type	Description
1	SAT_ALL	float	Midpoint of SAT scores at the institution (critical reading ,math)
2	MD_FAMINC	float	Median family income
3	COSTT4_A	float	Average cost of attendance (academic year institutions)
4	POVERTY_RATE	float	Poverty rate
5	URBAN	binary	Urban or rural area
6	PRIVATE	binary	Private or public school

3. Model Descriptions

Description of the separate model

In the separate model, posteriors for the parameters are constructed. In the context of the project, the separate model considers all regions independent from each other, meaning that each region have individual parameters (mu, sigma).

Mathematical description:

$y_{ij}|\mu_j, \sigma_j \sim \mathcal{N}(\mu_j, \sigma_j^2)$

where $\mu_j = \alpha_j + \beta_{\mathbf{j}}\mathbf{X}$

The parameters of the parameter vector are given in the section 4 with their priors.

Description of the pooled model

As in the separate model, the pooled model constructs posteriors for the parameters. However, the regions are considered as one entity meaning that all regions share the same distribution and parameter values.

Mathematical description:

$y_i|\mu, \sigma \sim \mathcal{N}(\mu, \sigma^2)$

where $\mu = \alpha + \beta\mathbf{X}$

The parameters of the parameter vector are given in the section 4 with their priors.

Description of the hierarchical model

Contrary to the other two models, in the hierarchical model posteriors are constructed for the prior parameters. With the hierarchical model, the regions are considered as individual but similar. In the context of the project, the regions share same sigma and the parameters forming mu are similarly distributed (sharing the hyperparameters).

Mathematical description:

$y_{ij}|\mu_j \sim \mathcal{N}(\mu_j, \sigma^2)$

where $\mu_j|\mu_P, \sigma_P^2 \sim \mathcal{N}(\mu_P, \sigma_P^2)$

The hyperparameters and parameters are given in the section 4 with their priors.

4. Choice of Priors

We use weakly informative priors, as we do not posses enough information about the dependency between the independent variables and the dependent variable. When selecting the priors, proper distribution type was considered for each variable. The prior standard deviations were selected based on the magnitude of absolute values of the variables and what kind of impact they could have for income, with loose enough estimates to avoid limiting the values too much. Alternative approach could have been to standardize all variables, which would have reduced the need to think about magnitudes of absolute values.

SAT_ALL ~ Normal(43, 500)

Justification: We agreed that it was somewhat reasonable to expect scores in the SAT exam to be associated with higher income. Due to our approach of using weakly informative priors, we set a high standard deviation, but set a positive mean due to expected positive association.

The mean is calculated with the following formula: Median individual income in the United States / Average SAT Score (math and writing).

The median individual income in the US was approximately 31 000 in 2020 (Data Commons, 2020). The average SAT score in the US in 2020 were 523 in Math and 582 in Evidence-Based Reading and Writing (Number2, 2020).

mu = 31 000 / (523 + 528/2) = 43.2

MD_FAMINIC ~ Normal(0, 100)

Justification: Weakly informative prior is again selected as we don't posses enough information on the dependency. The absolute values are in general high, (for example compared to AGE_OF_ENTRY) and thus the standard deviation is set lower for this variable. However, there could be cases where MD_FAMINIC is low, due to for example unemployment, so the standard deviation is still set considerably high.

COSTT4_A ~ Normal(0, 500)

Justification: Weakly informative prior is selected as we don't posses enough information on the dependency. The average cost per academic year is likely to take lower values than median family income, but higher values than average age of entry and thus the standard deviation is set between the standard deviations of those.

POVERTY_RATE ~ Normal(0, 2500)

Justification: The possible values are between 0 and 100. There could be situations were poverty rate in an area is really low, for instance 0.5%. For that reason, the standard deviation in the prior is set high to enable possibly high weight for a small value.

URBAN ~ Normal(0, 2500)

Justification: Weakly informative prior was selected for URBAN. Although, urban areas could potentially have higher wages on average, in this project, we are considering earnings ten years after entry, which means that there might have been a lot of people moving from rural areas to urban areas after graduation and vice versa. For that reason, the prior is not set to be too informative. Because the values are always either 0 and 1, the standard deviation is set high.

PRIVATE ~ Normal(0, 2500)

Justification: For MASTER and PRIVATE weakly informative prior, is selected as we don't posses enough information on the dependency with the dependent variable. Because the values are always either 0 and 1, the standard deviation is set high.

Alpha ~ Normal(0, 10000) Sigma ~ Normal(10000, 10000)

Justification: For Alpha and Sigma, weekly informative priors were selected as we did not posses enough information.

Hyperpriors:

$Alpha_p$ ~ Normal(100, 100)

SAT_p ~ Normal(100, 100)

$MD\_FAMINIC$ ~ Normal(10, 100)

$COSTT4\_A_p$ ~ Normal(50, 500)

$POVERTY\_RATE_p$ ~ Normal(500, 500)

$URBAN_p$ ~ Normal(500, 500)

$PRIVATE_p$ ~ Normal(500, 500)

$Sigma$~ Normal(500, 500)

Justifications for the hyperpriors: When selecting hyperpriors it was considered that the constructions would be consistent in a sense that there would not be conflicting levels of informativeness.

5. Stan Code

Before fitting the models in Stan, we split our data set of 793 observations into a training set of 763 observations and a test set of 30 observations. The test set is used in assessing the predictive ability of our models in Section 9.

We used cmdstanr for our models. Source code for all three models can be found in in the Appendix.

Separate model

Summary of model fit for main parameters:

```
##              variable      mean    median      sd     mad        q5       q95
##  lp__                -63023.35 -63023.10    6.08    6.08 -63033.60 -63013.90
##  alpha[1]             -8599.92  -8532.26 3881.99 3722.23 -15063.98  -2377.00
##  alpha[2]            -25126.29 -25109.25 1697.45 1661.70 -27952.41 -22355.61
##  alpha[3]            -20235.88 -20247.65 2659.27 2697.07 -24499.21 -15810.14
##  alpha[4]              5151.70   5136.22 3140.40 3152.99    121.34  10277.07
##  alpha[5]             -5558.11  -5549.98 1743.31 1649.62  -8378.03  -2665.03
##  alpha[6]            -17296.75 -17297.15 3211.96 3183.66 -22546.38 -11977.47
##  alpha[7]            -16323.88 -16349.45 5246.52 5483.40 -24954.39  -7871.02
##  alpha[8]            -18092.03 -18120.10 3075.57 3096.63 -22980.20 -13075.72
##  alpha[9]               206.09    253.09 9657.90 9781.20 -15977.78  15574.76
##  beta_SAT_ALL[1]        121.07    121.02    7.16    7.24    109.38    132.83
##  beta_SAT_ALL[2]        109.98    110.07    3.54    3.55    104.05    115.60
##  beta_SAT_ALL[3]        109.80    109.89    4.63    4.75    102.31    117.30
##  beta_SAT_ALL[4]         55.59     55.53    5.93    5.99     45.75     65.30
##  beta_SAT_ALL[5]         76.88     76.99    3.10    2.94     71.79     81.83
##  beta_SAT_ALL[6]        106.93    107.06    6.29    6.24     96.39    116.71
##  beta_SAT_ALL[7]        122.30    122.41    8.91    8.84    107.32    136.74
##  beta_SAT_ALL[8]        105.71    105.52    5.43    5.30     96.87    114.90
##  beta_SAT_ALL[9]          3.13      1.19  171.66  174.34   -277.62    282.81
##  beta_MD_FAMINIC[1]       0.02      0.02    0.02    0.02     -0.02      0.06
##  beta_MD_FAMINIC[2]       0.08      0.08    0.01    0.01      0.07      0.10
##  beta_MD_FAMINIC[3]      -0.04     -0.04    0.02    0.01     -0.07     -0.02
##  beta_MD_FAMINIC[4]       0.13      0.13    0.02    0.02      0.10      0.16
##  beta_MD_FAMINIC[5]       0.09      0.09    0.01    0.01      0.07      0.11
##  beta_MD_FAMINIC[6]       0.04      0.04    0.03    0.03     -0.01      0.08
```

```
##  beta_MD_FAMINIC[7]        0.13      0.13    0.04    0.04      0.07      0.19
##  beta_MD_FAMINIC[8]        0.10      0.10    0.02    0.02      0.07      0.13
##  beta_MD_FAMINIC[9]        7.66      8.27   36.40   37.00    -51.36     67.15
##  beta_COSTT4_A[1]          0.05      0.05    0.06    0.06     -0.06      0.14
##  beta_COSTT4_A[2]          0.31      0.31    0.02    0.02      0.27      0.35
##  beta_COSTT4_A[3]          0.44      0.44    0.03    0.03      0.39      0.50
##  beta_COSTT4_A[4]          0.37      0.37    0.04    0.04      0.31      0.43
##  beta_COSTT4_A[5]          0.28      0.28    0.02    0.02      0.25      0.31
##  beta_COSTT4_A[6]          0.27      0.27    0.04    0.04      0.21      0.33
##  beta_COSTT4_A[7]         -0.04     -0.04    0.04    0.05     -0.12      0.03
##  beta_COSTT4_A[8]          0.30      0.30    0.03    0.03      0.25      0.35
##  beta_COSTT4_A[9]         -9.37    -10.21   50.98   51.94    -93.55     72.94
##  beta_POVERTY_RATE[1]   -957.02   -954.62  285.64  281.41  -1423.83   -466.89
##  beta_POVERTY_RATE[2]    308.66    308.10   58.15   56.80    212.84    405.21
##  beta_POVERTY_RATE[3]     81.65     86.55  123.94  117.85   -133.38    278.80
##  beta_POVERTY_RATE[4]   -338.67   -334.47  175.16  174.85   -629.01    -58.06
##  beta_POVERTY_RATE[5]   -237.88   -238.45   53.35   53.49   -324.93   -151.31
##  beta_POVERTY_RATE[6]     72.45     73.73   66.55   66.89    -38.21    180.41
##  beta_POVERTY_RATE[7]  -1173.55  -1172.16  209.74  203.47  -1504.26   -831.10
##  beta_POVERTY_RATE[8]    565.44    565.29  111.32  108.59    382.39    751.39
##  beta_POVERTY_RATE[9]    213.35    248.44 2179.03 2242.04  -3310.81   3784.96
##  beta_URBAN[1]          1250.94   1257.19  904.17  857.31   -263.74   2757.65
##  beta_URBAN[2]          4992.75   4997.60  488.96  483.56   4169.67   5804.37
##  beta_URBAN[3]           651.69    651.02  431.20  415.52    -56.16   1373.20
##  beta_URBAN[4]           192.16    193.87  504.60  485.86   -629.48   1038.93
##  beta_URBAN[5]          1983.92   1989.39  303.66  301.69   1489.58   2478.68
##  beta_URBAN[6]          -561.26   -576.39  525.23  512.68  -1371.08    315.99
##  beta_URBAN[7]          3468.86   3458.14  810.68  801.70   2163.59   4828.62
##  beta_URBAN[8]          -364.40   -375.93  621.15  591.28  -1387.78    695.39
##  beta_URBAN[9]           -69.02    -40.14 2519.12 2469.08  -4231.04   3921.94
##  beta_PRIVATE[1]       -1127.10  -1184.13 1877.77 1885.92  -4206.23   1995.82
##  beta_PRIVATE[2]       -6502.61  -6502.52  701.14  685.15  -7666.64  -5313.35
##  beta_PRIVATE[3]       -6949.69  -6937.43  799.96  786.16  -8304.20  -5656.89
##  beta_PRIVATE[4]       -7530.20  -7515.66 1013.79 1022.67  -9250.35  -5914.06
##  beta_PRIVATE[5]       -6060.34  -6058.84  447.25  436.83  -6797.57  -5328.46
##  beta_PRIVATE[6]       -6788.08  -6795.42  843.02  834.21  -8127.26  -5379.79
##  beta_PRIVATE[7]        3209.44   3176.57  966.82  948.23   1655.86   4854.19
##  beta_PRIVATE[8]       -8938.78  -8925.40  894.63  890.72 -10436.14  -7453.43
##  beta_PRIVATE[9]         144.82    113.95 2537.21 2513.99  -3982.07   4319.11
##  sigma[1]               5702.79   5699.48  139.74  138.51   5474.46   5932.49
##  sigma[2]               5699.35   5694.35  142.44  148.42   5475.49   5941.72
##  sigma[3]               5704.75   5699.27  146.93  146.78   5471.16   5959.50
##  sigma[4]               5687.02   5678.32  143.12  142.11   5460.51   5934.89
##  sigma[5]               5686.61   5681.70  142.35  142.65   5462.81   5933.66
##  sigma[6]               5693.11   5687.68  151.05  146.46   5452.92   5945.24
##  sigma[7]               5704.38   5699.18  143.08  144.75   5474.12   5940.95
##  sigma[8]               5697.26   5689.57  142.51  145.85   5476.03   5933.09
##  sigma[9]               5699.06   5697.49  150.48  157.73   5453.85   5953.81
##  rhat ess_bulk ess_tail
##  1.00      740     1691
##  1.01      187      413
##  1.02      174      530
##  1.03      179      415
##  1.01      265      494
```

```
##  1.02      125      283
##  1.01      224      330
##  1.02      178      372
##  1.04      122      189
##  1.02      416      838
##  1.01      188      373
##  1.03      130      311
##  1.03      175      346
##  1.01      271      548
##  1.03      122      265
##  1.01      199      251
##  1.01      178      420
##  1.04      155      263
##  1.06       95      162
##  1.01      252      617
##  1.01      350      773
##  1.03      185      581
##  1.01      413      985
##  1.01      342      682
##  1.01      354      720
##  1.02      217      324
##  1.00      425      727
##  1.06       94      147
##  1.01      197      372
##  1.03      142      305
##  1.01      238      473
##  1.02      325      682
##  1.01      275      878
##  1.01      239      567
##  1.02      225      752
##  1.02      219      432
##  1.06       94      150
##  1.01      236      481
##  1.01      318      602
##  1.02      256      660
##  1.01      336      831
##  1.02      199      592
##  1.01      372      696
##  1.01      307      478
##  1.03      153      318
##  1.06       93      127
##  1.01      382      633
##  1.01      458      979
##  1.01      443      642
##  1.02      350      840
##  1.01      357      467
##  1.01      365      577
##  1.01      334      553
##  1.01      311      466
##  1.00      512      906
##  1.01      222      469
##  1.03      144      365
##  1.01      221      434
##  1.02      310      659
```

```
##  1.01      252      684
##  1.02      236      515
##  1.01      356      790
##  1.02      211      576
##  1.02      358      824
##  1.01      540     1191
##  1.01      461      921
##  1.01      401      840
##  1.01      491      992
##  1.01      459      984
##  1.01      489      828
##  1.00      516     1061
##  1.01      442      901
##  1.01      416      961
##
##  # showing 73 of 1667 rows (change via 'max_rows' argument or 'cmdstanr_max_rows' option)
```

Pooled model

Summary of model fit for main parameters:

```
##           variable      mean    median      sd     mad        q5      q95 rhat
##  lp__               -7121.07  -7120.77    1.97    1.88  -7124.73 -7118.44 1.00
##  alpha             -12403.04 -12419.05 2677.43 2654.08 -16804.88 -7985.99 1.00
##  beta_SAT_ALL          94.83     94.76    5.22    5.30     86.29   103.29 1.00
##  beta_MD_FAMINIC        0.05      0.05    0.02    0.02      0.02     0.08 1.00
##  beta_COSTT4_A          0.36      0.36    0.03    0.03      0.31     0.42 1.00
##  beta_POVERTY_RATE   -202.47   -202.80   77.81   77.89   -331.33   -76.49 1.00
##  beta_URBAN          2118.20   2111.77  563.15  566.99   1207.57  3028.83 1.00
##  beta_PRIVATE       -7424.26  -7412.18  900.18  902.21  -8920.26 -5968.03 1.00
##  sigma               6639.77   6637.82  168.48  167.82   6366.31  6923.45 1.00
##  ess_bulk ess_tail
##      1899     2343
##      1798     2321
##      1712     2272
##      2518     2527
##      1769     2270
##      2642     2967
##      3569     2270
##      1717     2152
##      3647     2879
##
##  # showing 9 of 1603 rows (change via 'max_rows' argument or 'cmdstanr_max_rows' option)
```

Hierarchical model

Summary of model fit for main parameters:

```
##              variable      mean    median      sd     mad        q5       q95
##  lp__                   -7303.47  -7304.16   11.95   11.75  -7321.90  -7282.74
##  alpha[1]               -3203.81  -3283.73 6218.09 6111.41 -13207.11   7398.77
##  alpha[2]              -19128.67 -19179.05 4289.44 4319.33 -26262.42 -12147.62
##  alpha[3]              -13641.37 -13667.25 5406.40 5277.39 -22390.69  -4492.78
##  alpha[4]               -3183.94  -3231.94 6111.17 6135.62 -12709.73   7176.40
##  alpha[5]               -7838.31  -7990.70 4560.28 4731.35 -15184.17   -260.58
```

```
##  alpha[6]             -10166.61 -10166.60 5533.15 5437.28 -19354.58   -730.50
##  alpha[7]              -4318.45  -4213.42 7017.12 7097.10 -15980.59   6995.12
##  alpha[8]             -10742.15 -10688.10 5596.03 5728.54 -20074.80  -1806.04
##  alpha[9]              -3570.33  -3576.82 7273.87 7080.45 -15521.54   8404.23
##  beta_SAT_ALL[1]          97.58     97.23    9.59    8.66     82.35    113.75
##  beta_SAT_ALL[2]         106.18    105.99    8.28    8.34     93.03    120.17
##  beta_SAT_ALL[3]          97.80     97.46    8.84    8.34     83.83    112.68
##  beta_SAT_ALL[4]          85.31     86.37   10.75   10.88     66.38    100.96
##  beta_SAT_ALL[5]          85.54     85.71    7.74    7.95     72.21     97.69
##  beta_SAT_ALL[6]          95.55     95.65    9.51    9.03     79.86    111.06
##  beta_SAT_ALL[7]         100.40     99.55   10.50   10.02     84.75    118.54
##  beta_SAT_ALL[8]         100.55    100.28    9.25    8.89     85.91    116.08
##  beta_SAT_ALL[9]          92.29     93.01   13.89   11.62     68.32    113.30
##  beta_MD_FAMINIC[1]        0.05      0.06    0.03    0.03      0.00      0.10
##  beta_MD_FAMINIC[2]        0.07      0.07    0.02    0.02      0.03      0.11
##  beta_MD_FAMINIC[3]        0.04      0.04    0.03    0.03     -0.03      0.09
##  beta_MD_FAMINIC[4]        0.07      0.07    0.03    0.03      0.03      0.13
##  beta_MD_FAMINIC[5]        0.07      0.07    0.03    0.02      0.03      0.11
##  beta_MD_FAMINIC[6]        0.06      0.06    0.04    0.03      0.00      0.11
##  beta_MD_FAMINIC[7]        0.06      0.06    0.04    0.03      0.00      0.12
##  beta_MD_FAMINIC[8]        0.07      0.07    0.03    0.03      0.02      0.13
##  beta_MD_FAMINIC[9]        0.06      0.06    0.04    0.03     -0.01      0.12
##  beta_COSTT4_A[1]          0.20      0.21    0.07    0.06      0.07      0.29
##  beta_COSTT4_A[2]          0.27      0.27    0.05    0.05      0.19      0.35
##  beta_COSTT4_A[3]          0.27      0.26    0.06    0.06      0.18      0.38
##  beta_COSTT4_A[4]          0.23      0.23    0.06    0.05      0.14      0.32
##  beta_COSTT4_A[5]          0.24      0.24    0.04    0.04      0.17      0.31
##  beta_COSTT4_A[6]          0.22      0.23    0.06    0.06      0.12      0.31
##  beta_COSTT4_A[7]          0.19      0.21    0.07    0.06      0.06      0.29
##  beta_COSTT4_A[8]          0.24      0.24    0.05    0.05      0.16      0.33
##  beta_COSTT4_A[9]          0.23      0.24    0.08    0.07      0.09      0.35
##  beta_POVERTY_RATE[1]   -738.10   -739.53  538.87  532.60  -1631.90    132.86
##  beta_POVERTY_RATE[2]    180.19    180.48  166.11  163.05    -92.91    453.50
##  beta_POVERTY_RATE[3]     95.58    101.82  332.11  324.63   -456.54    635.19
##  beta_POVERTY_RATE[4]   -594.16   -589.92  430.65  435.18  -1296.09    110.12
##  beta_POVERTY_RATE[5]   -275.52   -275.56  149.40  150.07   -515.34    -27.24
##  beta_POVERTY_RATE[6]    -21.89    -21.30  184.89  181.09   -324.86    272.93
##  beta_POVERTY_RATE[7]  -1285.38  -1286.05  496.61  510.66  -2095.87   -474.30
##  beta_POVERTY_RATE[8]    293.17    294.16  286.49  288.56   -174.91    760.33
##  beta_POVERTY_RATE[9]   -447.71   -446.12  189.33  190.94   -763.05   -138.71
##  beta_URBAN[1]           804.12    788.92 1785.17 1774.72  -2085.41   3698.00
##  beta_URBAN[2]          3611.35   3595.78 1293.70 1267.83   1501.19   5773.99
##  beta_URBAN[3]           872.97    868.80 1183.82 1164.89  -1109.66   2786.20
##  beta_URBAN[4]          -255.89   -253.51 1288.04 1270.16  -2374.14   1850.79
##  beta_URBAN[5]          1628.11   1643.11  861.40  880.52    187.01   2999.97
##  beta_URBAN[6]          -214.93   -210.98 1290.82 1247.92  -2338.81   1871.07
##  beta_URBAN[7]          1388.89   1341.20 1669.32 1592.21  -1275.97   4292.31
##  beta_URBAN[8]           -23.55    -17.08 1474.08 1462.84  -2442.38   2392.30
##  beta_URBAN[9]           299.88    344.00 2352.82 2167.53  -3716.60   4089.56
##  beta_PRIVATE[1]       -4287.66  -4300.20 2404.11 2374.57  -8212.90   -361.76
##  beta_PRIVATE[2]       -4648.32  -4632.10 1552.57 1531.14  -7148.08  -2093.10
##  beta_PRIVATE[3]       -3838.46  -3805.72 1649.62 1552.02  -6692.44  -1132.81
##  beta_PRIVATE[4]       -3630.28  -3638.58 1735.17 1755.56  -6446.92   -756.02
##  beta_PRIVATE[5]       -5028.85  -5048.76 1175.33 1145.27  -6944.34  -3083.89
```

```
##  beta_PRIVATE[6]       -5504.69  -5523.42 1697.12 1752.86  -8252.18  -2713.14
##  beta_PRIVATE[7]        -188.35   -174.01 2066.58 2107.09  -3632.52   3179.03
##  beta_PRIVATE[8]       -6214.05  -6169.92 1822.84 1835.57  -9243.39  -3310.41
##  beta_PRIVATE[9]        -609.65   -713.51 3279.20 3149.21  -6039.16   4924.02
##  sigma                  5934.20   5927.77  157.69  156.07   5679.35   6203.95
##  mu_alpha                 -8.24    -10.87   99.96   97.52   -175.52    158.45
##  sigma_alpha           10001.89  10000.80  100.30  100.74   9839.86  10167.01
##  mu_SAT_ALL               95.44     95.37    7.22    6.38     84.19    106.49
##  sigma_SAT_ALL            14.27     12.40   10.92    6.83      3.42     29.85
##  mu_MD_FAMINIC             0.06      0.06    0.02    0.02      0.02      0.10
##  sigma_MD_FAMINIC          0.03      0.03    0.02    0.02      0.00      0.08
##  mu_COSTT4_A               0.23      0.23    0.04    0.04      0.16      0.30
##  sigma_COSTT4_A            0.06      0.05    0.04    0.04      0.01      0.15
##  mu_POVERTY_RATE        -179.45   -184.45  340.37  318.42   -723.79    386.16
##  sigma_POVERTY_RATE     1359.23   1313.23  525.58  544.58    549.76   2276.37
##  mu_URBAN                278.85    277.96  465.22  463.82   -488.29   1050.97
##  sigma_URBAN            2309.95   2306.90  462.08  471.58   1577.15   3075.12
##  mu_PRIVATE             -726.99   -727.72  494.66  504.97  -1534.59     90.46
##  sigma_PRIVATE          3093.78   3090.69  432.93  436.76   2394.08   3795.91
##  mu_sigma              10000.71   9997.39  505.82  515.67   9185.60  10827.47
##  sigma_sigma            9970.02   9969.10  509.55  495.78   9109.97  10817.52
##  rhat ess_bulk ess_tail
##  1.00      319      500
##  1.00     2377     1868
##  1.00     1284     2279
##  1.00     2052     2523
##  1.00     1610     2336
##  1.00     1353     2531
##  1.00     2490     1878
##  1.00     2141     2596
##  1.00     2246     2508
##  1.00     2731     2453
##  1.00     1764     1849
##  1.00     1084     2454
##  1.00     1775     2417
##  1.00     1044     2219
##  1.00      991     1846
##  1.00     2187     1971
##  1.00     2081     2728
##  1.00     1995     2384
##  1.00     2321     2195
##  1.00     1658     1692
##  1.00     2248     2442
##  1.00      860     1906
##  1.00     1682     2633
##  1.00     1784     2768
##  1.00     1781     1991
##  1.00     2042     1838
##  1.00     2221     1962
##  1.00     2034     1581
##  1.00      878     1730
##  1.00     1131     1571
##  1.00     1239     1784
##  1.00     1551     2123
```

```
##  1.00     1657     1809
##  1.00     1468     1605
##  1.00     1019     1645
##  1.00     1589     2118
##  1.00     1831     1405
##  1.00     2646     2950
##  1.00     2498     2520
##  1.00     2017     2503
##  1.00     2998     2776
##  1.00     2879     2771
##  1.00     3268     2936
##  1.00     2772     2763
##  1.00     2779     1642
##  1.00     2870     2667
##  1.00     3708     2527
##  1.00     3670     3032
##  1.00     3658     2190
##  1.00     3487     2845
##  1.00     3030     1893
##  1.00     4231     3034
##  1.00     3751     2386
##  1.00     3736     1977
##  1.00     3959     2877
##  1.00     1379     2151
##  1.00     1632     2430
##  1.00     1888     2091
##  1.00     1859     2366
##  1.00     1847     1652
##  1.00     2082     2574
##  1.00     2831     2922
##  1.00     1971     2693
##  1.00     3131     1980
##  1.00     3805     2685
##  1.00     4520     2744
##  1.00     4381     2742
##  1.00     1739     2276
##  1.01      360      288
##  1.00     1339     1967
##  1.02      240      265
##  1.00     1112     1716
##  1.00      451      521
##  1.00     3485     2806
##  1.00     2803     2062
##  1.00     4103     2809
##  1.00     2850     2629
##  1.00     3184     2405
##  1.00     2245     2435
##  1.00     4422     3085
##  1.00     3704     2166
##
##  # showing 81 of 1675 rows (change via 'max_rows' argument or 'cmdstanr_max_rows' option)
```

6. Stan Specifications

We used the following Stan options:

- Interface: cmdstanr
- Seed: 1234
- Chains: 4 (default)
- Iterations per chain: 2000 (default)
- Warmup/burnin: 1000 iterations (default)

7. Convergence Diagnostics

In this section, we are performing the diagnostics for the separate and hierarchical model based on the first region (N = 37). This region was selected at random, and it is not the region with the lowest or highest number of samples. Although, considering separately all regions would have given a better overall picture, for simplicity reasons, one region is selected. For the project group, this seemed to be more interesting way to do analysis than blending all the regions together with for example averages.

Rhat

We used the rhat() function for all models to get the rhat convergence diagnostic. The function compares the between chain and within chain estimates for model parameters and other univariate quantities of interest. If the between chain and within chain estimates agree, it is said that the chains have mixed well. This happens when R-hat is less than 1.05 or less than 1.01 depending on the standard. We decided not be strict with our multivariate model and chose the less strict threshold of 1.05.

Rhat statistics revealed that the separate model converged worse than the hierarchical or pooled model, both of which got good rhat convergence values for every beta in every chain. The fact that the separate model has problems with convergence while the pooled model and the hierarchical model converge very well speaks clearly against the separate model while suggesting the other two models perform well on this front.

Separate model Rhat

```
## # A tibble: 7 x 5
##   Parameter            `Chain 1` `Chain 2` `Chain 3` `Chain 4`
##   <chr>                    <dbl>     <dbl>     <dbl>     <dbl>
## 1 alpha[1]                  1.01      1.01      1.01      1.01
## 2 beta_SAT_ALL[1]           1.04      1.00      1.03      1.03
## 3 beta_MD_FAMINIC[1]        1.01      1.00      1.02      1.00
## 4 beta_COSTT4_A[1]          1.01      1.01      1.01      1.04
## 5 beta_POVERTY_RATE[1]      1.01      1.03      1.01      1.00
## 6 beta_URBAN[1]             1.00      1.01      1.02      1.01
## 7 beta_PRIVATE[1]           1.01      1.01      1.02      1.04
```

Pooled model Rhat

```
## # A tibble: 8 x 5
##   Parameter         `Chain 1` `Chain 2` `Chain 3` `Chain 4`
##   <chr>                 <dbl>     <dbl>     <dbl>     <dbl>
## 1 alpha                 0.999     1.00      1.01      0.999
## 2 beta_SAT_ALL          0.999     1.00      1.00      1.00
## 3 beta_MD_FAMINIC       1.01      1.00      0.999     1.00
## 4 beta_COSTT4_A         1.00      0.999     0.999     1.01
```

```
## 5 beta_POVERTY_RATE     0.999     0.999     1.00      0.999
## 6 beta_URBAN            0.999     1.00      1.00      1.01
## 7 beta_PRIVATE          0.999     1.00      1.00      1.01
## 8 sigma                 1.00      0.999     1.00      1.00
```

Hierarchical model Rhat

```
## # A tibble: 7 x 5
##   Parameter            `Chain 1` `Chain 2` `Chain 3` `Chain 4`
##   <chr>                    <dbl>     <dbl>     <dbl>     <dbl>
## 1 alpha[1]                  1.00     1.00       1.00     0.999
## 2 beta_SAT_ALL[1]           1.00     0.999      1.00     1.00
## 3 beta_MD_FAMINIC[1]        1.01     1.00       1.00     1.00
## 4 beta_COSTT4_A[1]          1.01     1.00       1.00     1.00
## 5 beta_POVERTY_RATE[1]      1.00     1.00       1.00     1.00
## 6 beta_URBAN[1]             1.00     1.00       1.00     1.00
## 7 beta_PRIVATE[1]           1.01     1.01       1.00     1.00
```

The codes for the plots are provided in the appendix.

Effective sample size (ESS)

Effective sample size (ESS) evaluates the uncertainty in estimates that is caused by autocorrelation of the chains. The value of effective sample size represents the number of independent samples that poses the same predictive power as all the autocorrelated samples. In other words, if the number of effective sample size is high, then the number of independent samples is high. (Stan reference manual, n.d)

For the separate model the ESS values range from 54 to 96 (N = 37). For the pooled model the ESS values range from 408 to 996 (N = 766). For the hierarchical model the ESS values range from 340 to 1225 (N = 766).

For pooled and hierarchical models, for many but not all parameters there are auto correlated samples that are reduced from the effective sample size. There are also in some cases a lot of independent samples when ESS is close to N. However, for all the three models, especially for the separate modelthere are ESS values that are greater than N.

According to Stan reference manual, ESS can be higher than the number of samples due to anthitehetic Markov Chains which have negative auto correlations on odd lags. Based on the same source, another reason for this could be the NUTS algorithm used in Stan, which might give ESS that is higher than the sample size for parameters that have close to Gaussian posterior and low dependency on the rest of the parameters.

Separate Model ESS

```
## # A tibble: 7 x 2
##   Parameter              ESS
##   <chr>                <dbl>
## 1 alpha[1]              55.7
## 2 beta_COSTT4_A[1]      50.0
## 3 beta_MD_FAMINIC[1]    69.2
## 4 beta_POVERTY_RATE[1]  59.8
## 5 beta_PRIVATE[1]       55.7
## 6 beta_SAT_ALL[1]       49.5
## 7 beta_URBAN[1]        105.
```

Pooled Model ESS

```
## # A tibble: 8 x 2
```

```
##   Parameter           ESS
##   <chr>             <dbl>
## 1 alpha              452.
## 2 beta_COSTT4_A      444.
## 3 beta_MD_FAMINIC    643.
## 4 beta_POVERTY_RATE  635.
## 5 beta_PRIVATE       427.
## 6 beta_SAT_ALL       431.
## 7 beta_URBAN         879.
## 8 sigma              911.
```

Hierarchical Model ESS

```
## # A tibble: 7 x 2
##   Parameter              ESS
##   <chr>                <dbl>
## 1 alpha[1]              593.
## 2 beta_COSTT4_A[1]      222.
## 3 beta_MD_FAMINIC[1]    390.
## 4 beta_POVERTY_RATE[1]  655.
## 5 beta_PRIVATE[1]       355.
## 6 beta_SAT_ALL[1]       444.
## 7 beta_URBAN[1]         903.
```

HMC specific convergence diagnostics for all models (divergences, tree depth)

HMC specific convergence diagnostics were analysed for all models. The R code and output for diagnostics can be found from the appendix. Based on the HMC specific convergence diagnostics, the separate model reached the initial maximum tree depth of 10 in 100% of the transitions. The diagnostics state that this leads to premature termination of trajectories and slow exploration. The proposed action of increasing the limit was tested, but it resulted to too long running time for the already time consuming fitting of the separate model. The rest of the diagnostics consisting of divergences, E-BMFI, effective sample size and split R-hat satisfactory.

For the pooled model, tree depth, divergences, E-BFMI, effective sample size and split R-hat were satisfactory.

For the hierarchical model, only 4 out of 4000 hit the maximum tree depth and 23 out of 4000 transitions did not converge. Due to relatively low numbers (0.1% and 0.57%) no actions were taken for the hierarchical model. The rest of the diagnostics were satisfactory.

8. Posterior Predictive Checks and Model Comparison

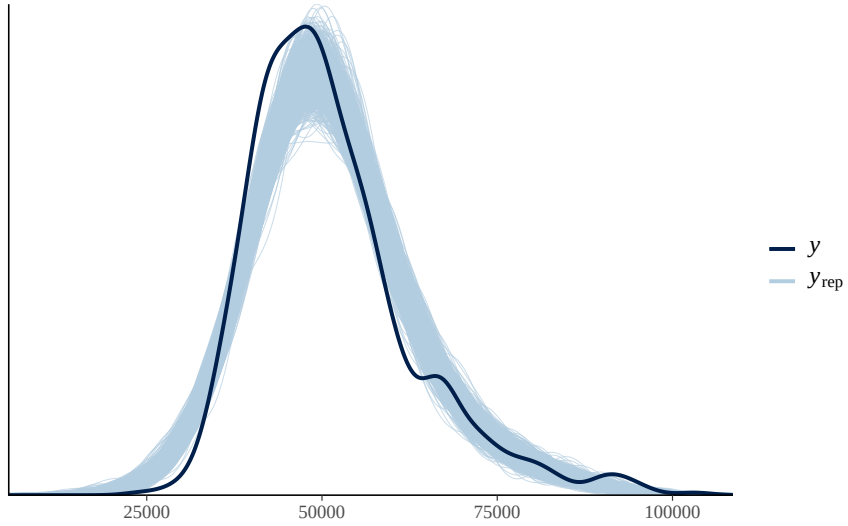
Posterior Predictive Checks

When doing posterior predictive checking we look for systematic discrepancies between the real observations and the data we get from simulating replicated data under the fitted model (Gelman and Hill, 2006). Posterior predictive checking is a form of internal validation that is a helpful phase of model building and checking in assessing whether our fitted model makes sense.

When comparing the densities of y (real sample values) and y_rep (Stan normal_rng generated values) in the graphs below, it looks like that y and y_rep are aligning quite well, with all three models, although it seems that the y_rep values are on average somewhat more on right (higher estimates for earnings). There are also intervals where y is suddenly changing direction, which are not visible in y_rep values.

Separate model

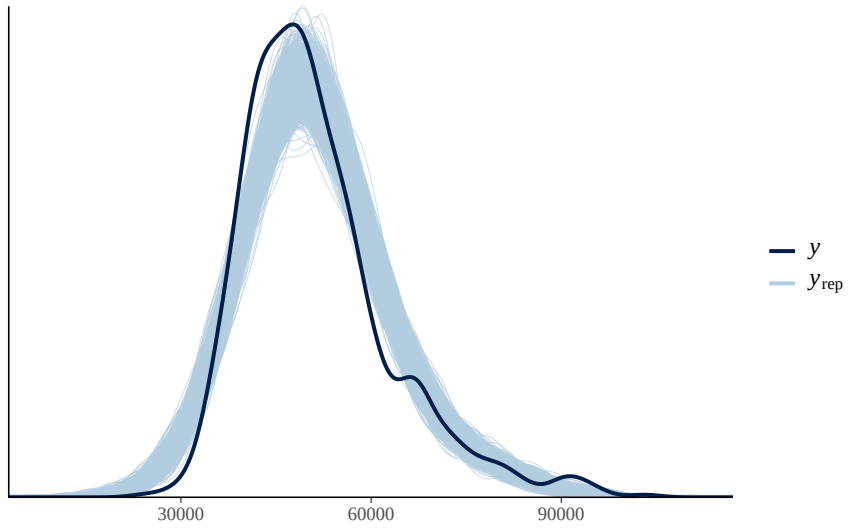
Comparing densities of y and y_rep



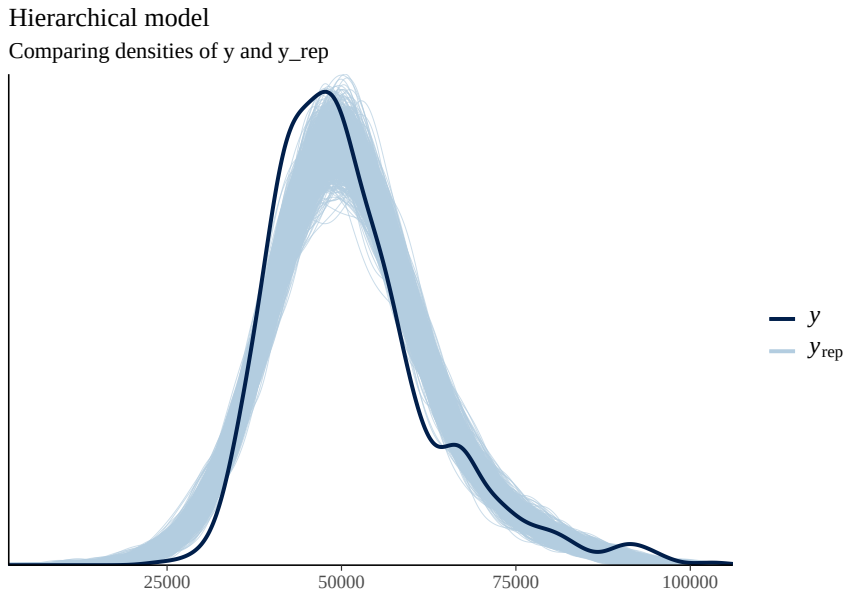

```
## Saving 6.5 x 4.5 in image
```

Pooled model

Comparing densities of y and y_rep




```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

Model comparison (LOO, K hat)

LOO elpd_loo is an estimate of the expected log pointwise pridictive dentisty (ELPD). elpd_loo sums individual pointwise log predictive densities (Stan Reference Manual, n.d).

Based on elpd_loo the model with the highest value should be selected (highest ELPD). From the tables below we can observe that separate model has the highest elpd_loo value, hierarchical model the second highest value, and the pooled the lowest value. For that reason, separate model is suggested by elpd_loo although the value of hierachical model comes quite close.

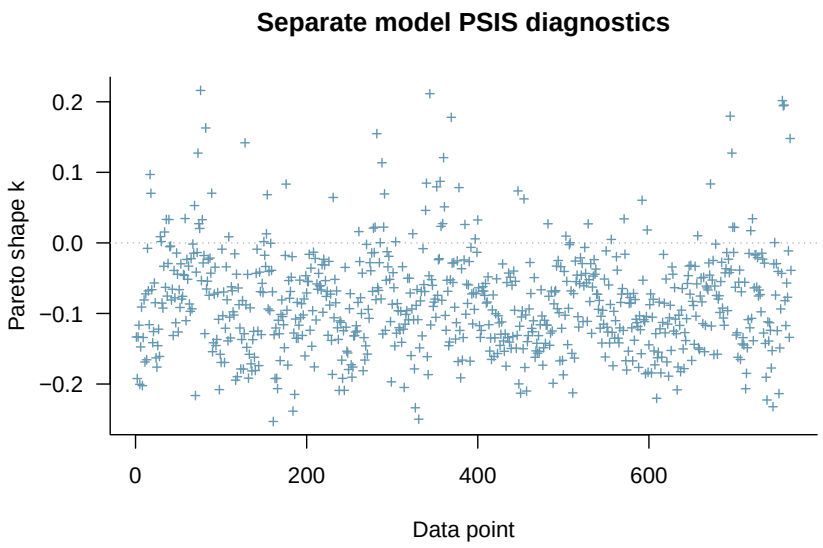
```
##        elpd_diff  se_diff
## model1        0.0        0.0
## model3      -50.6        6.2
## model2 -7510312.4    76791.0
```

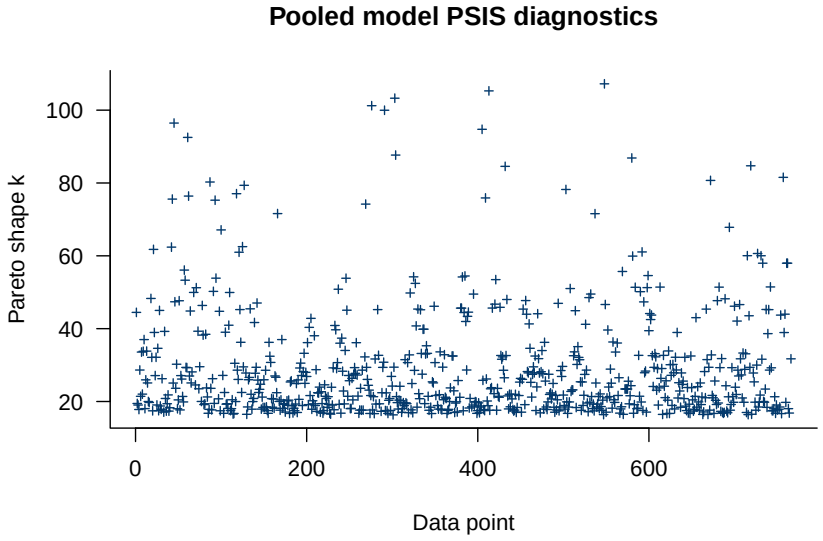
K hat

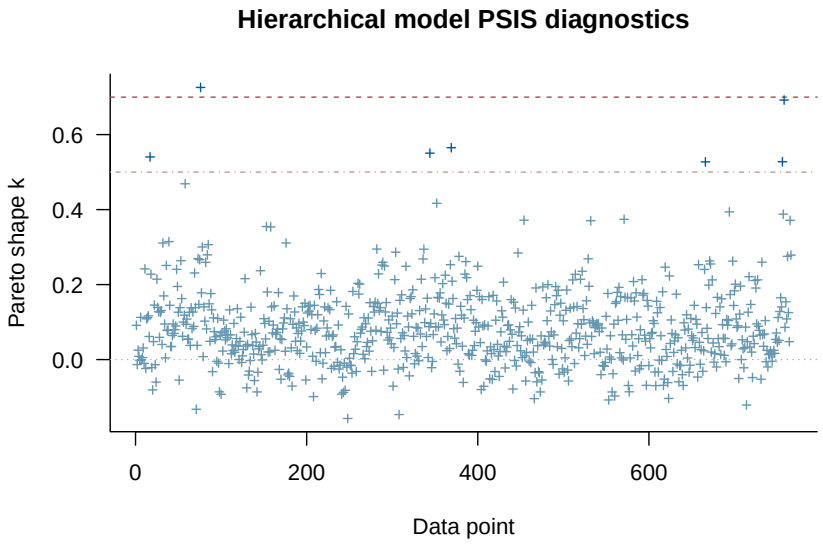
In general, if $\hat{k}$ values are greater than 0.7 the PSIS-LOO estimates might be too optimistic (biased), and on the contrary, if the $\hat{k}$ values are smaller than 0.7 the estimated can be considered reliable. (Gabry, Gelman, Vehtari, 2016)

As can be observed from the graphs "Separate mode PSIS diagnostics", "Pooled model PSIS diagnostics" and "Hierarchical model PSIS diagnostics" for separate model all $\hat{k}$ values are smaller than 0.7, for hierarchical only one value $\hat{k}$ is greater than 0.7. This means that PSIS-LOO estimates for these models can be considered reliable. However, for pooled model there are many $\hat{k}$ values that are a lot greater than 0.7. This indicates that this model may be biased and the estimates might not be reliable.

The more precise diagnostics for $\hat{k}$ are available in appendix.


Separate model PSIS diagnostics
Pareto shape k
0.2
0.1
0.0
−0.1
−0.2
0
200
400
600
Data point


Pooled model PSIS diagnostics
Pareto shape k
100
80
60
40
20
0
200
400
600
Data point



9. Predictive Performance Assessment

We use RMSE for assessing the predictive ability of our model. RMSE is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \tilde{y})^2}$$

, where y is the observed value and $\tilde{y}$ is the predicted value.

Separate model

```
##     variable     mean   median      sd     mad       q5      q95 rhat ess_bulk
##  y_tilde[1]  50024.65 49994.80 5632.13 5637.51 40808.43 59236.57 1.00     3567
##  y_tilde[2]  43066.82 43044.70 5583.30 5615.87 33891.36 52289.05 1.00     3909
##  y_tilde[3]  56423.50 56371.80 5685.63 5611.86 47247.52 65883.54 1.00     4036
##  y_tilde[4]  51135.80 51161.55 5658.34 5709.71 41775.26 60387.50 1.00     3718
##  y_tilde[5]  55297.81 55266.30 5599.13 5578.73 46293.06 64695.44 1.00     3983
##  y_tilde[6]  54862.37 54807.80 5748.50 5776.21 45600.48 64175.22 1.00     4186
##  y_tilde[7]  44448.33 44549.65 5752.58 5654.41 35020.63 53457.44 1.00     3265
##  y_tilde[8]  47565.15 47612.65 5735.95 5786.96 38226.07 56928.38 1.00     4084
##  y_tilde[9]  44235.76 44176.25 5749.20 5713.87 34923.36 53801.53 1.00     3602
##  y_tilde[10] 50325.90 50286.40 5598.19 5634.03 41177.46 59628.46 1.00     4015
##  y_tilde[11] 49366.67 49324.30 5699.62 5656.64 40062.38 58722.53 1.00     3945
##  y_tilde[12] 37016.38 37000.80 5692.19 5645.15 27493.23 46469.74 1.00     4040
##  y_tilde[13] 61014.66 60989.75 5843.77 5764.65 51115.62 70516.84 1.00     3860
##  y_tilde[14] 40111.47 40028.45 5655.79 5770.80 30992.56 49452.31 1.00     3730
##  y_tilde[15] 53410.94 53357.40 5887.75 5753.53 43575.70 63121.20 1.00     4052
##  y_tilde[16] 40599.40 40602.75 5784.04 5732.62 31019.03 50029.54 1.00     3912
##  y_tilde[17] 42460.03 42466.35 5683.28 5643.81 32997.29 51644.20 1.00     3753
##  y_tilde[18] 58177.34 58203.65 5713.66 5647.89 48864.28 67584.90 1.00     3919
##  y_tilde[19] 36274.96 36189.00 5730.97 5763.24 26884.18 45725.20 1.00     3831
##  y_tilde[20] 38032.75 37940.90 5714.92 5595.18 28522.11 47453.74 1.00     3884
##  y_tilde[21] 52755.67 52739.15 5774.21 5762.87 43209.55 62208.61 1.00     3855
```

```
##  y_tilde[22] 54040.73 54169.60 5689.72 5753.97 44576.90 63335.09 1.00     3828
##  y_tilde[23] 53668.47 53611.40 5717.41 5797.34 44462.82 63076.66 1.00     4049
##  y_tilde[24] 57467.66 57398.90 5684.83 5616.39 48274.77 67119.70 1.00     4130
##  y_tilde[25] 56570.46 56569.15 5776.63 5800.08 47379.47 65907.75 1.00     4019
##  y_tilde[26] 48302.54 48269.30 5725.11 5771.39 38990.44 57677.45 1.00     3945
##  y_tilde[27] 50720.78 50876.55 5788.48 5550.71 41152.57 60022.49 1.00     3932
##  y_tilde[28] 49080.90 49017.30 5897.33 5978.44 39206.50 58656.10 1.00     3659
##  y_tilde[29] 51036.44 50988.00 5823.69 5959.90 41184.27 60513.63 1.00     3749
##  y_tilde[30] 49702.78 49793.95 5652.66 5692.74 40489.29 58797.02 1.00     3882
##  ess_tail
##      3681
##      3641
##      3889
##      3563
##      4015
##      3890
##      3665
##      3831
##      3929
##      3876
##      3689
##      3940
##      3612
##      3695
##      3850
##      4056
##      3700
##      3758
##      3506
##      3576
##      4056
##      3833
##      3844
##      3888
##      3848
##      3653
##      3661
##      3518
##      3726
##      4055
```

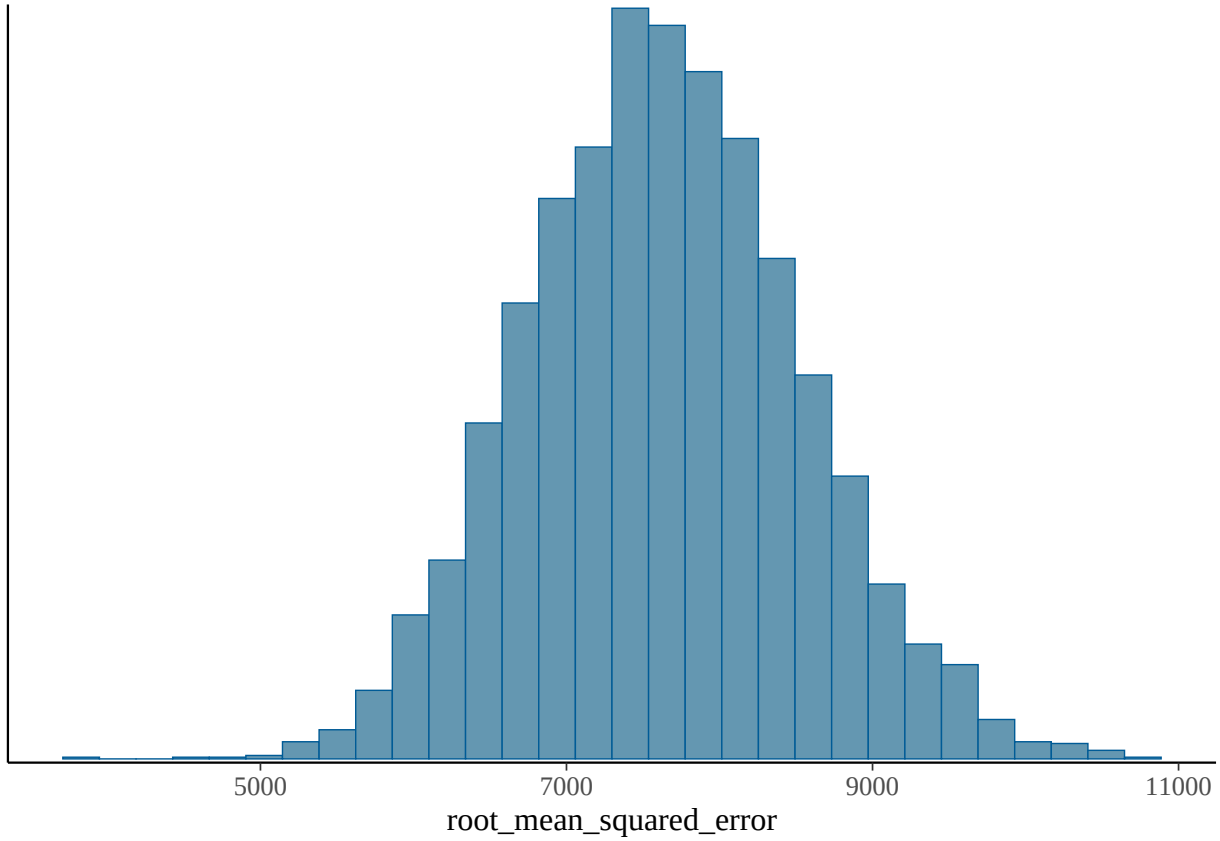
```
##                   variable         mean       median           sd          mad
##  indiv_squared_errors[1]   41147935.13  19223500.00  55884470.50  26154257.49
##  indiv_squared_errors[2]   31835855.93  14803400.00  43963129.55  20093403.52
##  indiv_squared_errors[3]   54197286.67  27899600.00  72270091.36  37680953.58
##  indiv_squared_errors[4]   64250769.07  35253400.00  78639467.92  46883792.78
##  indiv_squared_errors[5]   38216300.22  17402550.00  53981617.04  23790778.12
##  indiv_squared_errors[6]   38789126.22  18237900.00  53394117.03  24738678.43
##  indiv_squared_errors[7]   35434978.23  16534500.00  50371029.09  22465889.69
##  indiv_squared_errors[8]   44976487.06  21226650.00  61845070.69  28992672.95
##  indiv_squared_errors[9]   77998787.03  45802600.00  91398224.40  60015662.83
##  indiv_squared_errors[10]  33573153.48  15894200.00  46097082.20  21435838.52
##  indiv_squared_errors[11]  32554431.19  14471800.00  46303505.22  19905165.21
##  indiv_squared_errors[12]  56721304.22  29549600.00  72113212.89  39665895.53
```

```
##  indiv_squared_errors[13]  74650034.76  43092200.00  89142832.48  55467698.37
##  indiv_squared_errors[14]  82142180.93  52795600.00  90436718.07  66652232.62
##  indiv_squared_errors[15]  88106787.93  54149950.00 100556776.21  68421530.39
##  indiv_squared_errors[16]  54300572.24  27562250.00  72637782.83  37003946.53
##  indiv_squared_errors[17]  33753367.01  14960000.00  47435861.58  20844752.58
##  indiv_squared_errors[18]  36029483.59  17146650.00  51220788.00  23304151.73
##  indiv_squared_errors[19]  69936629.48  40895900.00  82938298.59  53954252.88
##  indiv_squared_errors[20]  42180225.62  20279150.00  57948422.62  27457588.91
##  indiv_squared_errors[21]  69722140.03  39741700.00  85045812.05  52910502.48
##  indiv_squared_errors[22]  79916034.95  50725000.00  88498643.42  64526799.50
##  indiv_squared_errors[23] 176501382.08 145192000.00 143385643.89 130459904.40
##  indiv_squared_errors[24]  96095974.01  63601700.00 104455448.77  76327657.98
##  indiv_squared_errors[25]  71041771.11  40941750.00  85865815.23  54178874.19
##  indiv_squared_errors[26]  33347107.86  15116550.00  47061618.55  20641683.82
##  indiv_squared_errors[27]  87561520.99  57913550.00  95912849.68  70892075.73
##  indiv_squared_errors[28]  46686845.00  22038250.00  64308104.59  30068447.51
##  indiv_squared_errors[29]  38989391.85  17828200.00  53307082.50  24602553.51
##  indiv_squared_errors[30]  41420006.84  18859150.00  56948478.07  25643805.73
##          q5          q95 rhat ess_bulk ess_tail
##   187440.80 156306250.00 1.00     4251     3957
##   142931.70 118621850.00 1.00     3487     3738
##   242977.40 194622350.00 1.00     3702     3887
##   347249.70 226162050.00 1.00     3609     3537
##   161451.80 149351500.00 1.00     4033     3918
##   161451.70 149863750.00 1.00     3958     3945
##   156044.30 134004650.00 1.00     3809     3812
##   195664.05 167590450.00 1.00     4163     3952
##   439642.30 264731000.00 1.00     3939     3954
##   129590.80 127866100.00 1.00     3817     3665
##   126912.55 126944000.00 1.00     3780     3756
##   315111.30 206949850.00 1.00     3904     3940
##   521847.95 251755500.00 1.00     3842     3971
##   693501.65 262488100.00 1.00     3769     3695
##   682895.75 289721250.00 1.00     4183     3799
##   291175.20 200821350.00 1.00     3825     4055
##   133211.40 131627800.00 1.00     4048     3945
##   148689.80 138538200.00 1.00     4138     3724
##   398863.00 239880600.00 1.00     3639     3323
##   171143.90 164791300.00 1.00     4119     3970
##   419915.90 242689550.00 1.00     3617     3890
##   553909.45 262120450.00 1.00     3872     3887
##  7689941.00 449362500.00 1.00     4038     3764
##  1179233.50 311124400.00 1.00     4135     3888
##   472414.30 239497650.00 1.00     3975     4058
##   134304.35 126714350.00 1.00     3956     3493
##   750135.15 277371300.00 1.00     4135     3890
##   211311.70 181511650.00 1.00     4162     3532
##   166894.05 146061550.00 1.00     3938     3576
##   195732.70 153869300.00 1.00     3952     3817

##        variable          mean        median           sd          mad
##  sum_of_squares 1772077873.00 1740905000.00 411138217.38 402978093.00
##              q5           q95 rhat ess_bulk ess_tail
##  1150069500.00 2496030500.00 1.00     3844     3857
```

```
##                 variable    mean  median     sd    mad      q5     q95 rhat
##  root_mean_squared_error 7634.15 7617.76 888.40 886.37 6191.57 9121.47 1.00
##  ess_bulk ess_tail
##      3844     3857
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Pooled model

```
##     variable     mean   median      sd     mad       q5      q95 rhat ess_bulk
##  y_tilde[1]  49131.13 48954.80 6761.80 6734.64 38326.88 60122.77 1.00     4205
##  y_tilde[2]  46422.00 46387.30 6589.51 6585.56 35606.80 57344.37 1.00     3911
##  y_tilde[3]  59186.63 59130.55 6649.03 6648.13 48361.22 70237.63 1.00     3691
##  y_tilde[4]  45598.56 45400.30 6562.88 6446.12 34962.12 56772.33 1.00     3860
##  y_tilde[5]  60137.66 60139.80 6690.51 6678.96 49175.85 71231.37 1.00     3722
##  y_tilde[6]  50560.25 50562.65 6668.20 6582.23 39710.85 61532.78 1.00     3486
##  y_tilde[7]  44694.38 44710.00 6663.78 6687.79 33660.71 55764.10 1.00     4013
##  y_tilde[8]  47751.56 47891.00 6616.97 6645.09 36952.29 58476.71 1.00     3986
##  y_tilde[9]  47284.08 47202.00 6658.89 6756.58 36260.28 58424.84 1.00     4085
##  y_tilde[10] 48876.57 48905.05 6676.11 6788.16 38151.39 59919.76 1.00     3907
##  y_tilde[11] 50558.84 50551.85 6670.06 6610.84 39781.20 61826.17 1.00     3859
##  y_tilde[12] 40206.71 40333.75 6657.74 6659.99 29112.46 51095.35 1.00     3952
##  y_tilde[13] 62584.88 62457.00 6661.15 6698.83 52164.36 73612.78 1.00     3937
##  y_tilde[14] 38794.90 38846.90 6724.77 6714.18 27682.66 49867.80 1.00     3992
```

```
##  y_tilde[15] 52260.37 52343.90 6558.56 6641.90 41693.13 63131.38 1.00     4089
##  y_tilde[16] 39438.95 39468.20 6782.08 6805.73 28223.56 50350.32 1.00     3492
##  y_tilde[17] 45286.32 45264.70 6635.45 6571.18 34275.77 56181.54 1.00     4013
##  y_tilde[18] 55890.74 55854.35 6763.09 6753.54 44394.68 66863.67 1.00     4070
##  y_tilde[19] 38626.34 38644.35 6804.20 6852.73 27470.80 49872.45 1.00     3885
##  y_tilde[20] 41572.74 41559.20 6689.02 6713.81 30731.42 52459.46 1.00     4085
##  y_tilde[21] 49752.46 49707.75 6641.45 6562.14 38786.04 60665.37 1.00     3568
##  y_tilde[22] 54047.00 54094.30 6617.23 6581.11 42776.63 64788.56 1.00     3958
##  y_tilde[23] 51931.68 52019.70 6684.18 6631.30 40769.96 62910.65 1.00     3879
##  y_tilde[24] 58872.97 58848.80 6698.12 6809.06 47797.52 69979.52 1.00     4019
##  y_tilde[25] 54270.03 54313.85 6650.90 6691.05 43611.11 65050.44 1.00     3798
##  y_tilde[26] 50022.60 50086.20 6703.02 6647.76 39095.90 60799.14 1.00     4095
##  y_tilde[27] 49450.35 49486.30 6721.89 6672.52 38359.77 60562.26 1.00     4042
##  y_tilde[28] 50343.11 50411.35 6738.58 6766.81 39350.21 61403.31 1.00     3904
##  y_tilde[29] 49563.17 49594.45 6653.03 6664.88 38708.05 60344.20 1.00     3916
##  y_tilde[30] 48328.27 48299.45 6744.63 6677.11 37072.50 59324.39 1.00     3724
##  ess_tail
##      3725
##      3687
##      3622
##      3862
##      4081
##      3556
##      3735
##      3784
##      3970
##      3723
##      3814
##      3954
##      3851
##      3912
##      3849
##      3866
##      4082
##      4088
##      4003
##      3635
##      3954
##      3848
##      3967
##      3987
##      3545
##      3767
##      3970
##      3960
##      3828
##      3743
```

```
##                  variable         mean       median           sd          mad
##  indiv_squared_errors[1]   50454764.69  22843250.00  73586029.83  31478770.76
##  indiv_squared_errors[2]   60833075.70  28797950.00  81998750.51  39099305.11
##  indiv_squared_errors[3]   47863395.75  21357000.00  68667662.91  29045105.10
##  indiv_squared_errors[4]  168846585.79 130466500.00 155991537.76 136777930.17
##  indiv_squared_errors[5]  100428088.03  59125300.00 117862079.26  76463812.55
```

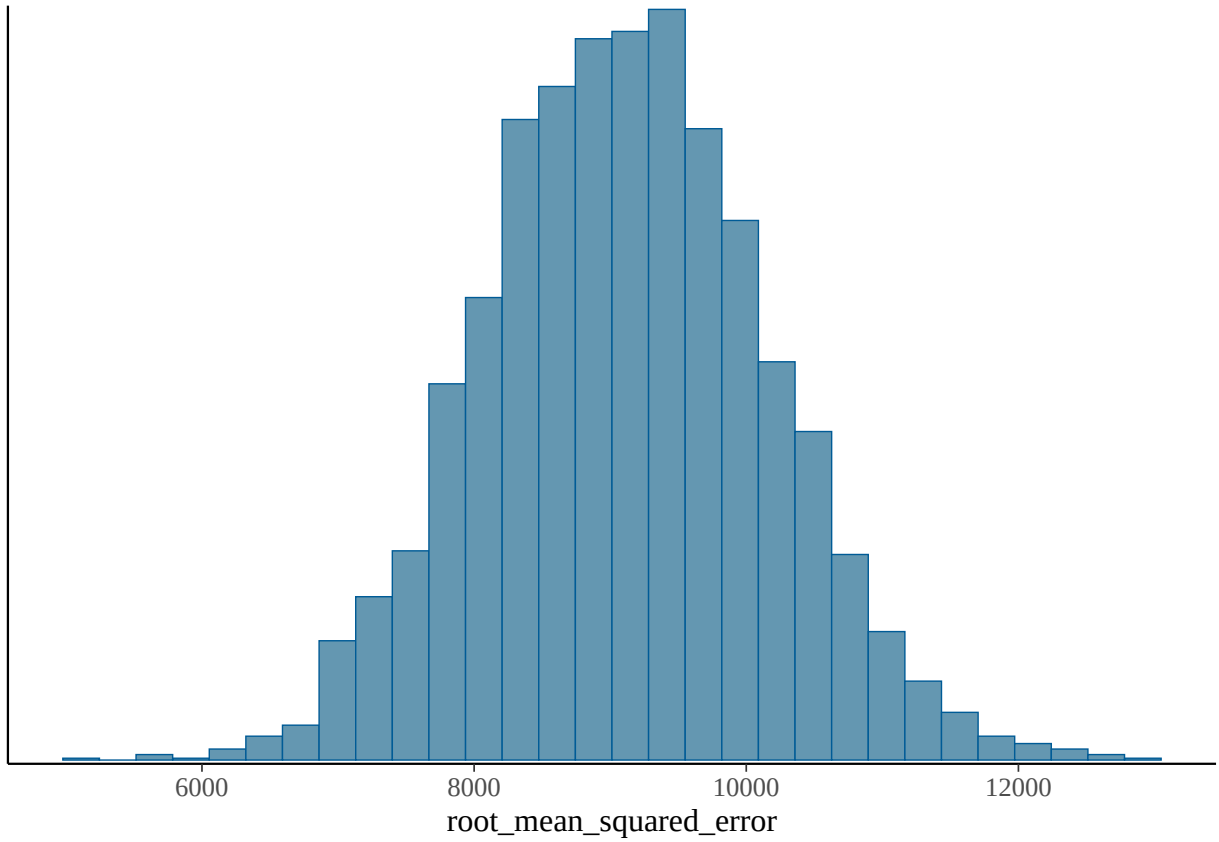
```
##  indiv_squared_errors[6]   48077988.35  22059050.00  66621141.62  30096817.06
##  indiv_squared_errors[7]   47560996.12  21948850.00  68002704.82  29998565.16
##  indiv_squared_errors[8]   57187719.03  27734500.00  78040800.02  37447295.82
##  indiv_squared_errors[9]  139452348.59  94081000.00 144780728.80 113266859.37
##  indiv_squared_errors[10]  53240706.14  24618900.00  72448200.99  34013275.46
##  indiv_squared_errors[11]  45315554.31  20283200.00  65759468.18  27598940.00
##  indiv_squared_errors[12] 110292803.68  69626100.00 123088074.72  88253099.34
##  indiv_squared_errors[13] 107322169.63  62994450.00 126825881.76  82692378.24
##  indiv_squared_errors[14] 115756149.83  71889750.00 130283319.86  91206364.41
##  indiv_squared_errors[15]  80954215.49  43418150.00 101544414.17  58065747.31
##  indiv_squared_errors[16]  78784247.71  39939000.00  99314938.70  54029227.96
##  indiv_squared_errors[17]  60301943.56  28822100.00  83349011.05  39091625.24
##  indiv_squared_errors[18]  62770435.18  29687300.00  85427255.54  40606197.51
##  indiv_squared_errors[19]  60270577.92  29360350.00  80505357.12  40082372.69
##  indiv_squared_errors[20]  88645513.69  48531050.00 111227510.37  64725734.77
##  indiv_squared_errors[21] 125738803.79  83337650.00 134568704.25 101752913.64
##  indiv_squared_errors[22]  91414474.97  52662350.00 108056036.06  68838363.38
##  indiv_squared_errors[23] 233161362.57 186085500.00 194815913.89 174515956.44
##  indiv_squared_errors[24] 133062697.44  88647550.00 138143192.30 107800439.04
##  indiv_squared_errors[25]  58953829.12  27593200.00  81766603.78  37272059.92
##  indiv_squared_errors[26]  51072660.80  23448300.00  72111997.76  31974500.31
##  indiv_squared_errors[27]  82167538.58  44095250.00 103431054.68  59027383.90
##  indiv_squared_errors[28]  50192645.55  23090850.00  71239143.50  31652954.02
##  indiv_squared_errors[29]  44861919.81  21359450.00  62494171.64  28821566.09
##  indiv_squared_errors[30]  65305448.87  30979800.00  88313533.52  42124513.35
##          q5          q95 rhat ess_bulk ess_tail
##   185015.65 192964800.00 1.00     3994     3800
##   332318.55 232441500.00 1.00     3763     3948
##   151437.30 179330500.00 1.00     3908     3920
##  3520893.50 477504600.00 1.00     3867     3862
##   518801.35 345041650.00 1.00     4022     3929
##   179731.55 183623200.00 1.00     3652     3594
##   196414.25 181021150.00 1.00     3817     3833
##   250401.25 214644950.00 1.00     4184     3862
##  1905981.50 436553050.00 1.00     3908     3648
##   220220.20 197994300.00 1.00     3821     3711
##   206679.65 178921800.00 1.00     4087     3681
##   793983.15 361432100.00 1.00     4007     4041
##   739319.05 359587250.00 1.00     3840     3931
##   814086.60 380691900.00 1.00     3915     4098
##   396475.25 291147700.00 1.00     3921     3809
##   425825.50 288027650.00 1.00     3470     3891
##   254631.75 224471500.00 1.00     3953     4219
##   215775.65 246034150.00 1.00     4101     4117
##   232707.60 225380700.00 1.00     4139     3930
##   566316.45 307169100.00 1.00     4126     3837
##  1065379.50 400078800.00 1.00     3699     4003
##   603356.20 311294150.00 1.00     4030     3640
##  9564462.50 619565750.00 1.00     3831     3969
##  1389302.00 420191250.00 1.00     3934     3892
##   292879.70 216340750.00 1.00     3783     3735
##   193428.15 192896450.00 1.00     3928     3806
##   302639.05 296867550.00 1.00     3824     3636
##   179115.85 188128750.00 1.00     3630     3530
```

```
##   182135.00 171926900.00 1.00     4007     4013
##   291923.40 248573850.00 1.00     3749     3991
```

```
##        variable          mean        median           sd          mad
##  sum_of_squares 2520290672.00 2485835000.00 571683432.29 564744579.00
##              q5           q95 rhat ess_bulk ess_tail
##  1652053000.00 3512691500.00 1.00     3932     3866
```

```
##                 variable    mean  median      sd     mad      q5      q95 rhat
##  root_mean_squared_error 9106.79 9102.82 1037.47 1042.15 7420.81 10820.82 1.00
##  ess_bulk ess_tail
##      3932     3866
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Hierarchical model

```
##     variable     mean   median      sd     mad       q5      q95 rhat ess_bulk
##  y_tilde[1]  51112.21 51064.25 5943.57 5911.57 41395.96 61036.06 1.00     3650
##  y_tilde[2]  43257.83 43263.85 5877.93 5886.81 33783.93 52952.07 1.00     3832
##  y_tilde[3]  56916.08 56973.00 6058.32 6074.95 46823.38 67080.17 1.00     3949
##  y_tilde[4]  51090.66 51112.65 6149.63 6196.60 40974.47 61201.77 1.00     3953
##  y_tilde[5]  55247.26 55414.75 5950.88 5831.36 45531.06 65198.13 1.00     3903
##  y_tilde[6]  55569.41 55444.80 6055.23 6077.18 45618.52 65503.92 1.00     4173
##  y_tilde[7]  45313.25 45230.95 6100.62 6210.91 35266.04 55279.82 1.00     4187
```

```
##  y_tilde[8]  48379.53 48315.25 6110.50 6052.71 38476.89 58425.95 1.00     3660
##  y_tilde[9]  45150.24 45190.60 6029.81 6006.90 35306.37 54970.66 1.00     3870
##  y_tilde[10] 51098.43 51232.85 6028.20 6052.57 41168.35 60921.32 1.00     3649
##  y_tilde[11] 49826.86 49901.20 6198.63 6322.03 39587.35 59720.83 1.00     4034
##  y_tilde[12] 36322.42 36279.20 6101.42 6095.04 26402.69 46296.58 1.00     3757
##  y_tilde[13] 60619.02 60683.45 6208.02 6350.49 50317.02 70624.57 1.00     3967
##  y_tilde[14] 41079.85 41029.10 6136.02 6016.91 31137.34 51251.87 1.00     4132
##  y_tilde[15] 53230.22 53207.35 6088.84 6096.90 43273.56 63297.23 1.00     4014
##  y_tilde[16] 41408.90 41437.80 6107.00 6074.58 31215.29 51417.04 1.00     4033
##  y_tilde[17] 43032.18 43080.00 6020.37 5927.14 33103.73 53071.61 1.00     3893
##  y_tilde[18] 57795.06 57777.00 6022.75 6120.77 47927.65 67721.24 1.00     4048
##  y_tilde[19] 37149.37 37139.75 6056.47 6101.34 27039.23 47143.36 1.00     3883
##  y_tilde[20] 37346.31 37366.35 5940.98 5910.01 27710.19 46945.44 1.00     3993
##  y_tilde[21] 53145.16 53069.95 5912.19 5913.13 43807.66 63008.52 1.00     4013
##  y_tilde[22] 53632.20 53678.15 6065.32 5997.86 43623.17 63656.07 1.00     3888
##  y_tilde[23] 53402.86 53248.75 6073.09 6037.67 43423.56 63438.43 1.00     3546
##  y_tilde[24] 57179.82 57167.40 6030.92 5857.60 47305.37 67302.61 1.00     3954
##  y_tilde[25] 54230.75 54439.80 6092.34 6178.22 44200.32 63965.89 1.00     4091
##  y_tilde[26] 48857.29 48884.85 6028.02 5986.66 38999.97 58829.62 1.00     4175
##  y_tilde[27] 52292.52 52359.55 6082.11 6084.00 42095.18 61901.55 1.00     4178
##  y_tilde[28] 50607.51 50621.75 6040.72 6012.02 40526.69 60421.20 1.00     3767
##  y_tilde[29] 52536.05 52465.10 6051.26 6113.06 42522.15 62314.60 1.00     3924
##  y_tilde[30] 49007.47 49083.35 6043.93 6134.63 38992.21 58929.40 1.00     4137
##  ess_tail
##      3694
##      3891
##      3706
##      4008
##      3848
##      3666
##      4146
##      3812
##      3872
##      3875
##      4013
##      3797
##      4011
##      4009
##      3809
##      3851
##      3679
##      3851
##      3846
##      3932
##      3581
##      4012
##      4010
##      3799
##      3920
##      3721
##      3930
##      3889
##      3889
##      3722
```

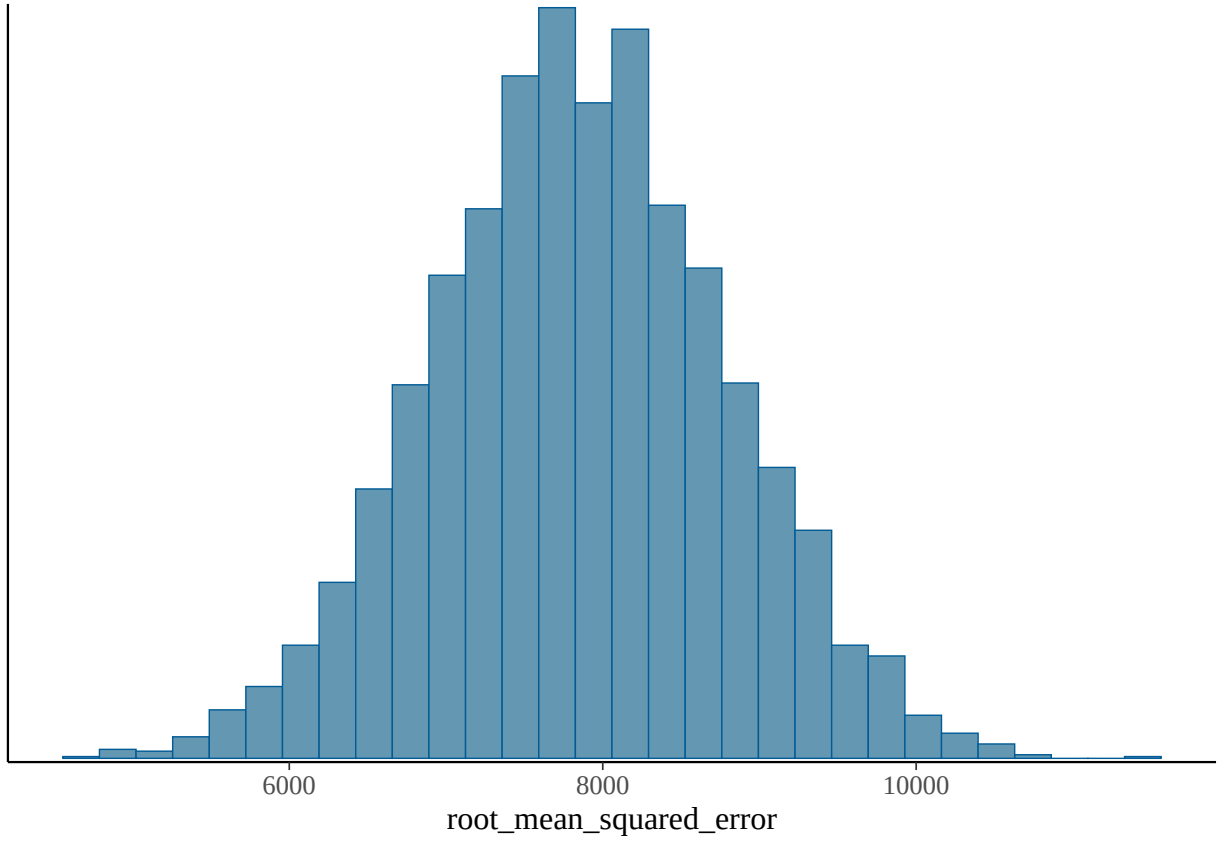
```
##                  variable         mean       median           sd          mad
##  indiv_squared_errors[1]   52616170.06  25498450.00  71230088.25  34417736.16
##  indiv_squared_errors[2]   35561145.61  16740000.00  50357276.37  22938579.64
##  indiv_squared_errors[3]   54207648.04  25763200.00  72744970.89  35332959.96
##  indiv_squared_errors[4]   70565082.78  36537400.00  88408692.99  49472152.93
##  indiv_squared_errors[5]   42015484.32  19133100.00  59055777.72  25971200.87
##  indiv_squared_errors[6]   46300235.29  21572700.00  64825558.45  29563696.34
##  indiv_squared_errors[7]   42959834.56  19669300.00  58711434.31  26786275.05
##  indiv_squared_errors[8]   55737509.07  26121500.00  76235015.67  35729599.94
##  indiv_squared_errors[9]   94402436.44  59689700.00 106205086.79  74978321.48
##  indiv_squared_errors[10]  36855154.05  16757550.00  51943973.72  22882381.68
##  indiv_squared_errors[11]  38446898.77  18045950.00  53506057.43  24694259.73
##  indiv_squared_errors[12]  55182197.59  26382000.00  74373014.67  35849015.96
##  indiv_squared_errors[13]  74159037.42  41100650.00  91110585.52  54702825.03
##  indiv_squared_errors[14]  75024088.13  41450450.00  92959828.69  53837075.59
##  indiv_squared_errors[15]  87904848.80  52683350.00 100616204.01  67965460.39
##  indiv_squared_errors[16]  51401925.25  23966550.00  71328008.08  32585398.23
##  indiv_squared_errors[17]  39408392.91  17734300.00  54601143.28  24376153.07
##  indiv_squared_errors[18]  41210322.87  19631050.00  56513545.28  26817772.88
##  indiv_squared_errors[19]  63884868.58  33180100.00  82847853.64  44456213.19
##  indiv_squared_errors[20]  41047866.48  19495950.00  57091604.03  26262124.06
##  indiv_squared_errors[21]  66786875.29  36967200.00  81376138.46  48903798.22
##  indiv_squared_errors[22]  78862751.12  45244000.00  92331641.48  59203991.22
##  indiv_squared_errors[23] 187135120.89 154063000.00 155393295.78 144154754.73
##  indiv_squared_errors[24]  95634814.49  59810150.00 107284702.29  74020524.82
##  indiv_squared_errors[25]  51537776.39  26233800.00  69823777.49  35219630.02
##  indiv_squared_errors[26]  38057992.00  17749850.00  54576044.75  23888844.69
##  indiv_squared_errors[27] 116629830.33  81269350.00 117888842.23  94561858.86
##  indiv_squared_errors[28]  40188667.23  17969000.00  56599385.47  24665816.05
##  indiv_squared_errors[29]  50701506.28  24305500.00  67278718.22  32939013.16
##  indiv_squared_errors[30]  50759454.53  23987850.00  68852163.87  32488732.71
##          q5          q95 rhat ess_bulk ess_tail
##   238936.15 198809100.00 1.00     3983     4143
##   129489.25 133010650.00 1.00     3712     3892
##   177733.80 204283350.00 1.00     4144     3687
##   337029.90 252414250.00 1.00     3781     3931
##   174640.40 166854450.00 1.00     4002     3889
##   189980.05 177048850.00 1.00     4095     3767
##   161429.55 160108550.00 1.00     3851     3950
##   236154.30 207544400.00 1.00     4011     4085
##   690115.55 304140350.00 1.00     3933     4014
##   127729.15 140933700.00 1.00     4121     3981
##   136510.50 147286650.00 1.00     3905     3927
##   229874.45 205355200.00 1.00     3831     3980
##   345516.05 255186600.00 1.00     3967     3837
##   362703.90 257816250.00 1.00     4196     3896
##   520669.55 295743600.00 1.00     3964     3908
##   198363.80 196025800.00 1.00     4055     4012
##   149465.45 155484100.00 1.00     3698     3926
##   181057.95 155428150.00 1.00     3971     3890
##   261303.50 235127950.00 1.00     3935     3887
##   199641.00 154220600.00 1.00     3991     3930
##   384938.45 224706800.00 1.00     4028     3579
##   482957.60 272616950.00 1.00     3979     4057
```

```
##  7247777.00 494502850.00 1.00     3529     3960
##   786202.50 317611000.00 1.00     3976     3784
##   211063.50 184848650.00 1.00     4083     3960
##   151003.90 146807150.00 1.00     3669     3872
##  1224838.50 343492750.00 1.00     4144     4013
##   140796.90 157767150.00 1.00     4177     3941
##   211826.25 189001000.00 1.00     4089     3769
##   214800.30 191698350.00 1.00     4123     3754
```

```
##        variable          mean        median           sd          mad
##  sum_of_squares 1875185925.25 1843625000.00 441807966.84 430309824.00
##              q5           q95 rhat ess_bulk ess_tail
##  1194632500.00 2640407500.00 1.00     4105     3814
```

```
##                 variable    mean  median     sd    mad      q5     q95 rhat
##  root_mean_squared_error 7850.65 7839.28 934.71 925.80 6310.40 9381.55 1.00
##  ess_bulk ess_tail
##      4105     3814
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## Saving 6.5 x 4.5 in image
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

10. Prior Sensitivity Analysis

To test whether our results are sensitive to prior choices, and to see how changing the priors affects our results, we ran all the models with both wider and narrower priors. The wide priors were obtained by multiplying each standard deviation by three, and the narrow priors were obtained by dividing the original priors by two.

For the pooled model, when scaling sigma values with 3, there were only minor changes in beta values. Also, when scaling sigma values with 0.5 the results stayed about the same except for beta_PRIVATE whose mean increased from around -7000 to around -5000. Based on these results it seems that pooled model is not that sensitive for prior changes that are this magnitude.

In conclusion, separate and hierarchical models are more sensitive for changes in priors compared to pooled model. This robustness could be explained by the fact that in pooled model the regions are considered as one and for that reason, there are more observations for that one entity compared to individual regions.

As the codes of the sensitivity analysis are long, they are included in the appendix.

Separate model sensitivity analysis

11. Discussion of Issues and Potential Improvements

During the project, the large amount of variables also posed challenges, as it was arguably rather slow and burdensome to find the most relevant variables to use in our analysis. It was somewhat surprising how fast the observation count started to shrink in data cleaning process, so in hindsight more attention could have been paid to cleanliness, as this dataset had for example a lot of missing values.

It was a shame we couldn't use a larger testing dataset because all data used for testing would be away from training the model. We tried to reduce the need to have a very long dataset by working very hard on removing independent variables. That work also ended up expanding our observation count from roughly 200 to over 700 observations, as we found variables with less null values and removed variables with too much missing data. For the feature selection, with more time we could have experimented with alternative methods, like the LASSO or Ridge regression.

As we were building a predictive model, we had to take certain ethical questions into account, especially to avoid building a discriminatory model. For example, if our model was used to by an employer to assess how well alumni from a certain school would perform in their career, it would be discriminatory to have a model that predicts lower performance for a university with say, high female or black student population as the real reason for differences in our data might be something else entirely than skin color or gender. In the end, this wasn't a big issue as for example the correlations between both gender vs income and share of white population vs income were very low and they would have been dropped from our model no matter what.

Another clear improvement would be to do more extensive predictive assessment of the models. We only assessed the pooled model, so both the separate model and hierarchical model could be assessed if we had more time and space. Furthermore, we only computed a simple RMSE metric. Perhaps it would be interesting to look at the probabilistic certainty of the predictions by plotting confidence regions, for example. The test set could also have been a bit bigger, as it is under 5% of the total number of available observations.

12. Conclusions

Like many previous studies, we found a strong link between education and income after school. The main aim of our project was to find a way to predict average income after studying at a university for any non-specialized university in the United States. We built three different multivariate models – separate, pooled and hierarchical – to predict income after school. All the models used the same set of numerical and categorical university-level independent variables.

As our initial dataset had almost 3000 features, feature selection was a very important and influential part of our project. We conducted feature selection in three phases: 1) We selected a subset of features based on

insights from previous studies and also used common sense to come up with additional features that could predict future earnings. 2) We assessed the relationship of the features with our dependent variable and accounted for any multicollinearity arising from mutually correlated features. We also visualized categorical variables and engineered new features to be added to the model. 3) We utilized stepwise regression in streamlining our model to include only the most significant features.

All our priors strived to be weakly informative but not too vague. We experimented with three different prior specifications: our conservative estimate for a good set of priors as well as two sets of priors set to be either wider or narrower than the conservative estimate. Separate and hierarchical models are more sensitive to changes in priors compared to pooled model. This robustness could be explained by the fact that in pooled model the regions are considered as one and for that reason, there are more observations for that one entity compared to individual regions.

We diagnosed convergence by calculating rhat, effective sample size and HMC specific convergence diagnostics for all models. None of the models converged perfectly according to all statistics.

Rhat statistics revealed that the separate model converged worse than the hierarchical or pooled model, both of which got good rhat convergence values for every beta in every chain. ESS diagnostics revealed that both the pooled model and the hierarchical model had autocorrelation issues. For the HMC specific convergence diagnostics, the separate model failed to produce results in reasonable running time. The separate model got satisfactory results from divergence diagnostics, E-BMFI, ESS and split rhat. The pooled model was overall satisfactory regarding these tests and the hierarchical model was also satisfactory save for a few hiccups when trying to obtain the HMC specific convergence diagnostics.

We ran posterior predictive checks to compare our data to simulated results. The visual comparison did not give any reason for concern and our models performed well on these checks. Additionally, we used LOO to perform model comparison. LOO suggested the separate model followed closely by the hierarchical model.

We got good k hat values for the separate model and the hierarchical model, which means the PSIS-LOO estimates for these models can be considered reliable. However, for pooled model there were many k hat values that were much greater than 0.7 . This indicates that the pooled model may be biased and the estimates might not be reliable.

As we have been building a predictive model, we though it would be important to perform predictive performance assessment. We use RMSE to assess the predictive ability of our model. Our data had been initially split into a training set and a test set. Due to limited space, we assessed only the pooled model. We consider our RMSE results for model accuracy to be satisfactory. Of course, our model is quite simple and there are several other variables which affect earnings and which we have been unable to include in the model.

13. Self-reflection

In hindsight taking on a task of creating three multivariate models was very ambitious, because multivariable models had not been covered very much during the course. That ambition paid off, however, and we think we have now a much stronger grasp of how the different models - pooled, separate and hierarchical - work and also know how to build multivariate bayesian linear models in general. We also learned a lot about feature selection, because the 3000 variables in the dataset forced us to create sensible procedures for selection. The combination of using logic and feature engireering tools like stepwise regression is a very powerful skill we developed while making this project.

I think this project made us think really hard about what do we want to show the reader to make our work as understandable as possible and we learned to visualize data and results in interesting ways and apply the visualization techniques covered on the course. Making the visualizations also aided our own thinking and certain visualizations gave valuable insight on our results and their quality.

Working with Stan is something none of us had done before this course. Stan is clearly a very powerful tool for statistical and data science use and we feel this project really ingrained basic Stan workflows and made working with Stan feel more of an efficient routine than an obstacle.

14. References

Card, D. (1999). THE CAUSAL EFFECT OF EDUCATION ON EARNINGS. Wolla, S. A., & Sullivan, J. (2017). Education, Income, and Wealth. https://fred.stlouisfed.org/graph/?g=7yKu.

Brewer, Dominic J., Eric R. Eide, and Ronald G. Ehrenberg. "Does it pay to attend an elite private college? Cross cohort evidence on the effects of college quality on earnings." (1996).

Card, David, and Alan B. Krueger. "Does school quality matter? Returns to education and the characteristics of public schools in the United States." Journal of political Economy 100.1 (1992): 1-40.

Data Commons. (2020). Gross domestic product per capita in United States of America [Graph]. Retrieved from https://datacommons.org/place/country/USA?utm_medium=explore&mprop=income&popt=Person&cpv=age%2CYears15Onwards&hl=en

Gelman, Andrew, and Jennifer Hill. Data analysis using regression and multilevel/hierarchical models. Cambridge university press, 2006.

Gabry, Gelman, Vehtari. (2016) Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC.

Number2. (2020). Average SAT Score [Blog Post]. Retrieved from: https://www.number2.com/average-sat-score/

Stan. (n.d) LOO package glossary [Website]. Retrieved from: ttps://mc-stan.org/loo/reference/loo-glossary.html

Stan. (n.d) Effective Sample Size [Website]. Retrieved from: https://mc-stan.org/docs/2_19/reference-manual/effective-sample-size-section.html

15. Appendix

Stepwise regression

```
data.stepwise <- select(data.joined.model, -c(MASTER, DOCTORAL))

model <- lm(MD_EARN_WNE_P10 ~ ., data = data.stepwise)

# step wise regression implied "best" model in terms of AIC
step(model, direction = "backward")

final.model <- lm(formula = MD_EARN_WNE_P10 ~ SAT_ALL + MD_FAMINC + COSTT4_A +
    POVERTY_RATE + URBAN + PRIVATE, data = data.stepwise)

summary(final.model)
```

Separate model Stan code (separate.stan)

```
print(separate.model)
```

```
## data {
##
##   int<lower=0> N; // number of observations in training set
##   int<lower=0> N_tilde; // number of observations in test set
##
##   int<lower=0> K; // number of regions
##   int<lower=1, upper=K> x[N]; // discrete group indicators in training set
```

```
##   int<lower=1, upper=K> x_tilde[N_tilde]; // discrete group indicators in test set
##
##   // data vectors
##   vector[N] SAT_ALL; // composite SAT score
##   vector[N] MD_FAMINIC; // median family income
##   vector[N] COSTT4_A; // cost of education
##   vector[N] POVERTY_RATE; // povery rate
##   vector[N] URBAN; // urban dummy
##   vector[N] PRIVATE; // private institution dummy
##   vector[N] y; // dependent variable (median earnings 10 years post-graduation)
##
##   // data vectors test set
##   vector[N_tilde] SAT_ALL_test; // SAT score
##   vector[N_tilde] MD_FAMINIC_test; // medain family income
##   vector[N_tilde] COSTT4_A_test; // average cost of education
##   vector[N_tilde] POVERTY_RATE_test; // poverty rate in school area
##   vector[N_tilde] URBAN_test; // urban dummy
##   vector[N_tilde] PRIVATE_test; // private innstitution dummy
##   vector[N_tilde] y_test; // median earnings
##
##   // prior distribution parameters
##   real pm_alpha; // prior mean of intercept term
##   real ps_alpha; // prior sd of intercept term
##   real pm_SAT_ALL;
##   real ps_SAT_ALL;
##   real pm_MD_FAMINC;
##   real ps_MD_FAMINC;
##   real pm_COSTT4_A;
##   real ps_COSTT4_A;
##   real pm_POVERTY_RATE;
##   real ps_POVERTY_RATE;
##   real pm_URBAN;
##   real ps_URBAN;
##   real pm_PRIVATE;
##   real ps_PRIVATE;
##   real pm_sigma;
##   real ps_sigma;
##
## }
##
## parameters {
##
##   // regression model parameters
##   vector[K] alpha;
##   vector[K] beta_SAT_ALL;
##   vector[K] beta_MD_FAMINIC;
##   vector[K] beta_COSTT4_A;
##   vector[K] beta_POVERTY_RATE;
##   vector[K] beta_URBAN;
##   vector[K] beta_PRIVATE;
##   vector<lower=0>[K] sigma;
##
## }
##
```

```
## model {
##
##   // priors
##   for (j in 1:K) {
##     alpha[j] ~ normal(pm_alpha, ps_alpha);
##     beta_SAT_ALL[j] ~ normal(pm_SAT_ALL, ps_SAT_ALL);
##     beta_MD_FAMINIC[j] ~ normal(pm_MD_FAMINC, ps_MD_FAMINC);
##     beta_COSTT4_A[j] ~ normal(pm_COSTT4_A, ps_COSTT4_A);
##     beta_POVERTY_RATE[j] ~ normal(pm_POVERTY_RATE, ps_POVERTY_RATE);
##     beta_URBAN[j] ~ normal(pm_URBAN, ps_URBAN);
##     beta_PRIVATE[j] ~ normal(pm_PRIVATE, ps_PRIVATE);
##     sigma[j] ~ normal(pm_sigma, ps_sigma);
##   }
##
##
##   // likelihoods
##   for (i in 1:N) {
##     y[i] ~ normal(alpha[x[i]] + beta_SAT_ALL[x[i]] * SAT_ALL[i] + beta_MD_FAMINIC[x[i]] *
##     MD_FAMINIC[i] + beta_COSTT4_A[x[i]] * COSTT4_A[i] +
##     beta_POVERTY_RATE[x[i]] * POVERTY_RATE[i] +
##     beta_URBAN[x[i]] * URBAN[i] + beta_PRIVATE[x[i]] * PRIVATE[i], sigma);
##   }
##
## }
##
## generated quantities {
##
##   vector[N] log_lik;
##   vector[N] y_rep;
##   vector[N_tilde] y_tilde;
##   vector[N_tilde] indiv_squared_errors;
##   real <lower = 0> sum_of_squares;
##   real <lower = 0> root_mean_squared_error;
##
##   for (i in 1:N) {
##     log_lik[i] = normal_lpdf(y[i] | alpha[x[i]] + beta_SAT_ALL[x[i]]
##     * SAT_ALL[i] + beta_MD_FAMINIC[x[i]] * MD_FAMINIC[i] + beta_COSTT4_A[x[i]] * COSTT4_A[i] + beta_
##     * POVERTY_RATE[i] + beta_URBAN[x[i]] * URBAN[i] +
##     beta_PRIVATE[x[i]] * PRIVATE[i], sigma[x[i]]);
##
##     y_rep[i] = normal_rng(alpha[x[i]] + beta_SAT_ALL[x[i]]
##     * SAT_ALL[i] + beta_MD_FAMINIC[x[i]] * MD_FAMINIC[i] + beta_COSTT4_A[x[i]] * COSTT4_A[i] + beta_
##     * POVERTY_RATE[i] + beta_URBAN[x[i]] * URBAN[i] +
##     beta_PRIVATE[x[i]] * PRIVATE[i], sigma[x[i]]);
##   }
##
##   for (i in 1:N_tilde) {
##     y_tilde[i] = normal_rng(alpha[x_tilde[i]] + beta_SAT_ALL[x_tilde[i]] * SAT_ALL_test[i] +
##     beta_MD_FAMINIC[x_tilde[i]] * MD_FAMINIC_test[i] + beta_COSTT4_A[x_tilde[i]] * COSTT4_A_test[i] +
##     beta_POVERTY_RATE[x_tilde[i]] * POVERTY_RATE_test[i] + beta_URBAN[x_tilde[i]] * URBAN_test[i] +
##     beta_PRIVATE[x_tilde[i]] * PRIVATE_test[i], sigma[x_tilde[i]]);
##
##     indiv_squared_errors[i] = (y_test[i] - y_tilde[i])^2;
##  }
```

```
##
##  sum_of_squares = sum(indiv_squared_errors);
##  root_mean_squared_error = sqrt(sum_of_squares / N_tilde);
##
## }
```

Pooled model Stan code (pooled.stan)

```
print(pooled.model)
```

```
## data {
##
##   int<lower=0> N; // number of observations
##   int<lower=0> N_tilde; // number of observations in test set
##
##   // data vectors
##   vector[N] SAT_ALL;
##   vector[N] MD_FAMINIC;
##   vector[N] COSTT4_A;
##   vector[N] POVERTY_RATE;
##   vector[N] URBAN;
##   vector[N] PRIVATE;
##   vector[N] y;
##
##   // data vectors test set
##   vector[N_tilde] SAT_ALL_test; // SAT score
##   vector[N_tilde] MD_FAMINIC_test; // medain family income
##   vector[N_tilde] COSTT4_A_test; // average cost of education
##   vector[N_tilde] POVERTY_RATE_test; // poverty rate in school area
##   vector[N_tilde] URBAN_test; // urban dummy
##   vector[N_tilde] PRIVATE_test; // private innstitution dummy
##   vector[N_tilde] y_test; // median earnings
##
##   // prior distribution parameters
##   real pm_alpha; // prior mean of intercept term
##   real ps_alpha; // prior sd of intercept term
##   real pm_SAT_ALL;
##   real ps_SAT_ALL;
##   real pm_MD_FAMINC;
##   real ps_MD_FAMINC;
##   real pm_COSTT4_A;
##   real ps_COSTT4_A;
##   real pm_POVERTY_RATE;
##   real ps_POVERTY_RATE;
##   real pm_URBAN;
##   real ps_URBAN;
##   real pm_PRIVATE;
##   real ps_PRIVATE;
##   real pm_sigma;
##   real ps_sigma;
##
## }
##
## parameters {
```

```
##   real alpha;
##   real beta_SAT_ALL;
##   real beta_MD_FAMINIC;
##   real beta_COSTT4_A;
##   real beta_POVERTY_RATE;
##   real beta_URBAN;
##   real beta_PRIVATE;
##   real<lower=0> sigma;
##
## }
##
## model {
##
##   // weakly informative priors
##   alpha ~ normal(pm_alpha, ps_alpha);
##   beta_SAT_ALL ~ normal(pm_SAT_ALL, ps_SAT_ALL);
##   beta_MD_FAMINIC ~ normal(pm_MD_FAMINC, ps_MD_FAMINC);
##   beta_COSTT4_A ~ normal(pm_COSTT4_A, ps_COSTT4_A);
##   beta_POVERTY_RATE ~ normal(pm_POVERTY_RATE, ps_POVERTY_RATE);
##   beta_URBAN ~ normal(pm_URBAN, ps_URBAN);
##   beta_PRIVATE ~ normal(pm_PRIVATE, ps_PRIVATE);
##   sigma ~ normal(pm_sigma, ps_sigma);
##
##   y ~ normal(alpha + beta_SAT_ALL * SAT_ALL + beta_MD_FAMINIC * MD_FAMINIC +
##    beta_COSTT4_A * COSTT4_A +
##   beta_POVERTY_RATE * POVERTY_RATE + beta_URBAN * URBAN +
##   beta_PRIVATE * PRIVATE, sigma);
## }
##
## generated quantities {
##   vector[N] log_lik;
##   vector[N] y_rep;
##   vector[N_tilde] y_tilde;
##   vector[N_tilde] indiv_squared_errors;
##   real <lower = 0> sum_of_squares;
##   real <lower = 0> root_mean_squared_error;
##
##   for (i in 1:N) {
##     log_lik[i] = normal_lpdf(y[i] | alpha + beta_SAT_ALL * SAT_ALL
##     + beta_MD_FAMINIC * MD_FAMINIC +
##     beta_COSTT4_A * COSTT4_A + beta_POVERTY_RATE * POVERTY_RATE +
##     beta_URBAN * URBAN + beta_PRIVATE * PRIVATE, sigma);
##
##     y_rep[i] = normal_rng(alpha + beta_SAT_ALL * SAT_ALL[i]
##     + beta_MD_FAMINIC * MD_FAMINIC[i] +
##     beta_COSTT4_A * COSTT4_A[i] + beta_POVERTY_RATE * POVERTY_RATE[i]
##     + beta_URBAN * URBAN[i] + beta_PRIVATE * PRIVATE[i], sigma);
##   }
##
##   for (i in 1:N_tilde) {
##    y_tilde[i] = normal_rng(alpha + beta_SAT_ALL * SAT_ALL_test[i]
##     + beta_MD_FAMINIC * MD_FAMINIC_test[i]  + beta_COSTT4_A * COSTT4_A_test[i] + beta_POVERTY_RATE *
##     POVERTY_RATE_test[i] + beta_URBAN * URBAN_test[i] +
##     beta_PRIVATE * PRIVATE_test[i], sigma);
```

```
##
##    indiv_squared_errors[i] = (y_test[i] - y_tilde[i])^2;
##
##  }
##
##  sum_of_squares = sum(indiv_squared_errors);
##  root_mean_squared_error = sqrt(sum_of_squares / N_tilde);
##
## }
```

Hierarchical model Stan code (hierarchical.stan)

```
print(hierarchical.model)
```

```
## data {
##
##   int<lower=0> N; // number of observations in training set
##   int<lower=0> N_tilde; // number of observations in test set
##
##   int<lower=0> K; // number of regions
##   int<lower=1, upper=K> x[N]; // discrete group indicators in training set
##   int<lower=1, upper=K> x_tilde[N_tilde]; // discrete group indicators in test set
##
##   // data vectors training set
##   vector[N] SAT_ALL; // SAT score
##   vector[N] MD_FAMINIC; // medain family income
##   vector[N] COSTT4_A; // average cost of education
##   vector[N] POVERTY_RATE; // poverty rate in school area
##   vector[N] URBAN; // urban dummy
##   vector[N] PRIVATE; // private innstitution dummy
##   vector[N] y; // median earnings
##
##   // data vectors test set
##   vector[N_tilde] SAT_ALL_test; // SAT score
##   vector[N_tilde] MD_FAMINIC_test; // medain family income
##   vector[N_tilde] COSTT4_A_test; // average cost of education
##   vector[N_tilde] POVERTY_RATE_test; // poverty rate in school area
##   vector[N_tilde] URBAN_test; // urban dummy
##   vector[N_tilde] PRIVATE_test; // private innstitution dummy
##   vector[N_tilde] y_test; // median earnings
##
##   // params for prior distributions
##   real pm_alpha; // prior mean of intercept term
##   real ps_alpha; // prior sd of intercept term
##   real pm_s_alpha; // sd of prior mean of intercept term
##   real ps_s_alpha; // sd of prior sd of intercept term
##
##   real pm_SAT_ALL;
##   real ps_SAT_ALL;
##   real pm_s_SAT_ALL;
##   real ps_s_SAT_ALL;
##
##   real pm_MD_FAMINC;
##   real ps_MD_FAMINC;
```

```
##   real pm_s_MD_FAMINC;
##   real ps_s_MD_FAMINC;
##
##   real pm_COSTT4_A;
##   real ps_COSTT4_A;
##   real pm_s_COSTT4_A;
##   real ps_s_COSTT4_A;
##
##   real pm_POVERTY_RATE;
##   real ps_POVERTY_RATE;
##   real pm_s_POVERTY_RATE;
##   real ps_s_POVERTY_RATE;
##
##   real pm_URBAN;
##   real ps_URBAN;
##   real pm_s_URBAN;
##   real ps_s_URBAN;
##
##   real pm_PRIVATE;
##   real ps_PRIVATE;
##   real pm_s_PRIVATE;
##   real ps_s_PRIVATE;
##
##   real pm_sigma;
##   real ps_sigma;
##   real pm_s_sigma;
##   real ps_s_sigma;
##
## }
##
## parameters {
##
##   vector[K] alpha;
##   vector[K] beta_SAT_ALL;
##   vector[K] beta_MD_FAMINIC;
##   vector[K] beta_COSTT4_A;
##   vector[K] beta_POVERTY_RATE;
##   vector[K] beta_URBAN;
##   vector[K] beta_PRIVATE;
##   real<lower=0> sigma; // common standard deviation
##
##   real mu_alpha;
##   real<lower=0> sigma_alpha;
##
##   real mu_SAT_ALL;
##   real<lower=0> sigma_SAT_ALL;
##
##   real mu_MD_FAMINIC;
##   real<lower=0> sigma_MD_FAMINIC;
##
##   real mu_COSTT4_A;
##   real<lower=0> sigma_COSTT4_A;
##
##   real mu_POVERTY_RATE;
```

```
##   real<lower=0> sigma_POVERTY_RATE;
##
##   real mu_URBAN;
##   real<lower=0> sigma_URBAN;
##
##   real mu_PRIVATE;
##   real<lower=0> sigma_PRIVATE;
##
##   real mu_sigma;
##   real<lower=0> sigma_sigma;
##
## }
##
## model {
##
##   mu_alpha ~ normal(pm_alpha, pm_s_alpha);
##   sigma_alpha ~ normal(ps_alpha, ps_s_alpha);
##   alpha ~ normal(mu_alpha, sigma_alpha);
##
##   mu_SAT_ALL ~  normal(pm_SAT_ALL, pm_s_SAT_ALL);
##   sigma_SAT_ALL ~ normal(ps_SAT_ALL, ps_s_SAT_ALL);
##   beta_SAT_ALL ~ normal(mu_SAT_ALL, sigma_SAT_ALL);
##
##   mu_MD_FAMINIC ~  normal(pm_MD_FAMINC, pm_s_MD_FAMINC);
##   sigma_MD_FAMINIC ~ normal(ps_MD_FAMINC, ps_s_MD_FAMINC);
##   beta_MD_FAMINIC ~ normal(mu_MD_FAMINIC, sigma_MD_FAMINIC);
##
##   mu_COSTT4_A ~  normal(pm_COSTT4_A, pm_s_COSTT4_A);
##   sigma_COSTT4_A ~ normal(ps_COSTT4_A, ps_s_COSTT4_A);
##   beta_COSTT4_A ~  normal(mu_COSTT4_A, sigma_COSTT4_A);
##
##   mu_POVERTY_RATE ~  normal(pm_POVERTY_RATE, pm_s_POVERTY_RATE);
##   sigma_POVERTY_RATE ~ normal(ps_POVERTY_RATE, ps_s_POVERTY_RATE);
##   beta_POVERTY_RATE ~ normal(mu_POVERTY_RATE, sigma_POVERTY_RATE);
##
##   mu_URBAN ~  normal(pm_URBAN, pm_s_URBAN);
##   sigma_URBAN ~ normal(ps_URBAN, ps_s_URBAN);
##   beta_URBAN ~ normal(mu_URBAN, sigma_URBAN);
##
##   mu_PRIVATE ~  normal(pm_PRIVATE, pm_s_PRIVATE);
##   sigma_PRIVATE ~ normal(ps_PRIVATE, ps_s_PRIVATE);
##   beta_PRIVATE ~ normal(mu_PRIVATE, sigma_PRIVATE);
##
##   mu_sigma ~ normal(pm_sigma, pm_s_sigma);
##   sigma_sigma ~ normal(ps_sigma, ps_s_sigma);
##   sigma ~ normal(mu_sigma, sigma_sigma);
##
##
##   // likelihoods
##   for (i in 1:N) {
##     y[i] ~ normal(alpha[x[i]] + beta_SAT_ALL[x[i]] * SAT_ALL[i] + beta_MD_FAMINIC[x[i]] * MD_FAMINIC
##     + beta_POVERTY_RATE[x[i]] * POVERTY_RATE[i] + beta_URBAN[x[i]] * URBAN[i] + beta_PRIVATE[x[i]] *
##   }
##
```

```
## }
##
## generated quantities {
##
##   vector[N] log_lik;
##   vector[N] y_rep;
##   vector[N_tilde] y_tilde;
##   vector[N_tilde] indiv_squared_errors;
##   real <lower = 0> sum_of_squares;
##   real <lower = 0> root_mean_squared_error;
##
##   for (i in 1:N) {
##
##     log_lik[i] = normal_lpdf(y[i] | alpha[x[i]] + beta_SAT_ALL[x[i]] *
##     SAT_ALL[i] + beta_MD_FAMINIC[x[i]] * MD_FAMINIC[i] + beta_COSTT4_A[x[i]] * COSTT4_A[i] + beta_PO
##     POVERTY_RATE[i] + beta_URBAN[x[i]] * URBAN[i] +
##     beta_PRIVATE[x[i]] * PRIVATE[i], sigma);
##
##     y_rep[i] = normal_rng(alpha[x[i]] + beta_SAT_ALL[x[i]] * SAT_ALL[i]
##     + beta_MD_FAMINIC[x[i]] * MD_FAMINIC[i]  + beta_COSTT4_A[x[i]] * COSTT4_A[i] + beta_POVERTY_RATE
##     POVERTY_RATE[i] + beta_URBAN[x[i]] * URBAN[i] +
##     beta_PRIVATE[x[i]] * PRIVATE[i], sigma);
##  }
##
##  for (i in 1:N_tilde) {
##    y_tilde[i] = normal_rng(alpha[x_tilde[i]] + beta_SAT_ALL[x_tilde[i]] * SAT_ALL_test[i]
##     + beta_MD_FAMINIC[x_tilde[i]] * MD_FAMINIC_test[i]  + beta_COSTT4_A[x_tilde[i]] * COSTT4_A_test[
##     POVERTY_RATE_test[i] + beta_URBAN[x_tilde[i]] * URBAN_test[i] +
##     beta_PRIVATE[x_tilde[i]] * PRIVATE_test[i], sigma);
##
##    indiv_squared_errors[i] = (y_test[i] - y_tilde[i])^2;
##
##  }
##
##  sum_of_squares = sum(indiv_squared_errors);
##  root_mean_squared_error = sqrt(sum_of_squares / N_tilde);
##
## }
```

Model fitting

Separate model fit

```
separate.model <- cmdstan_model(stan_file = "./Stan/separate.stan")

separate.model.data <- list(N = nrow(data.joined.stan),
                            K = length(unique(data.joined.stan$REGION)),
                            x = data.joined.stan$REGION,

                            SAT_ALL = data.joined.stan$SAT_ALL,
                            MD_FAMINIC = data.joined.stan$MD_FAMINC,
                            COSTT4_A = data.joined.stan$COSTT4_A,
                            POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                            URBAN = data.joined.stan$URBAN,
```

```
                            PRIVATE = data.joined.stan$PRIVATE,
                            y = data.joined.stan$MD_EARN_WNE_P10,

                            pm_alpha = 0,
                            ps_alpha = 10000,
                            pm_SAT_ALL = 50,
                            ps_SAT_ALL = 500,
                            pm_MD_FAMINC = 0,
                            ps_MD_FAMINC = 100,
                            pm_COSTT4_A = 0,
                            ps_COSTT4_A = 500,
                            pm_POVERTY_RATE = 0,
                            ps_POVERTY_RATE = 2500,
                            pm_URBAN = 0,
                            ps_URBAN = 2500,
                            pm_PRIVATE = 0,
                            ps_PRIVATE = 2500,
                            pm_sigma = 10000,
                            ps_sigma = 10000

)


separate.fit <- separate.model$sample(data = separate.model.data, seed = 1234, refresh = 1e3)

separate.fit$summary()
```

Pooled model fit

```
pooled.model <- cmdstan_model(stan_file = "./Stan/pooled.stan")

pooled.model.data <- list(N = nrow(data.joined.stan),

                          SAT_ALL = data.joined.stan$SAT_ALL,
                          MD_FAMINIC = data.joined.stan$MD_FAMINC,
                          COSTT4_A = data.joined.stan$COSTT4_A,
                          POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                          URBAN = data.joined.stan$URBAN,
                          PRIVATE = data.joined.stan$PRIVATE,
                          y = data.joined.stan$MD_EARN_WNE_P10,

                          pm_alpha = 0,
                          ps_alpha = 10000,
                          pm_SAT_ALL = 50,
                          ps_SAT_ALL = 500,
                          pm_MD_FAMINC = 0,
                          ps_MD_FAMINC = 100,
                          pm_COSTT4_A = 0,
                          ps_COSTT4_A = 500,
                          pm_POVERTY_RATE = 0,
                          ps_POVERTY_RATE = 2500,
                          pm_URBAN = 0,
                          ps_URBAN = 2500,
```

```
                            pm_PRIVATE = 0,
                            ps_PRIVATE = 2500,
                            pm_sigma = 10000,
                            ps_sigma = 10000

                            )


pooled.fit <- pooled.model$sample(data = pooled.model.data, seed = 1234, refresh = 1e3)

pooled.fit$summary()
```

Hierarchical model fit

Summary of model fit for main parameters:

```
hierarchical.model <- cmdstan_model(stan_file = "./Stan/hierarchical.stan")

hierarchical.model.data <- list(N = nrow(data.joined.stan),
                                K = length(unique(data.joined.stan$REGION)),
                                x = data.joined.stan$REGION,

                                SAT_ALL = data.joined.stan$SAT_ALL,
                                MD_FAMINIC = data.joined.stan$MD_FAMINC,
                                COSTT4_A = data.joined.stan$COSTT4_A,
                                POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                                URBAN = data.joined.stan$URBAN,
                                PRIVATE = data.joined.stan$PRIVATE,
                                y = data.joined.stan$MD_EARN_WNE_P10,

                                pm_alpha = 0,
                                ps_alpha = 10000,
                                pm_s_alpha = 100,
                                ps_s_alpha = 100,

                                pm_SAT_ALL = 43,
                                ps_SAT_ALL = 500,
                                pm_s_SAT_ALL = 100,
                                ps_s_SAT_ALL = 100,

                                pm_MD_FAMINC = 0,
                                ps_MD_FAMINC = 100,
                                pm_s_MD_FAMINC = 10,
                                ps_s_MD_FAMINC = 100,

                                pm_COSTT4_A = 0,
                                ps_COSTT4_A = 500,
                                pm_s_COSTT4_A = 50,
                                ps_s_COSTT4_A = 500,

                                pm_POVERTY_RATE = 0,
                                ps_POVERTY_RATE = 2500,
                                pm_s_POVERTY_RATE = 500,
                                ps_s_POVERTY_RATE = 500,
```

```
                                  pm_URBAN = 0,
                                  ps_URBAN = 2500,
                                  pm_s_URBAN = 500,
                                  ps_s_URBAN = 500,

                                  pm_PRIVATE = 0,
                                  ps_PRIVATE = 2500,
                                  pm_s_PRIVATE = 500,
                                  ps_s_PRIVATE = 500,

                                  pm_sigma = 10000,
                                  ps_sigma = 10000,
                                  pm_s_sigma = 500,
                                  ps_s_sigma = 500)

hierarchical.fit <- hierarchical.model$sample(data = hierarchical.model.data,
                                              seed = 1234, refresh = 1e3)

hierarchical.fit$summary()
```

Separate model Rhat

```
rhat.df <- tibble()

params <- c("alpha[1]", "beta_SAT_ALL[1]", "beta_MD_FAMINIC[1]", "beta_COSTT4_A[1]",
            "beta_POVERTY_RATE[1]", "beta_URBAN[1]", "beta_PRIVATE[1]")

for (param in params) {
  rhats <- extract_variable_matrix(separate.fit$draws(), variable = param) %>% apply(2, rhat)
  row <- tibble("Parameter" = param,
                "Chain 1" = rhats[1],
                "Chain 2" = rhats[2],
                "Chain 3" = rhats[3],
                "Chain 4" = rhats[4])
  rhat.df <- rbind(rhat.df, row)
}

rhat.df
```

Pooled model Rhat

```
params <- pooled.model$variables()$parameters %>% names()
rhat.df <- tibble()
for (param in params) {
  rhats <- extract_variable_matrix(pooled.fit$draws(), variable = param) %>% apply(2, rhat)
  row <- tibble("Parameter" = param,
                "Chain 1" = rhats[1],
                "Chain 2" = rhats[2],
                "Chain 3" = rhats[3],
                "Chain 4" = rhats[4])
  rhat.df <- rbind(rhat.df, row)
}
```

```
rhat.df
```

Hierarchical model Rhat

```
rhat.df <- tibble()

params <- c("alpha[1]", "beta_SAT_ALL[1]", "beta_MD_FAMINIC[1]", "beta_COSTT4_A[1]",
            "beta_POVERTY_RATE[1]", "beta_URBAN[1]", "beta_PRIVATE[1]")

for (param in params) {
  rhats <- extract_variable_matrix(hierarchical.fit$draws(), variable = param) %>% apply(2, rhat)
  row <- tibble("Parameter" = param,
                "Chain 1" = rhats[1],
                "Chain 2" = rhats[2],
                "Chain 3" = rhats[3],
                "Chain 4" = rhats[4])
  rhat.df <- rbind(rhat.df, row)
}

rhat.df
```

HMC diagnostics output

```
separate.fit$cmdstan_diagnose()
```

```
## Processing csv files: /tmp/RtmpV5LLDy/separate-202212152133-1-0c0758.csv, /tmp/RtmpV5LLDy/separate-2
##
## Checking sampler transitions treedepth.
## 4000 of 4000 (100.00%) transitions hit the maximum treedepth limit of 10, or 2^10 leapfrog steps.
## Trajectories that are prematurely terminated due to this limit will result in slow exploration.
## For optimal performance, increase this limit.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## The following parameters had split R-hat greater than 1.05:
##   beta_SAT_ALL[9], beta_MD_FAMINIC[9], beta_COSTT4_A[9], beta_POVERTY_RATE[9]
## Such high values indicate incomplete mixing and biased estimation.
## You should consider regularizating your model with additional prior information or a more effective p
##
## Processing complete.
```

```
pooled.fit$cmdstan_diagnose()
```

```
## Processing csv files: /tmp/RtmpV5LLDy/pooled-202212152239-1-2d8304.csv, /tmp/RtmpV5LLDy/pooled-20221
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
```

```
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete, no problems detected.
```

```
hierarchical.fit$cmdstan_diagnose()
```

```
## Processing csv files: /tmp/RtmpV5LLDy/hierarchical-202212152241-1-11b2a8.csv, /tmp/RtmpV5LLDy/hierarc
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## 76 of 4000 (1.90%) transitions ended with a divergence.
## These divergent transitions indicate that HMC is not fully able to explore the posterior distributio
## Try increasing adapt delta closer to 1.
## If this doesn't remove all divergences, try to reparameterize the model.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete.
```

K hat

```
pareto_k_table(separate.fit$loo())
```

```
##
## All Pareto k estimates are good (k < 0.5).
```

```
pareto_k_table(pooled.fit$loo())
```

```
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)       0     0.0%  <NA>
##  (0.5, 0.7]   (ok)         0     0.0%  <NA>
##    (0.7, 1]   (bad)        0     0.0%  <NA>
##    (1, Inf)   (very bad) 766   100.0%  0
```

```
pareto_k_table(hierarchical.fit$loo())
```

```
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     759   99.1%   507
##  (0.5, 0.7]   (ok)         6    0.8%   280
```

```
##    (0.7, 1]   (bad)        1    0.1%   90
##    (1, Inf)   (very bad)   0    0.0%   <NA>
```

RMSE

```
pooled.fit.coeff.df <- pooled.fit$summary() %>% slice(2:8)

alpha.hat <- pooled.fit.coeff.df %>% head(1) %>% select(median)
beta.hat <- pooled.fit.coeff.df %>% tail(-1) %>% select(median) %>% as.matrix()

test.X <- data.joined.stan.test %>%
  select(SAT_ALL, MD_FAMINC, AGE_ENTRY, COSTT4_A, POVERTY_RATE, PRIVATE) %>%
  as.matrix()

N <- nrow(data.joined.stan.test)
y.hat <- numeric(N)
y <- data.joined.stan.test$MD_EARN_WNE_P10
se <- numeric(N)
for (i in 1:N) {

  y.hat[i] <- alpha.hat+beta.hat%*%test.X[i,]
  se[i] <- (y.hat[[i]]-y[i])^2

}

rmse <- sqrt(mean(se))
```

Test set

```
data.joined.stan.test %>% print(n=30)
```

```
## # A tibble: 30 x 11
##    REGION MD_EAR~1 SAT_ALL MD_FA~2 AGE_E~3 COSTT~4 POVER~5 URBAN PRIVATE DOCTO~6
##     <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
##  1      2    46953     520.  45776.    22.3   45458    4.86     1       1       0
##  2      5    42248     526.  36486.    25.1   45333    8.44     0       1       1
##  3      3    61101     600   70982     20.0   28626    6.10     1       0       1
##  4      8    56814     515   27217     23.2   21576   10.1      1       0       0
##  5      5    52676     611   54206     20.8   32010    8.28     1       0       1
##  6      8    52464     570   34261     22.7   18756    7.25     1       0       1
##  7      3    42915     485   58500     21.4   46420    5.23     0       1       0
##  8      2    44089     515   56548     20.3   28442    7.80     0       0       1
##  9      6    37531     514   36206.    22.8   44585    7.19     1       1       0
## 10      2    51823     508.  37962     21.4   50245    6.85     1       1       0
## 11      3    49644     552.  57838     21.4   25078    6.14     0       0       1
## 12      5    32084     474.  22080.    20.1   21892   17.0      1       0       0
## 13      3    54650     632.  79777     20.8   33805    5.07     0       0       1
## 14      6    47194     500   36174.    22.9   31865    9.86     0       1       0
## 15      1    46100     536.  44704     23.1   50959    6.5      1       1       1
## 16      6    45166     498.  32295     29.8   35240   11.0      0       1       0
## 17      5    41251     525   41738     24.0   39983    5.39     0       1       0
## 18      2    60019     565   68011     21.3   29130    4.69     1       0       0
## 19      5    42366     502.  30172.    30.0   32055   10.7      0       1       0
```

```
## 20      5    34946    485   21266     23.3   24570   19.1      1        0        1
## 21      8    58788    525   40508     22.0   26537    6.23     1        0        0
## 22      3    47145    575   58609     23.7   29534    7.82     0        0        1
## 23      2    65661    538.  43360.    21.8   29130    6.02     1        0        1
## 24      3    49481    612.  57822.    22.3   26892    7.11     1        0        1
## 25      7    50432    600   42950.    23.2   19886    7.39     1        0        1
## 26      3    47542    548.  45002     22.1   21127    6.20     1        0        1
## 27      3    43368    612.  49012     21.0   27793    5.74     0        1        0
## 28      3    52533    537   65294.    21.8   42869    6.55     1        1        0
## 29      8    48782    522.  42880     21.8   48086    7.83     1        1        0
## 30      3    52781    530   38451     21.7   43228    6.93     1        1        0
## # ... with 1 more variable: MASTER <dbl>, and abbreviated variable names
## #   1: MD_EARN_WNE_P10, 2: MD_FAMINC, 3: AGE_ENTRY, 4: COSTT4_A,
## #   5: POVERTY_RATE, 6: DOCTORAL
## # i Use `colnames()` to see all variable names
```

Separate model sensitivity analysis

Summary of model fit for main parameters with wide priors:

```
separate.model <- cmdstan_model(stan_file = "./Stan/separate.stan")

scale_param <- 3

separate.model.data <- list(N = nrow(data.joined.stan),
                            K = length(unique(data.joined.stan$REGION)),
                            x = data.joined.stan$REGION,

                            SAT_ALL = data.joined.stan$SAT_ALL,
                            MD_FAMINIC = data.joined.stan$MD_FAMINC,
                            COSTT4_A = data.joined.stan$COSTT4_A,
                            POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                            URBAN = data.joined.stan$URBAN,
                            PRIVATE = data.joined.stan$PRIVATE,
                            y = data.joined.stan$MD_EARN_WNE_P10,

                            pm_alpha = 0,
                            ps_alpha = 10000 * scale_param,
                            pm_SAT_ALL = 50,
                            ps_SAT_ALL = 500 * scale_param,
                            pm_MD_FAMINC = 0,
                            ps_MD_FAMINC = 100 * scale_param,
                            pm_COSTT4_A = 0,
                            ps_COSTT4_A = 500 * scale_param,
                            pm_POVERTY_RATE = 0,
                            ps_POVERTY_RATE = 2500 * scale_param,
                            pm_URBAN = 0,
                            ps_URBAN = 2500 * scale_param,
                            pm_PRIVATE = 0,
                            ps_PRIVATE = 2500 * scale_param,
                            pm_sigma = 10000 * scale_param,
                            ps_sigma = 10000 * scale_param

)
```

```
separate.fit <- separate.model$sample(data = separate.model.data, seed = 1234, refresh = 1e3)

separate.fit$summary()
```

Summary of model fit for main parameters with narrow priors:

```
separate.model <- cmdstan_model(stan_file = "./Stan/separate.stan")

scale_param <- 0.5

separate.model.data <- list(N = nrow(data.joined.stan),
                            K = length(unique(data.joined.stan$REGION)),
                            x = data.joined.stan$REGION,

                            SAT_ALL = data.joined.stan$SAT_ALL,
                            MD_FAMINIC = data.joined.stan$MD_FAMINC,
                            COSTT4_A = data.joined.stan$COSTT4_A,
                            POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                            URBAN = data.joined.stan$URBAN,
                            PRIVATE = data.joined.stan$PRIVATE,
                            y = data.joined.stan$MD_EARN_WNE_P10,

                            pm_alpha = 0,
                            ps_alpha = 10000 * scale_param,
                            pm_SAT_ALL = 50,
                            ps_SAT_ALL = 500 * scale_param,
                            pm_MD_FAMINC = 0,
                            ps_MD_FAMINC = 100 * scale_param,
                            pm_COSTT4_A = 0,
                            ps_COSTT4_A = 500 * scale_param,
                            pm_POVERTY_RATE = 0,
                            ps_POVERTY_RATE = 2500 * scale_param,
                            pm_URBAN = 0,
                            ps_URBAN = 2500 * scale_param,
                            pm_PRIVATE = 0,
                            ps_PRIVATE = 2500 * scale_param,
                            pm_sigma = 10000 * scale_param,
                            ps_sigma = 10000 * scale_param

)


separate.fit <- separate.model$sample(data = separate.model.data, seed = 1234, refresh = 1e3)

separate.fit$summary()
```

Pooled model sensitivity analysis

Summary of model fit for main parameters with wide priors:

```
pooled.model <- cmdstan_model(stan_file = "./Stan/pooled.stan")

scale_param <- 3
```

```
pooled.model.data <- list(N = nrow(data.joined.stan),

                          SAT_ALL = data.joined.stan$SAT_ALL,
                          MD_FAMINIC = data.joined.stan$MD_FAMINC,
                          COSTT4_A = data.joined.stan$COSTT4_A,
                          POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                          URBAN = data.joined.stan$URBAN,
                          PRIVATE = data.joined.stan$PRIVATE,
                          y = data.joined.stan$MD_EARN_WNE_P10,

                          pm_alpha = 0,
                          ps_alpha = 10000 * scale_param,
                          pm_SAT_ALL = 50,
                          ps_SAT_ALL = 500 * scale_param,
                          pm_MD_FAMINC = 0,
                          ps_MD_FAMINC = 100 * scale_param,
                          pm_COSTT4_A = 0,
                          ps_COSTT4_A = 500 * scale_param,
                          pm_POVERTY_RATE = 0,
                          ps_POVERTY_RATE = 2500 * scale_param,
                          pm_URBAN = 0,
                          ps_URBAN = 2500,
                          pm_PRIVATE = 0,
                          ps_PRIVATE = 2500 * scale_param,
                          pm_sigma = 10000 * scale_param,
                          ps_sigma = 10000 * scale_param

                          )


pooled.fit <- pooled.model$sample(data = pooled.model.data, seed = 1234, refresh = 1e3)

pooled.fit$summary()
```

Summary of model fit for main parameters with narrow priors:

```
pooled.model <- cmdstan_model(stan_file = "./Stan/pooled.stan")

scale_param <- 0.5

pooled.model.data <- list(N = nrow(data.joined.stan),

                          SAT_ALL = data.joined.stan$SAT_ALL,
                          MD_FAMINIC = data.joined.stan$MD_FAMINC,
                          COSTT4_A = data.joined.stan$COSTT4_A,
                          POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                          URBAN = data.joined.stan$URBAN,
                          PRIVATE = data.joined.stan$PRIVATE,
                          y = data.joined.stan$MD_EARN_WNE_P10,

                          pm_alpha = 0,
                          ps_alpha = 10000 * scale_param,
                          pm_SAT_ALL = 50,
                          ps_SAT_ALL = 500 * scale_param,
```

```
                            pm_MD_FAMINC = 0,
                            ps_MD_FAMINC = 100 * scale_param,
                            pm_COSTT4_A = 0,
                            ps_COSTT4_A = 500 * scale_param,
                            pm_POVERTY_RATE = 0,
                            ps_POVERTY_RATE = 2500 * scale_param,
                            pm_URBAN = 0,
                            ps_URBAN = 2500,
                            pm_PRIVATE = 0,
                            ps_PRIVATE = 2500 * scale_param,
                            pm_sigma = 10000 * scale_param,
                            ps_sigma = 10000 * scale_param

                            )


pooled.fit <- pooled.model$sample(data = pooled.model.data, seed = 1234, refresh = 1e3)

pooled.fit$summary()
```

Hierarchical model sensitivity analysis

Summary of model fit for main parameters with wide priors:

```
hierarchical.model <- cmdstan_model(stan_file = "./Stan/hierarchical.stan")

scale_param <- 3

hierarchical.model.data <- list(N = nrow(data.joined.stan),
                                K = length(unique(data.joined.stan$REGION)),
                                x = data.joined.stan$REGION,

                                SAT_ALL = data.joined.stan$SAT_ALL,
                                MD_FAMINIC = data.joined.stan$MD_FAMINC,
                                COSTT4_A = data.joined.stan$COSTT4_A,
                                POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                                URBAN = data.joined.stan$URBAN,
                                PRIVATE = data.joined.stan$PRIVATE,
                                y = data.joined.stan$MD_EARN_WNE_P10,

                                pm_alpha = 0,
                                ps_alpha = 10000 * scale_param,
                                pm_s_alpha = 100,
                                ps_s_alpha = 100 * scale_param,

                                pm_SAT_ALL = 43,
                                ps_SAT_ALL = 500 * scale_param,
                                pm_s_SAT_ALL = 100,
                                ps_s_SAT_ALL = 100 * scale_param,

                                pm_MD_FAMINC = 0,
                                ps_MD_FAMINC = 100 * scale_param,
                                pm_s_MD_FAMINC = 10,
                                ps_s_MD_FAMINC = 100 * scale_param,
```

```
                                pm_COSTT4_A = 0,
                                ps_COSTT4_A = 500 * scale_param,
                                pm_s_COSTT4_A = 50,
                                ps_s_COSTT4_A = 500 * scale_param,

                                pm_POVERTY_RATE = 0,
                                ps_POVERTY_RATE = 2500 * scale_param,
                                pm_s_POVERTY_RATE = 500,
                                ps_s_POVERTY_RATE = 500 * scale_param,

                                pm_URBAN = 0,
                                ps_URBAN = 2500 * scale_param,
                                pm_s_URBAN = 500,
                                ps_s_URBAN = 500 * scale_param,

                                pm_PRIVATE = 0,
                                ps_PRIVATE = 2500 * scale_param,
                                pm_s_PRIVATE = 500,
                                ps_s_PRIVATE = 500 * scale_param,

                                pm_sigma = 10000,
                                ps_sigma = 10000 * scale_param,
                                pm_s_sigma = 500,
                                ps_s_sigma = 500 * scale_param)

hierarchical.fit <- hierarchical.model$sample(data = hierarchical.model.data,
                                              seed = 1234, refresh = 1e3)

hierarchical.fit$summary()
```

Summary of model fit for main parameters with narrow priors:

```
hierarchical.model <- cmdstan_model(stan_file = "./Stan/hierarchical.stan")

scale_param <- 0.5

hierarchical.model.data <- list(N = nrow(data.joined.stan),
                                K = length(unique(data.joined.stan$REGION)),
                                x = data.joined.stan$REGION,

                                SAT_ALL = data.joined.stan$SAT_ALL,
                                MD_FAMINIC = data.joined.stan$MD_FAMINC,
                                COSTT4_A = data.joined.stan$COSTT4_A,
                                POVERTY_RATE = data.joined.stan$POVERTY_RATE,
                                URBAN = data.joined.stan$URBAN,
                                PRIVATE = data.joined.stan$PRIVATE,
                                y = data.joined.stan$MD_EARN_WNE_P10,

                                pm_alpha = 0,
                                ps_alpha = 10000 * scale_param,
                                pm_s_alpha = 100,
                                ps_s_alpha = 100 * scale_param,

                                pm_SAT_ALL = 43,
```

```
                                ps_SAT_ALL = 500 * scale_param,
                                pm_s_SAT_ALL = 100,
                                ps_s_SAT_ALL = 100 * scale_param,

                                pm_MD_FAMINC = 0,
                                ps_MD_FAMINC = 100 * scale_param,
                                pm_s_MD_FAMINC = 10,
                                ps_s_MD_FAMINC = 100 * scale_param,

                                pm_COSTT4_A = 0,
                                ps_COSTT4_A = 500 * scale_param,
                                pm_s_COSTT4_A = 50,
                                ps_s_COSTT4_A = 500 * scale_param,

                                pm_POVERTY_RATE = 0,
                                ps_POVERTY_RATE = 2500 * scale_param,
                                pm_s_POVERTY_RATE = 500,
                                ps_s_POVERTY_RATE = 500 * scale_param,

                                pm_URBAN = 0,
                                ps_URBAN = 2500 * scale_param,
                                pm_s_URBAN = 500,
                                ps_s_URBAN = 500 * scale_param,

                                pm_PRIVATE = 0,
                                ps_PRIVATE = 2500 * scale_param,
                                pm_s_PRIVATE = 500,
                                ps_s_PRIVATE = 500 * scale_param,

                                pm_sigma = 10000,
                                ps_sigma = 10000 * scale_param,
                                pm_s_sigma = 500,
                                ps_s_sigma = 500 * scale_param)

hierarchical.fit <- hierarchical.model$sample(data = hierarchical.model.data,
                                              seed = 1234, refresh = 1e3)

hierarchical.fit$summary()
```