Statistical Calibration of Energy-Engineering End-Use Electricity Consumption Estimates: A Bayesian Multilevel Model Approach

Hiroaki Minato
U. S. Energy Information Administration
1000 Independence Ave., SW, Washington, DC 20585

Abstract
How do American homes consume energy? We tackle the question statistically with survey data and engineering estimates. Energy choices and home environments are measured for a national sample of homes by the Residential Energy Consumption Survey (RECS) conducted by the U. S. Energy Information Administration. From the survey response variables on building characteristics as well as energy end choices and uses, engineering models are formulated to estimate end-use energy consumptions. The sample homes' total energy consumptions are also captured in RECS. A statistical objective is to develop a robust probabilistic model to partition the total consumption into end-use consumptions, even when the engineering models may not be very accurate. In this paper, we try a Bayesian multilevel modeling approach for the partitioning, taking electricity as an example. By partially pooling sample homes by geography and end-use combination and by accounting for respondent selection through the weights, a multilevel model of the annual electricity consumption is built from engineering-based electricity end-use estimates. Our uncertainties about model parameters are incorporated as priors.

Key Words: Bayesian multilevel models; engineering-based end-use energy consumption models; partial pooling; Residential Energy Consumption Survey (RECS); Stan

1. Introduction

Energy is literally a fuel in our daily home living. You wake up in the morning and the first thing you do may be to turn on the light. And, going to the kitchen, you might start cooking eggs while toasting bagels and brewing coffee. Your heater or air conditioning may already be providing you with a comfortable dining room temperature. At night, you might prepare your dinner from refrigerated vegetables or frozen meats, while you are doing your laundry. And, you might end your day by watching your favorite TV show or DVD. The question of how American homes consume energy is motivated by a bigger questions of how energy may be enabling people to pursue their activities or conditions of choice in their private homes.

There are several energy sources available for American homes, for example, electricity, natural gas, liquid propane gas, fuel oil, and kerosene. We can use each energy source for different purposes, for example, heating by natural gas, air conditioning by electricity, refrigeration by electricity, and water heating by natural gas. Electricity has the most possible end uses due to its plug-in accessibility, and we focus on it in this paper.

How can we know the electricity consumption amount for each end use in a given home for a given time period? There exist sub-metering technologies that attempt to measure them almost in the continuoustime scale; however, their reliability and adoptability are still under research. In the mean times, we approach the question through energy-engineering end-use modeling that utilizes survey measurements of housing and household characteristics, energy sources, and energy equipment and usage as well as weather data. The survey measurements come from the Residential Energy Consumption Survey (RECS) conducted periodically by the U. S. Energy Information Administration (EIA). The RECS data also contain total annual electricity consumption amount and expenditure, so-called billing data, for each respondent home in the survey sample, which are collected from the home's energy supplier company. In this study, we use the most recent RECS data, i.e., the 2009 RECS data (U. S. Energy Information Administration, 2016). The 2009 weather data are acquired from the National Oceanic and Atmospheric Administration (NOAA) (National Oceanic and Atmospheric Administration, 2016). Our substantive goal is to provide national and some sub-national estimates of electricity consumption by end use, specifically space heating, air conditioning (A/C), water heating, refrigerator, and others, for the calendar year of 2009.

Here, we start with, so we call, energy-engineering electricity end-use estimates and calibrate them against the total annual electricity consumption amount. There are many ways to construct a calibration model with the simplest being a uniform calibration model, i.e., applying one multiplicative factor to all end-use estimates so that they sum to the total annual electricity consumption amount for a given home. We try a Bayesian multilevel modeling approach in differentially calibrating end-use estimates, followed by the uniform calibration.

In the next section, we describe the survey and weather data that are used by energy engineers to estimate the annual electricity consumption for each end use. In Section 3, we briefly explain the energy-engineering annual electricity end-use models and their estimates. In Section 4, we specify a Bayesian multilevel model that calibrates the energy-engineering annual electricity end-use estimates against the total annual electricity consumption values. And, in the final section, we provide our Bayesian inference of annual electricity consumption by end use at the national and Census Region level, along with the currently-published statistical-model-based end-use estimates.

2. Data

Our data come from two sources. One is the periodic national complex-design survey of housing and household characteristics, energy sources, and energy equipment and usage, which is conducted by the Office of Energy Consumption and Efficiency Statistics (OECES) at EIA. It is called the Residential Energy Consumption Survey (RECS), and we use its most recent 2009 data of the sample size 12,083 (U. S. Energy Information Administration, 2016). For the survey respondents' homes, energy consumption data, namely electricity and natural gas consumption and expenditure data, are collected from the corresponding energy supplier companies in the form of energy bills. After the data are edited and processed, any missing values are imputed. We take this cleaned and completed survey data as they are *before* the disclosure control. (The disclosure-controlled 2009 RECS public micro data are available at EIA's Web site (U. S. Energy Information Administration, 2016).)

The other source of data is the Quality Controlled Local Climatological Data (QCLCD) available at the National Centers for Environmental Information of the National Oceanic and Atmospheric Administration (National Oceanic and Atmospheric Administration, 2016). We utilize their weather stations' hourly temperature data to estimate daily average temperatures for heating degree days and cooling degree days, which are summed over the calendar year of 2009, for each survey respondent's home. The weather stations and the respondents' homes are geocoded so that they can be matched in terms of geographical closeness.

3. Energy-Engineering End-Use Models and Estimates

In 2014, OECES conducted a year-long research project to improve its residential and commercial-building energy end-use modeling and estimation methods (U. S. Energy Information Administration, 2015). One of the main tasks was to introduce energy-engineering end-use models to the residential energy end-use modeling, similarly to the commercial-building counterpart. Professional energy engineers from Leidos-Navigant developed those models specifically for: lighting, **space heating**, **A/C**, **water heating**, major appliances (**refrigerators**, freezers, stoves, ovens, and "stovens" (stove-oven components), dishwashers, clothes washers, and clothes dryers), kitchen appliances (microwaves, coffee makers, and toasters), miscellaneous electric loads (dehumidifiers, humidifiers, ceiling fans, computers and monitors, printer, fax, and copier, home network equipment, televisions, set top boxes, DVD's, VCR's, and combo DVD/VCR units, video game consoles, and rechargeable tools and electronics), and others (pool and spa heaters, well pumps, and automobile block/battery heaters). Electricity can be an energy source for any one of these end uses; in fact, all of the 2009 RECS sample cases responded, saying they use electricity for at least one of these end uses. In this paper, we focus on electricity and its four end uses—space heating, A/C, water heating, and refrigerator (bold-faced in the above)—and all the other end uses as one end-use group. (The 2009 RECS used the same five end-use groups for its data publication (U. S. Energy Information Administration, 2016).)

The electricity cooling model, for example, is specified (with some of its details omitted here) by:

Annual Cooling Fuel = Annual Total Cooling Load / Fractional Efficiency of Electric Cooling Equipment

> *where*
>
> Annual Total Cooling Load = Annual Building Cooling Load Conduction Loss + Annual Outside Infiltration Cooling Load
>
> > *where*
> >
> > Annual Building Cooling Load Conduction Loss = 24 Hours/Day of Conduction Load * Cooling Degree Days * Cooling Degree Days Multiplier based on Survey Responses * Weighted Average Building Heat Gain
> >
> > > *where*
> > >
> > > ...
> >
> > *and*
> >
> > Annual Outside Infiltration Cooling Load = Annual Infiltration Sensible Cooling Load + Annual Infiltration Latent Cooling Load
> >
> > > *where*
> > >
> > > ... (such as dew point temperature, elevation and standard pressure, and outside air humidity from ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) Handbooks)
>
> *and*
>
> Fractional Efficiency of Electric Cooling Equipment = …
>
> > *where*
> >
> > ... (such as seasonal adjustments, ENERGY STAR adjustments, and Department of Energy's Fan Efficiency Rating)

Most of the energy-engineering end-use models are not optimal in the sense that they are formulated only with existing 2009 RECS survey measures. Some measurements simply cannot be collected even in an in-person survey like RECS. Also, the weather data for the immediate surrounding of a given home are approximate, as the weather measurements are made only at its closest weather station. Further, we are interested in the annual consumption, but seasonal, monthly, daily, or hourly variations are not always considered in the end-use models.

Thus, each energy-engineering end-use model can provide only an estimate or approximation of annual electricity consumption by that end-use for a given home. Those estimates have unknown error levels, but the levels of uncertainty may vary from one end use to the other and from one group of homes to the other.

Further, as our final substantive goal is to produce the national or sub-national estimates of end-use consumptions, we require some application of survey weights. However, we believe the survey weights can also have some errors due to frame and sample problems such as frame's under- or over-coverage and unit non-response, although we try hard to adjust our weights for those problems. We account for these errors by calibrating the *weighted* energy-engineering annual end-use estimates to the *weighted* reported annual overall consumption values. That is, we integrate survey weighting errors, energy-engineering end-use modeling and estimation errors, and possible data errors in the reported annual total electricity consumption under one framework so that we can handle the propagation of these errors in the final national or sub-national estimates of annual electricity consumptions by end use.

4. Bayesian Multilevel Calibration Models

For each 2009 RECS sample home, we have the annual total or overall electricity consumption value collected as part of the 2009 RECS. The energy-engineering end-use estimates can be calibrated against these annual total electricity consumption values.

As explained in the previous section, we work with the weighted quantities and model them directly. However, it may be worth mentioning that the survey weights could be used as a covariate in the calibration model along with any sample design variables. One way to think about survey weights is that each of the unique weight values (within any stratum) identifies a group of sample units that represent a group of similar or exchangeable units in the population. For example, if two sample units have the unique weight value of 100, they together represent 200 (100 times two) units in the population or stratum, which are assumed to provide more or less the "same" survey responses. The sameness does not mean that they all would give identical responses in every survey question item; rather, they are exchangeable in the sense that any two of the 200 units would represent the population in hypothetical repeated sampling.

There are many calibration methods. The simplest calibration method is a uniform calibration method, applying one multiplicative factor to all end-use estimates so that they sum to the total annual electricity consumption value for a given home. We first apply a Bayesian multilevel modeling approach in differentially calibrating end-use estimates by exploiting some multilevel structures in the rather sparse data. The results are only then uniformly calibrated to the available (reported or imputed) annual total electricity consumption values.

From here on, we omit the qualifier "weighted" unless necessary, because it does not affect our modeling approach. Let y_i = the reported annual total electricity consumption, $x1_i$ = the engineering-model-estimated space heating consumption, $x2_i$ = the engineering-model-estimated A/C consumption, $x3_i$ = the engineering-model-estimated water heating consumption, $x4_i$ = the engineering-model-estimated refrigerator consumption, and $x5_i$ = the engineering-model-estimated other consumption for home i (where i = 1, …, 12,083). Ideally, the sum of the annual energy-engineering electricity end-use estimates equals the reported annual total electricity consumption for each home:

$$y_i = x1_i + x2_i + x3_i + x4_i + x5_i \text{ for all } i.$$

However, our data require a better representation, and we consider the following data model with the varying differential multiplicative factors $\beta 1$, $\beta 2$, $\beta 3$, $\beta 4$, and $\beta 5$ for $x1$, $x2$, $x3$, $x4$, and $x5$, respectively, and the varying multiplicative Census Region adjustment factor α:

$$y_i \sim \text{normal}\left(\alpha_{r[i]}(\beta 1_{\mathrm{e}} x1_i + \beta 2_{\mathrm{e}} x2_i + \beta 3_{\mathrm{e}} x3_i + \beta 4_{\mathrm{e}} x4_i + \beta 5_{\mathrm{e}} x5_i), \sigma_{(r,\mathrm{e})[i]}\right),$$

where the second parameter in the normal distribution notation is the standard deviation, i indexes the sample homes as before, $r[i]$ indicates the Census Region that home i belongs to, e is one of the actual values that classify the sample homes by end-use combination (to be described later), and $(r, \mathrm{e})[i]$ identifies a group of homes that belong to the Census Region r and the end-use combination e. The Census Regions are Northeast Census Region (r = 1), Midwest Census Region (r = 2), South Census Region (r = 3), and West Census Region (r = 4). And, there are 14 electricity end-use consumption combinations in the sample data as below, where 1 means existent of the end use and 0 means non-existent.

Table 1: Electricity End-Use Consumption Combinations

End-Use Combination (e)	*x1 (Space Heating)*	*x2 (A/C)*	*x3 (Water Heating)*	*x4 (Refrigerator)*	*x5 (Others)*	*Sample Size*
1	0	0	0	0	1	7
2	0	0	1	0	1	2
3	1	0	0	0	1	2
4	0	1	1	0	1	1
5	0	1	0	0	1	1
6	1	1	1	0	1	6
7	0	0	0	1	1	1,056
8	1	0	0	1	1	437
9	1	1	0	1	1	1,983
10	0	0	1	1	1	244
11	0	1	0	1	1	3,784
12	1	0	1	1	1	395
13	1	1	1	1	1	3,331
14	0	1	1	1	1	834
All						12,083

So, e in the above data model is actually 13, that is, e = 13. This end-use combination was chosen to illustrate our model, because it is the combination that contains all five end uses. And, the i, in fact, indexes only those sample homes that said in the survey they use all five end uses with electricity. Another example: for the sample homes that said they use only *x5* end use with electricity (e = 1), we specify the following data model:

$$y_i \sim \text{normal}\left(\alpha_{r[i]} \beta 5_1 x5_i, \sigma_{(r,1)[i]}\right).$$

In short, we are pooling our independently-distributed data by the Census Region variable and the end-use combination variable. By these partial pooling decisions, we are assuming that homes in the same pooling group have an identical distribution of overall electricity consumption and allocate it to particular end uses in the same proportions. In fact, homes in the same Census Region have a similar external environment, climate, or culture (beyond what engineering-based end-use models might distinguish and account for), while homes in the same end-use combination have a common profile of fuel availability and equipment choice and usage (beyond what engineering-based end-use models can model individual home end uses—for example, the A/C electricity end-use model doesn't distinguish households with respect to whether they also use electricity for space heating or not).

Note that some end-use combinations, specifically e = 1, …, 6, have very small numbers of homes in the sample. They are rare because they responded to the survey by stating no electricity usage for $x4$ (refrigerator) in 2009. Also note that 31 ($= 2^5 - 1$) end-use consumption combinations are actually possible, excluding the all-zero end-use consumption combination, but that we happen to observe only 14 of them in our sample data. This is another indication of sparsity or holes of our data along this dimension.

The normal distributions are approximate models of our data or approximate likelihoods of our parameters. It is firstly because y_i are all positive but the support of the distribution is the real line. Some normalization or transformations of data, particularly log transformations of $x1_i$, $x2_i$, $x3_i$, $x4_i$, and $x5_i$ as well as y_i, were considered, but we decided to keep them in the original unit of kWh for a substantively easier description and interpretation of the model. The positivity approximation is not so unreasonable, however, as the distributions are pushed to positive values by the data and also by the parameters, which are discussed later. The normality of the distribution is also approximate, which is admittedly for theoretical and computational convenience, but it is not necessarily a bad default form for our data.

Next, taking a Bayesian approach, we model our uncertainties about the data model parameters. Our choices of priors are all informative and are based on normal distributions. However, since all our parameters are multiplicative factors for positive data values, we restrict the parameter space to be non-negative by folding all the normal distributions. For the Census Region parameters, we specify:

$$\alpha_r \sim \begin{cases} \text{folded normal}(1,\ 0.5), \text{ if r = Northeast or Midwest} \\ \text{folded normal}(1,\ 1), \text{ if r = South or West} \end{cases}.$$

The mean of the distribution is set to 1 for each Census Region, as we regularize these parameters toward the ideal model parameter value of 1. The standard deviation of the distribution distinguishes South and West from Northeast and Midwest. Electricity is used more and for more end uses in South and West, compared to Northeast and Midwest (e.g., with respect to space heating, water heating, and cooking), and this more popular and diverse use of electricity in the former Census Regions brings more uncertainty in their end-use estimates and the overall calibration of the aggregated end-use estimate to the reported total consumption. Specifically, the standard deviation for Northeast and Midwest is a half of the mean, while that for South and West is twice as large and is equal to the mean.

Similarly, for the end-use multipliers $\beta1_e$, $\beta2_e$, $\beta3_e$, $\beta4_e$, and $\beta5_e$, we specify the folded normal distributions with the means of one's as regularizers. Recall the existence of any of these end-use multiplicative factors in the model depends on which end-use combination the sample home belongs to and also that the end-use multipliers are unique to the end-use combination. Using the end-use combination 13 as an example again, we assume the following priors:

$$\beta1_{13} \sim \text{folded normal}(1,\ 0.5) \text{ for } x1 \text{ (space heating);}$$
$$\beta2_{13} \sim \text{folded normal}(1,\ 0.5) \text{ for } x2 \text{ (A/C);}$$
$$\beta3_{13} \sim \text{folded normal}(1,\ 0.5) \text{ for } x3 \text{ (water heating);}$$
$$\beta4_{13} \sim \text{folded normal}(1,\ 0.5) \text{ for } x4 \text{ (refrigerator); and}$$
$$\beta5_{13} \sim \text{folded normal}(1,\ 1) \text{ for } x5 \text{ (others).}$$

$x5$ (others) could contain electricity consumption of numerous other end uses, which can vary from one home to another, and we place more uncertainty in the engineering-model estimate of the other end uses than in the engineering-model estimates of space heating, A/C, water heating, or refrigerators. The standard deviation for $\beta 5_e$ is 100% of the mean for each *e*, while the standard deviations for $\beta 1_e$, $\beta 2_e$, $\beta 3_e$, and $\beta 4_e$ are just 50% of the means for each *e*. In short, these priors relatively strongly push $\beta 1_e$, $\beta 2_e$, $\beta 3_e$, and $\beta 4_e$ to 1's while keeping $\beta 5_e$ weak and forgiving of the data.

The uncertainty levels of the four engineering-based end-use estimates could actually vary. In fact, spacing heating and A/C have much more elaborate engineering models than water heating or refrigerator. However, it is not obvious how to differentiate our uncertainty level among them at this moment, and they are treated in the same way by identical prior specifications.

Finally, the standard deviation of y_i, $\sigma_{(r,\mathrm{e})}$, also gets the folded normal distribution but with the mean of zero so that the prior is peaked at zero and with some empirical standard deviation:

$$\sigma_{r,e} \sim \text{folded normal}(0, |\overline{w\delta_{r,e}}|/6) \text{ for } r \text{ and } e \text{ pairs,}$$

where $|\overline{w\delta_{r,e}}|$ is the absolute value of the observed average of $w_i y_i - w_i x1_i - w_i x2_i - w_i x3_i - w_i x4_i - w_i x5_i$ when the home *i* belongs to the end-use combination *e* and the Census Region *r*. The standard deviation of $\sigma_{r,e}$, $|\overline{w\delta_{r,e}}|/6$, makes the support of the non-folded normal distribution approximately $|\overline{w\delta_{r,e}}|$, because the interval from 3 times the minus of the standard deviation to 3 times the standard deviation contains almost 100% of the probability mass. This is an attempt to regularize $\sigma_{r,e}$ to be near 0, though it could still be as big as $|\overline{w\delta_{r,e}}|$ or even bigger. On one hand, E($\sigma_{r,e}$) = 0 is intended to push $\beta 1_e$, $\beta 2_e$, $\beta 3_e$, $\beta 4_e$, and $\beta 5_e$ to be more varying away from 1 because they would have to fill the large gap between $w_i y_i$ and $w_i x1_i + w_i x2_i + w_i x3_i + w_i x4_i + w_i x5_i$. On the other hand, E($\sigma_{r,e}$) = $|\overline{w\delta_{r,e}}|$ allows $\beta 1_e$, $\beta 2_e$, $\beta 3_e$, $\beta 4_e$, and $\beta 5_e$ to approach 1 if the data so dictate.

5. Bayesian Population Inference

In acquiring or estimating the posterior distributions of the data model parameters, we use Stan, "a free and open-source probabilistic programming language and Bayesian inference engine," through its R interface, RStan (Stan Development Team, 2016a and 2016b). Stan allows "full Bayesian statistical inference with MCMC sampling (NUTS, HMC)", "approximate Bayesian inference with variational inference (ADVI)", and "penalized maximum likelihood estimation with optimization (L-BFGS)" (Stan Development Team, 2016a). We have used NUTS (No-U-Turn Sampler), which optimizes HMC (Hamilton Monte Carlo) adaptively, to generate a single chain of 3,000 thinned-by-one cases after 34,000 warm-up cases. That is, we have taken every other case after the 34,000-th warm-up case until we accumulate 3,000 sample cases.

We have also tried multiple chains with shorter sequences, but our particular problem seems to require a long sequence (of 40,000 or more cases) to reach "reasonable" posterior distributions. Further investigations are still in need, and our posterior distributions are tentative, although the posterior means are close to the posterior medians. Our Stan codes are given in Appendix A, and some of our RStan codes in Appendix B.

For each sample home, we have generated 3,000 consumption values for each existing end use from the 3,000 NUTS sample and have computed their mean as the posterior mean of the given end-use consumption. For example, for the sample home *i* in the Census Region r with the end-use combination e, the posterior mean of the calibrated space heating consumption values is:

$$\frac{\sum_{j=1}^{3000} \alpha_{\mathrm{r},j} \beta 1_{\mathrm{e},j} x1_i}{3000},$$

where $\alpha_{\mathrm{r},j}$ is the j-th NUTS posterior sample value of the parameter α_{r} and $\beta 1_{\mathrm{e},j}$ is the j-th NUTS posterior sample value of the parameter $\beta 1_{\mathrm{e}}$.

For each sample home, after this Bayesian calibration of engineering-based end-use estimates, we apply the uniform calibration to make the sum of end-use estimates equal to the overall consumption value. We call this final end-use estimates. They are aggregated to the national and Census Region levels in Table 2. For comparisons, not for validations, the existing published estimates are also shown in the table.

Table 2: Final Electricity End-Use Estimates

Bayesian-calibrated engineering-based estimates (billion kWh)

Census Region	*Space heating*	*A/C*	*Water heating*	*Refrigerator*	*Others*	*Overall*
Northeast	8	14	18	16	112	168
Midwest	20	26	24	24	181	274
South	23	73	50	35	431	613
West	9	17	34	27	145	231
Nation	60	129	126	102	868	1,286

Currently-published statistical-model-based estimates (billion kWh)

Census Region	*Space heating*	*A/C*	*Water heating*	*Refrigerator*	*Others*	*Overall*
Northeast	11	11	13	23	109	168
Midwest	24	20	22	34	175	274
South	64	129	72	55	293	613
West	20	27	19	29	136	231
Nation	120	186	126	142	713	1,286

The Bayesian-calibrated engineering-based end-use estimates by Census Region are graphed in Figure 1.

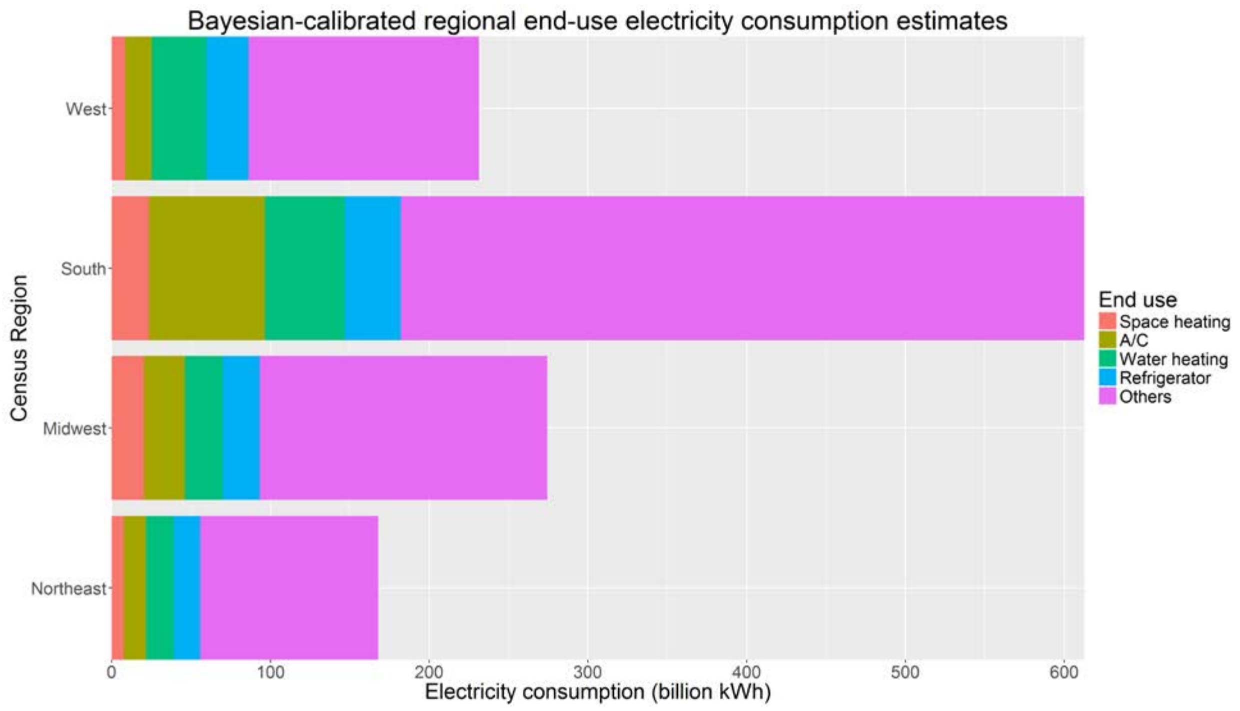

Figure 1: Bayesian-calibrated engineering-based end-use estimates by Census Region

The national-level comparison of the Bayesian-calibrated engineering-based end-use estimates against the currently published statistical-model-based estimates is graphed in Figure 2.

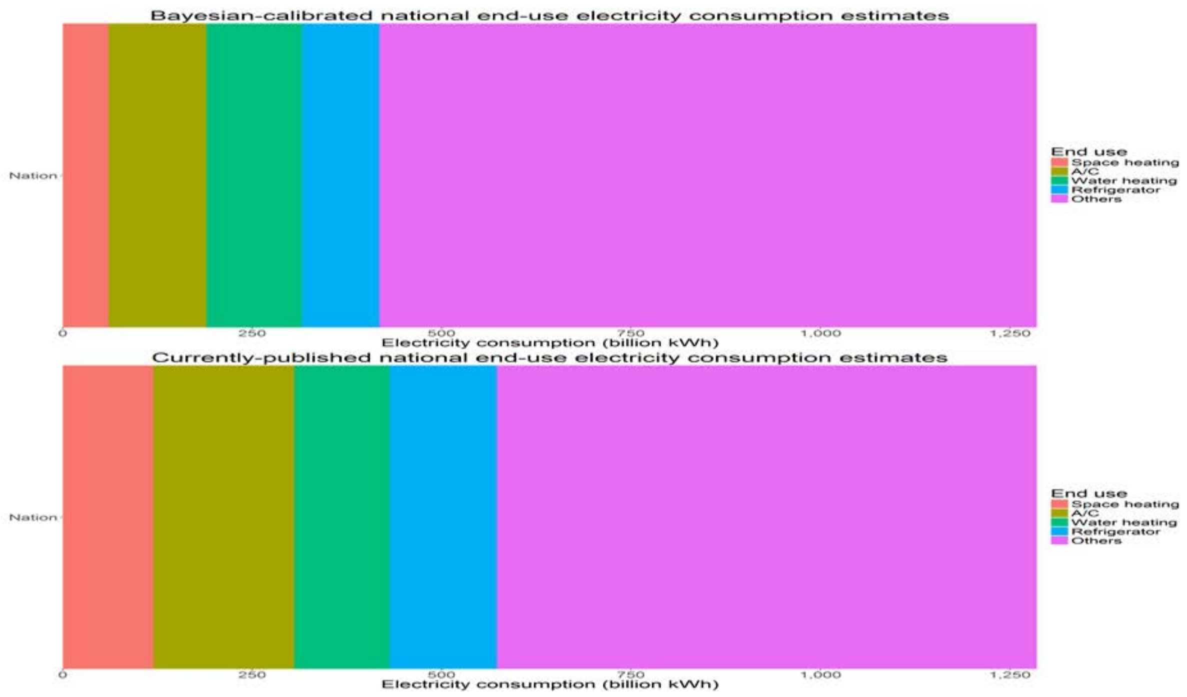

Figure 2: Comparison of Bayesian-calibrated engineering-based end-use estimates and the currently-published statistical-model-based estimates at the national level

Some of our next steps are (1) more rigorous evaluation of the NUTS samples, (2) cross-validation of the models, (3) examination of robustness against bad data, i.e., inaccurate engineering-based end-use estimates, (4) analysis of sensitivity of the posteriors to the priors, and (5) assessment of accuracy in the population inference. In addition, in the Bayesian framework, we will consider modeling the engineering-based end-use estimate errors and the survey weighting errors more directly as measurement errors (Stan Development Team, 2016b). Energy conditions our homes and fuels many of our home equipment and tools so that we can live comfortably, productively, or safely at our homes. More accurate description of the home energy anatomy is about to come.

Appendix A. Stan codes for multilevel nonlinear regression models of energy-engineering end-use electricity consumption estimates

```
data {
  int<lower = 1> N;  // The sample size
  int<lower = 1> n_eug;  // The number of end-use combinations
  int<lower = 1> n_reg;  // The number of Census Regions
  int<lower = 1, upper = n_eug> eug[N];  // The end-use combination variable
  int<lower = 1, upper = n_reg> reg[N];  // The Census Region variable
  int<lower = 1> S;  // The number of standard deviations for the pairs of eug and reg
  vector<lower = 0>[N] su_id;  // The sample case ID
  vector<lower = 0>[N] x1;  // Weighted space heating electricity consumption (kWh)
  vector<lower = 0>[N] x2;  // Weighted A/C electricity consumption (kWh)
  vector<lower = 0>[N] x3;  // Weighted water heating electricity consumption (kWh)
  vector<lower = 0>[N] x4;  // Weighted refrigerator electricity consumption (kWh)
  vector<lower = 0>[N] x5;  // Weighted other electricity consumption (kWh)
  vector<lower = 0>[N] y;   // Weighted overall (reported or imputed) electricity consumption
(kWh)
  vector[S] error_sigma;  // Absolute value of the average of (y - x1 - x2 - x3 - x4 - x5) in
each pair of eug and reg
}


parameters {
  real<lower = 0> e1_3;  // The end-use parameter for x1 in eum = 3
  real<lower = 0> e1_6;
  real<lower = 0> e1_8;
  real<lower = 0> e1_9;
  real<lower = 0> e1_12;
  real<lower = 0> e1_13;
  real<lower = 0> e2_4;  // The end-use parameter for x2 in eum = 4
  real<lower = 0> e2_5;
  real<lower = 0> e2_6;
  real<lower = 0> e2_9;
  real<lower = 0> e2_11;
  real<lower = 0> e2_13;
  real<lower = 0> e2_14;
  real<lower = 0> e3_2;  // The end-use parameter for x3 in eum = 2
  real<lower = 0> e3_4;
  real<lower = 0> e3_6;
  real<lower = 0> e3_10;
```

```
real<lower = 0> e3_12;
real<lower = 0> e3_13;
real<lower = 0> e3_14;
real<lower = 0> e4_7;  // The end-use parameter for x4 in eum = 7
real<lower = 0> e4_8;
real<lower = 0> e4_9;
real<lower = 0> e4_10;
real<lower = 0> e4_11;
real<lower = 0> e4_12;
real<lower = 0> e4_13;
real<lower = 0> e4_14;
real<lower = 0> e5_1;  // The end-use parameter for x5 in eum = 1
real<lower = 0> e5_2;
real<lower = 0> e5_3;
real<lower = 0> e5_4;
real<lower = 0> e5_5;
real<lower = 0> e5_6;
real<lower = 0> e5_7;
real<lower = 0> e5_8;
real<lower = 0> e5_9;
real<lower = 0> e5_10;
real<lower = 0> e5_11;
real<lower = 0> e5_12;
real<lower = 0> e5_13;
real<lower = 0> e5_14;

vector<lower = 0>[n_reg] r;  // A vector of the Census Region parameters

real<lower = 0> sigma_1_1;  // The sigma parameter for reg = 1 and eum = 1
real<lower = 0> sigma_1_2;
real<lower = 0> sigma_1_4;
real<lower = 0> sigma_1_7;
real<lower = 0> sigma_1_8;
real<lower = 0> sigma_1_9;
real<lower = 0> sigma_1_10;
real<lower = 0> sigma_1_11;
real<lower = 0> sigma_1_12;
real<lower = 0> sigma_1_13;
real<lower = 0> sigma_1_14;
real<lower = 0> sigma_2_1;  // The sigma parameter for reg = 2 and eum = 1
real<lower = 0> sigma_2_6;
real<lower = 0> sigma_2_7;
real<lower = 0> sigma_2_8;
real<lower = 0> sigma_2_9;
real<lower = 0> sigma_2_10;
real<lower = 0> sigma_2_11;
real<lower = 0> sigma_2_12;
real<lower = 0> sigma_2_13;
real<lower = 0> sigma_2_14;
```

```
  real<lower = 0> sigma_3_1;  // The sigma parameter for reg = 3 and eum = 1
  real<lower = 0> sigma_3_2;
  real<lower = 0> sigma_3_3;
  real<lower = 0> sigma_3_5;
  real<lower = 0> sigma_3_6;
  real<lower = 0> sigma_3_7;
  real<lower = 0> sigma_3_8;
  real<lower = 0> sigma_3_9;
  real<lower = 0> sigma_3_10;
  real<lower = 0> sigma_3_11;
  real<lower = 0> sigma_3_12;
  real<lower = 0> sigma_3_13;
  real<lower = 0> sigma_3_14;
  real<lower = 0> sigma_4_1;  // The sigma parameter for reg = 4 and eum = 1
  real<lower = 0> sigma_4_3;
  real<lower = 0> sigma_4_6;
  real<lower = 0> sigma_4_7;
  real<lower = 0> sigma_4_8;
  real<lower = 0> sigma_4_9;
  real<lower = 0> sigma_4_10;
  real<lower = 0> sigma_4_11;
  real<lower = 0> sigma_4_12;
  real<lower = 0> sigma_4_13;
  real<lower = 0> sigma_4_14;
}


model {
for (i in 1:N) {
       if (eug[i] == 1)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e5_1 * x5[i]), sigma_1_1);
           else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e5_1 * x5[i]), sigma_2_1);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e5_1 * x5[i]), sigma_3_1);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e5_1 * x5[i]), sigma_4_1);}
  else if (eug[i] == 2)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e3_2 * x3[i] + e5_2 * x5[i]),
sigma_1_2);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e3_2 * x3[i] + e5_2 * x5[i]),
sigma_3_2);}
  else if (eug[i] == 3)
          {     if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_3 * x1[i] + e5_3 * x5[i]),
sigma_3_3);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_3 * x1[i] + e5_3 * x5[i]),
sigma_4_3);}
  else if (eug[i] == 4)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e2_4 * x2[i] + e3_4 * x3[i] + e5_4
* x5[i]), sigma_1_4);}
  else if (eug[i] == 5)
```

```
          {     if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e2_5 * x2[i] + e5_5 * x5[i]),
sigma_3_5);}
  else if (eug[i] == 6)
          {     if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]), sigma_2_6);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]), sigma_3_6);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]), sigma_4_6);}
  else if (eug[i] == 7)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]),
sigma_1_7);
           else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]),
sigma_2_7);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]),
sigma_3_7);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]),
sigma_4_7);}
  else if (eug[i] == 8)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]), sigma_1_8);
           else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]), sigma_2_8);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]), sigma_3_8);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]), sigma_4_8);}
  else if (eug[i] == 9)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]), sigma_1_9);
           else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]), sigma_2_9);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]), sigma_3_9);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]), sigma_4_9);}
  else if (eug[i] == 10)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]), sigma_1_10);
           else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]), sigma_2_10);
           else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]), sigma_3_10);
           else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]), sigma_4_10);}
  else if (eug[i] == 11)
          {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]), sigma_1_11);
```

```
          else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]), sigma_2_11);
          else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]), sigma_3_11);
          else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]), sigma_4_11);}
  else if (eug[i] == 12)
         {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]), sigma_1_12);
          else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]), sigma_2_12);
          else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]), sigma_3_12);
          else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]), sigma_4_12);}
  else if (eug[i] == 13)
         {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]), sigma_1_13);
          else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]), sigma_2_13);
          else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]), sigma_3_13);
          else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]), sigma_4_13);}
  else if (eug[i] == 14)
         {     if (reg[i] == 1) y[i] ~ normal(r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]), sigma_1_14);
          else if (reg[i] == 2) y[i] ~ normal(r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]), sigma_2_14);
          else if (reg[i] == 3) y[i] ~ normal(r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]), sigma_3_14);
          else if (reg[i] == 4) y[i] ~ normal(r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]), sigma_4_14);}
  }

  e1_3  ~ normal(1, 0.5);
  e1_6  ~ normal(1, 0.5);
  e1_8  ~ normal(1, 0.5);
  e1_9  ~ normal(1, 0.5);
  e1_12 ~ normal(1, 0.5);
  e1_13 ~ normal(1, 0.5);
  e2_4  ~ normal(1, 0.5);
  e2_5  ~ normal(1, 0.5);
  e2_6  ~ normal(1, 0.5);
  e2_9  ~ normal(1, 0.5);
  e2_11 ~ normal(1, 0.5);
  e2_13 ~ normal(1, 0.5);
  e2_13 ~ normal(1, 0.5);
  e3_2  ~ normal(1, 0.5);
  e3_4  ~ normal(1, 0.5);
  e3_6  ~ normal(1, 0.5);
  e3_10 ~ normal(1, 0.5);
  e3_12 ~ normal(1, 0.5);
  e3_13 ~ normal(1, 0.5);
  e3_14 ~ normal(1, 0.5);
  e4_7  ~ normal(1, 0.5);
  e4_8  ~ normal(1, 0.5);
  e4_9  ~ normal(1, 0.5);
  e4_10 ~ normal(1, 0.5);
  e4_11 ~ normal(1, 0.5);
  e4_12 ~ normal(1, 0.5);
  e4_13 ~ normal(1, 0.5);
  e4_14 ~ normal(1, 0.5);
  e5_1  ~ normal(1, 1);
  e5_2  ~ normal(1, 1);
  e5_3  ~ normal(1, 1);
  e5_4  ~ normal(1, 1);
  e5_5  ~ normal(1, 1);
  e5_6  ~ normal(1, 1);
  e5_7  ~ normal(1, 1);
  e5_8  ~ normal(1, 1);
  e5_9  ~ normal(1, 1);
  e5_10 ~ normal(1, 1);
  e5_11 ~ normal(1, 1);
  e5_12 ~ normal(1, 1);
  e5_13 ~ normal(1, 1);
  e5_14 ~ normal(1, 1);

  r[1]  ~ normal(1, 0.5);
  r[2]  ~ normal(1, 0.5);
  r[3]  ~ normal(1, 1);
  r[4]  ~ normal(1, 1);

# E(sigma) = 0, which forces beta's to be more elastic or away from 1
# and the SD(sigma) is set to = |error_sigma| / 6, which makes the support to be about
|error_sigma|.
# That is, these priors center the sigmas at 0 but allow them to vary in the [0,
|error_sigma|] range.
  sigma_1_1  ~ normal(0, fabs(error_sigma[1]) / 6);
  sigma_1_2  ~ normal(0, fabs(error_sigma[2]) / 6);
  sigma_1_4  ~ normal(0, fabs(error_sigma[3]) / 6);
  sigma_1_7  ~ normal(0, fabs(error_sigma[4]) / 6);
  sigma_1_8  ~ normal(0, fabs(error_sigma[5]) / 6);
  sigma_1_9  ~ normal(0, fabs(error_sigma[6]) / 6);
  sigma_1_10 ~ normal(0, fabs(error_sigma[7]) / 6);
  sigma_1_11 ~ normal(0, fabs(error_sigma[8]) / 6);
  sigma_1_12 ~ normal(0, fabs(error_sigma[9]) / 6);
  sigma_1_13 ~ normal(0, fabs(error_sigma[10]) / 6);
  sigma_1_14 ~ normal(0, fabs(error_sigma[11]) / 6);
  sigma_2_1  ~ normal(0, fabs(error_sigma[12]) / 6);
  sigma_2_6  ~ normal(0, fabs(error_sigma[13]) / 6);
  sigma_2_7  ~ normal(0, fabs(error_sigma[14]) / 6);
  sigma_2_8  ~ normal(0, fabs(error_sigma[15]) / 6);
  sigma_2_9  ~ normal(0, fabs(error_sigma[16]) / 6);
  sigma_2_10 ~ normal(0, fabs(error_sigma[17]) / 6);
  sigma_2_11 ~ normal(0, fabs(error_sigma[18]) / 6);
  sigma_2_12 ~ normal(0, fabs(error_sigma[19]) / 6);
  sigma_2_13 ~ normal(0, fabs(error_sigma[20]) / 6);
  sigma_2_14 ~ normal(0, fabs(error_sigma[21]) / 6);
  sigma_3_1  ~ normal(0, fabs(error_sigma[22]) / 6);
  sigma_3_2  ~ normal(0, fabs(error_sigma[23]) / 6);
  sigma_3_3  ~ normal(0, fabs(error_sigma[24]) / 6);
  sigma_3_5  ~ normal(0, fabs(error_sigma[25]) / 6);
  sigma_3_6  ~ normal(0, fabs(error_sigma[26]) / 6);
  sigma_3_7  ~ normal(0, fabs(error_sigma[27]) / 6);
  sigma_3_8  ~ normal(0, fabs(error_sigma[28]) / 6);
  sigma_3_9  ~ normal(0, fabs(error_sigma[29]) / 6);
  sigma_3_10 ~ normal(0, fabs(error_sigma[30]) / 6);
  sigma_3_11 ~ normal(0, fabs(error_sigma[31]) / 6);
  sigma_3_12 ~ normal(0, fabs(error_sigma[32]) / 6);
  sigma_3_13 ~ normal(0, fabs(error_sigma[33]) / 6);
  sigma_3_14 ~ normal(0, fabs(error_sigma[34]) / 6);
  sigma_4_1  ~ normal(0, fabs(error_sigma[35]) / 6);
  sigma_4_3  ~ normal(0, fabs(error_sigma[36]) / 6);
  sigma_4_6  ~ normal(0, fabs(error_sigma[37]) / 6);
  sigma_4_7  ~ normal(0, fabs(error_sigma[38]) / 6);
  sigma_4_8  ~ normal(0, fabs(error_sigma[39]) / 6);
  sigma_4_9  ~ normal(0, fabs(error_sigma[40]) / 6);
  sigma_4_10 ~ normal(0, fabs(error_sigma[41]) / 6);
  sigma_4_11 ~ normal(0, fabs(error_sigma[42]) / 6);
  sigma_4_12 ~ normal(0, fabs(error_sigma[43]) / 6);
  sigma_4_13 ~ normal(0, fabs(error_sigma[44]) / 6);
  sigma_4_14 ~ normal(0, fabs(error_sigma[45]) / 6);
}


generated quantities {
  vector[N] x1_est;
  vector[N] x2_est;
  vector[N] x3_est;
  vector[N] x4_est;
  vector[N] x5_est;
  vector[N] y_est;
  for (i in 1:N) {
       if (eug[i] == 1)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e5_1 * x5[i]));
                                x1_est[i] = 0;
```

```
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_1 * x5[i]));}
          else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e5_1 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_1 * x5[i]));}
          else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e5_1 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_1 * x5[i]));}
          else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e5_1 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_1 * x5[i]));}}
else if (eug[i] == 2)
         {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e3_2 * x3[i] + e5_2 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_2 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_2 * x5[i]));}
          else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e3_2 * x3[i] + e5_2 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_2 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_2 * x5[i]));}}
else if (eug[i] == 3)
         {     if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_3 * x1[i] + e5_3 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_3 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_3 * x5[i]));}
          else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_3 * x1[i] + e5_3 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_3 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_3 * x5[i]));}}
else if (eug[i] == 4)
```

```
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e2_4 * x2[i] + e3_4 * x3[i] + e5_4
* x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_4 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_4 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_4 * x5[i]));}}
  else if (eug[i] == 5)
          {     if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e2_5 * x2[i] + e5_5 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_5 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_5 * x5[i]));}}
  else if (eug[i] == 6)
          {     if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_6 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_6 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_6 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_6 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_6 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_6 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_6 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_6 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_6 * x1[i] + e2_6 * x2[i] + e3_6
* x3[i] + e5_6 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_6 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_6 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_6 * x3[i]));
                              x4_est[i] = 0;
                              x5_est[i] = (r[reg[i]]*(e5_6 * x5[i]));}}
  else if (eug[i] == 7)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_7 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_7 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_7 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_7 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_7 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_7 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e4_7 * x4[i] + e5_7 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_7 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_7 * x5[i]));}}
  else if (eug[i] == 8)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_8 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_8 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_8 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_8 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;x4_est[i] = (r[reg[i]]*(e4_7 * x4[i]));
                              x4_est[i] = (r[reg[i]]*(e4_8 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_8 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_8 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_8 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_8 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_8 * x1[i] + e4_8 * x4[i] + e5_8
* x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_8 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_8 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_8 * x5[i]));}}
  else if (eug[i] == 9)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_9 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_9 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_9 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_9 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_9 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_9 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_9 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_9 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_9 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_9 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_9 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_9 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_9 * x1[i] + e2_9 * x2[i] + e4_9
* x4[i] + e5_9 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_9 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_9 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_9 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_9 * x5[i]));}}
  else if (eug[i] == 10)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_10 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_10 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_10 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_10 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_10 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_10 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_10 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_10 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_10 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e3_10 * x3[i] + e4_10 * x4[i] +
e5_10 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_10 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_10 * x4[i]));
```

```
                              x5_est[i] = (r[reg[i]]*(e5_10 * x5[i]));}}
  else if (eug[i] == 11)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_11 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_11 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_11 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_11 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_11 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_11 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_11 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_11 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_11 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e2_11 * x2[i] + e4_11 * x4[i] +
e5_11 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_11 * x2[i]));
                              x3_est[i] = 0;
                              x4_est[i] = (r[reg[i]]*(e4_11 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_11 * x5[i]));}}
  else if (eug[i] == 12)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_12 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_12 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_12 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_12 * x5[i]));}
           else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_12 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_12 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_12 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_12 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_12 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_12 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_12 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_12 * x5[i]));}
          else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_12 * x1[i] + e3_12 * x3[i] +
e4_12 * x4[i] + e5_12 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_12 * x1[i]));
                              x2_est[i] = 0;
                              x3_est[i] = (r[reg[i]]*(e3_12 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_12 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_12 * x5[i]));}}
  else if (eug[i] == 13)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_13 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_13 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_13 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_13 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_13 * x5[i]));}
          else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_13 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_13 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_13 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_13 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_13 * x5[i]));}
          else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_13 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_13 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_13 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_13 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_13 * x5[i]));}
          else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e1_13 * x1[i] + e2_13 * x2[i] +
e3_13 * x3[i] + e4_13 * x4[i] + e5_13 * x5[i]));
                              x1_est[i] = (r[reg[i]]*(e1_13 * x1[i]));
                              x2_est[i] = (r[reg[i]]*(e2_13 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_13 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_13 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_13 * x5[i]));}}
  else if (eug[i] == 14)
          {     if (reg[i] == 1) {y_est[i]  = (r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_14 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_14 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_14 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_14 * x5[i]));}
          else if (reg[i] == 2) {y_est[i]  = (r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]));
```

```
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_14 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_14 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_14 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_14 * x5[i]));}
           else if (reg[i] == 3) {y_est[i]  = (r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_14 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_14 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_14 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_14 * x5[i]));}
           else if (reg[i] == 4) {y_est[i]  = (r[reg[i]]*(e2_14 * x2[i] + e3_14 * x3[i] +
e4_14 * x4[i] + e5_14 * x5[i]));
                              x1_est[i] = 0;
                              x2_est[i] = (r[reg[i]]*(e2_14 * x2[i]));
                              x3_est[i] = (r[reg[i]]*(e3_14 * x3[i]));
                              x4_est[i] = (r[reg[i]]*(e4_14 * x4[i]));
                              x5_est[i] = (r[reg[i]]*(e5_14 * x5[i]));}}
  }
}
```

Appendix B: RStan codes for NUTS sampling of the parameters in the Bayesian multilevel nonlinear regression models of energy-engineering end-use electricity consumption estimate

```
stan_model_recs <- stan("RECS_VALUES_STAN_7.stan",
                        iter   = 40000,
                        warmup = 34000,
                        thin = 2,
                        chains = 1,
                        init = "random",
                        seed = sample.int(.Machine$integer.max, 1),
                        algorithm = c("NUTS", "HMC", "Fixed_param"),
                        sample_file = NULL,
                        diagnostic_file = NULL,
                        save_dso = TRUE,
                        verbose = TRUE,
                        include = TRUE,
                        cores = getOption("mc.cores", 1L),
                        init_r = 2,
                        control = list(adapt_engaged = TRUE,
                                       adapt_gamma = 0.05,
                                       adapt_delta = 0.8,
                                       adapt_kappa = 0.75,
                                       adapt_t0 = 150,
                                       adapt_init_buffer = 75,
                                       adapt_term_buffer = 50,
                                       adapt_window = 25,
                                       max_treedepth = 10),
                        boost_lib = NULL,
                        eigen_lib = NULL
                        )
```

References

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2016), "Stan: A probabilistic Programming Language," *Journal of Statistical Software* (in press).

Chen, Q., Gelman, A., Tracy, M., Norris, F. H., and Galea, S. (2015), "Incorporating the Sampling Design in Weighting Adjustments for Panel Attrition," *Statistics in Medicine*, 34, 3637-47.

Gelman, A. (2006), "Multilevel (Hierarchical) Modeling: What It Can and Cannot Do," *Technometrics*, 48, 432-435.

Gelman, A. (2007), "Struggles with Survey Weighting and Regression Modeling" (with discussion), *Statistical Science*, 22, 153-164.

Gelman, A. (2010), "Bayesian Statistics Then and Now," Discussion of "The Future of Indirect Evidence," by B. Efron, *Statistical Science*, 25,162-165.

Gelman, A. (2011), "The Pervasive Twoishness of Statistics; in Particular, the Sampling Distribution and the Likelihood are Two Different Models, and That is a Good Thing," Statistical Modeling, Causal Inference, and Social Science Blog, 20 June. Available at http://andrewgelman.com/2011/06/20/the sampling di 1/.

Gelman, A. (2014), "How Do We Choose Our Default Methods?," in Past, Present, and Future of Statistical Science, eds. X. Lin, C. Genest, D. L. Banks, G. Molenberghs, D. W. Scott, and J.-L.Wang, Boca Raton, FL: Chapman & Hall/CRC, pp. 291-299.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013), *Bayesian Data Analysis* (3rd ed.), Boca Raton, FL: Chapman & Hall/CRC.

Gelman, A., and Hill, J. (2006), *Data Analysis Using Regression and Multilevel/Hierarchical Models*, New York, NY: Cambridge University Press.

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015), "The Stan Math Library: Reverse-Mode Automatic Differentiation in C++," arXiv:1509.07164[cs.MS].

Gelman, A., Hwang, J., and Vehtari, A. (2014), "Understanding Predictive Information Criteria for Bayesian Models," *Statistics and Computing*, 24, 997-1016.

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y. S. (2008), "A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models," *Annals of Applied Statistics*, 2, 1360-1383.

Gelman, A., Lee, D., and Guo, J. (2015), "Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization," *Journal of Educational and Behavior Science* (in press).

Gelman, A., Meng, X.L., and Stern, H.S. (1996), Posterior Predictive Assessment of Model Fitness via Realized Discrepancies" (with discussion), *Statistica Sinica*, 6, 733-807.

Gelman, A., and Robert, C. (2013), "Not Only Defended but Also Applied: The Perceived Absurdity of Bayesian Inference" (with discussion), *The American Statistician*, 67, 1-5.

Gelman, A., and Shalizi, C. (2012), "Philosophy and the Practice of Bayesian Statistics" (with discussion), *British Journal of Mathematical and Statistical Psychology*, 66, 8-38.

Gelman, A., and Vehtari, A. (2014), "Bootstrap Averaging: Examples Where it Works and Where it Doesn't Work," Comment on "Estimation and Accuracy After Model Selection," by B. Efron, *Journal of the American Statistical Association*, 109, 1015-1016.

Hoffman, M. D., and Gelman, A. (2014), "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo," *Journal of Machine Learning Research*, 15, 1351-1381.

Liu, Y., Gelman, A., and Zheng, T. (2013), "Simulation-Efficient Shortest Probability Intervals," Technical Report, Columbia University, Dept. of Statistics.

McElreath, R. (2016), *Statistical Rethinking: A Bayesian Course with Examples in R and Stan,* Boca Raton, FL: Chapman & Hall/CRC.

National Oceanic and Atmospheric Administration (2016), *Quality Controlled Local Climatological Data (QCLCD)*. Available at http://www.ncdc.noaa.gov/qclcd.

Polson, N.G., and Scott, J.G. (2012), "On the Half-Cauchy Prior for a Global Scale Parameter," *Bayesian Analysis*, 7, 1-16.

Shirley, K. E., and Gelman, A. (2014), "Hierarchical Models for Estimating State and Demographic Trends in US Death Penalty Public Opinion," Journal of the Royal Statistical Society, *Series A*, 178, 1-28.

Stan Development Team (2016a), *RStan: The R interface to Stan* (version 2.9.0-3). Available at http://mc-stan.org/interfaces/rstan.

Stan Development Team (2016b), *Stan Modeling Language Users Guide and Reference Manual* (version 2.9.0). Available at http://mc-stan.org.

Stan Development Team (2016c), *Stan Wiki: Prior Choice Recommendations* (revision 8d41089). Available at https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations.

U. S. Energy Information Administration (2015), *An Assessment of Interval Data and Their Potential Application to Residential Electricity End-Use Modeling*. Available at http://www.eia.gov/consumption/residential/reports/smartmetering.

U. S. Energy Information Administration (2016), *2009 RECS Survey Data*. Available at http://www.eia.gov/consumption/residential/data/2009.

Wang, W., and Gelman, A. (2014), "Difficulty of Selecting Among Multilevel Models Using Predictive Accuracy," *Statistics and Its Interface*, 7: 1-8.