Practical 2 for SAE workshop

The following document is likely to become the LEMMA materials practical using StatJR for Small Area Estimation. There is too much material here for the remaining workshop time but we felt it was better to give you more material along with the memory stick so that you can try out all 3 examples in your own time after the workshop. For today we suggest you work through the practical as far as you can with our support and then work on the rest if you wish in your own time.

Small Area Estimation Practical using Stat-JR

This practical describes the functionality in Stat-JR for fitting Small Area Estimation (SAE) models and complements the Concepts and R practicals for the Small Area Estimation LEMMA module. In this practical we will use 3 data examples to illustrate some aspects of small area estimation. The first 'Toy' example will use the *tutorial* dataset which is an education dataset with pupils in schools and will be used to illustrate some of the general principles of small area estimation in a non-standard setting (where the areas are in fact schools). The second example is the *eusilc* dataset which is a more standard example where interest is in the average income in a series of geographical 'small' areas. It will be used to show an application of small area estimation on a larger dataset and also to illustrate the use of transformations (in this case the Box-Cox transform) in small area estimation. Finally the third example is the simulated *voteleave* dataset where interest lies in whether people will vote to leave or stay in the EU based on a sample of voting intentions from an exit poll and here this example is used to illustrate another typical usage of small area estimation but in particular to illustrate how small area estimation translates to other response types, in this case binary.

In this practical we will illustrate two interfaces into Stat-JR. We will first use the TREE interface to run the templates that have been written to do the SAE modelling directly and then we will show how these templates along with other existing templates have been combined into an eBook that can be used using the DEEP eBook interface to Stat-JR. We will begin with example 1.

Example 1 The tutorial dataset

The basic idea behind small area estimation is to estimate aggregated values of a variable at the level of a series of "small areas" e.g. what is the average income in each of a series of geographical subareas in a large dataset (see the second example) or what are the likely voting intentions and thus voting outcomes in each of a set of constituencies (see the third example). Here we begin with an example where actually the "small areas" are not areas but schools and this illustrates in fact that the term 'small area' is simply used to refer to groupings of observations into clusters – very much like higher levels in multilevel models.

We will here use the *tutorial* dataset that is commonly used in examples from our research centre and contains the results of 4,059 students from 65 schools in exam scores taken at age 16. In SAE we normally have 2 datasets – a census or population dataset which contains some variables (but not the variable of interest) for all observations in the population of interest, and a survey or sample dataset which contains the same variables as the population dataset for a subset of observations from the population with in addition the variable of interest.

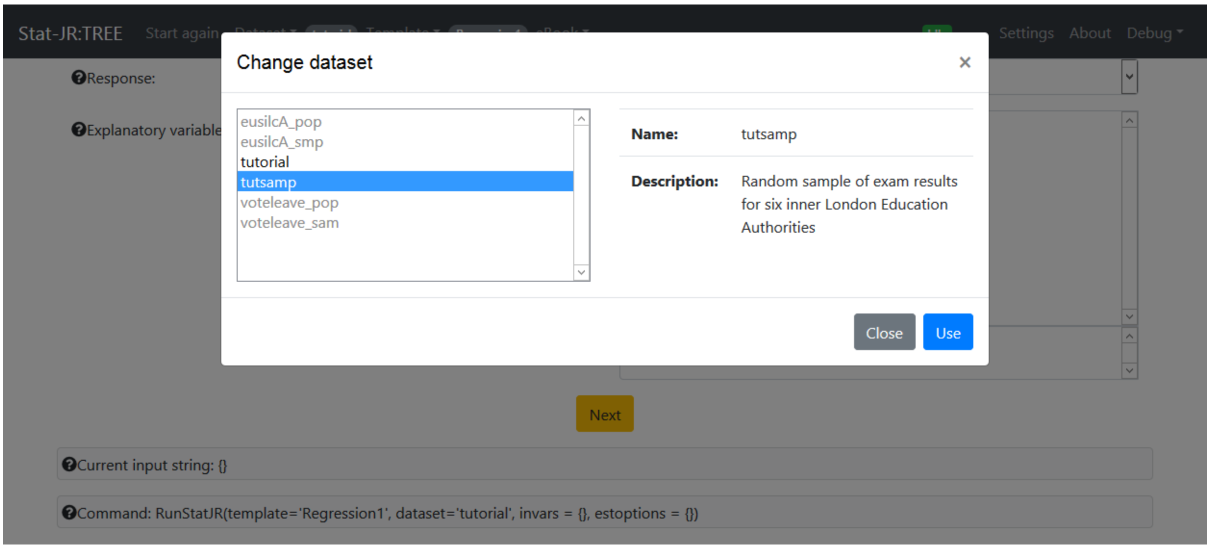
For illustration purposes here we are going to assume that the population is in fact the 4,059 students that are present in the tutorial dataset but that we have a roughly 10% sample dataset, *tut_smp* of 400 students. A possible scenario in education is that some form of intervention has been carried out for this sample and their exam scores are known but we are interested in finding out what the impact of the intervention might be for all pupils and what the average effect would be for each school.

Such data would be collected for real in the UK by for example projects funded by the Education Endowment Foundation (EEF) that looks at the effect of various school-based interventions. Often when people test for the impact of the intervention the scores on standard tests/exams are used.

So here we will look at our example:

Firstly we need to start up Stat-JR TREE so insert your memory sticks in your machine and in the home directory of the stick you should find a short cut to *TREE.exe* which if you click on it you should find that Stat-JR TREE starts up in your web browser.

We now need to change dataset so click on **Dataset** and **Choose** and select *tutsamp* from the list as shown below:



Response:

Explanatory variable

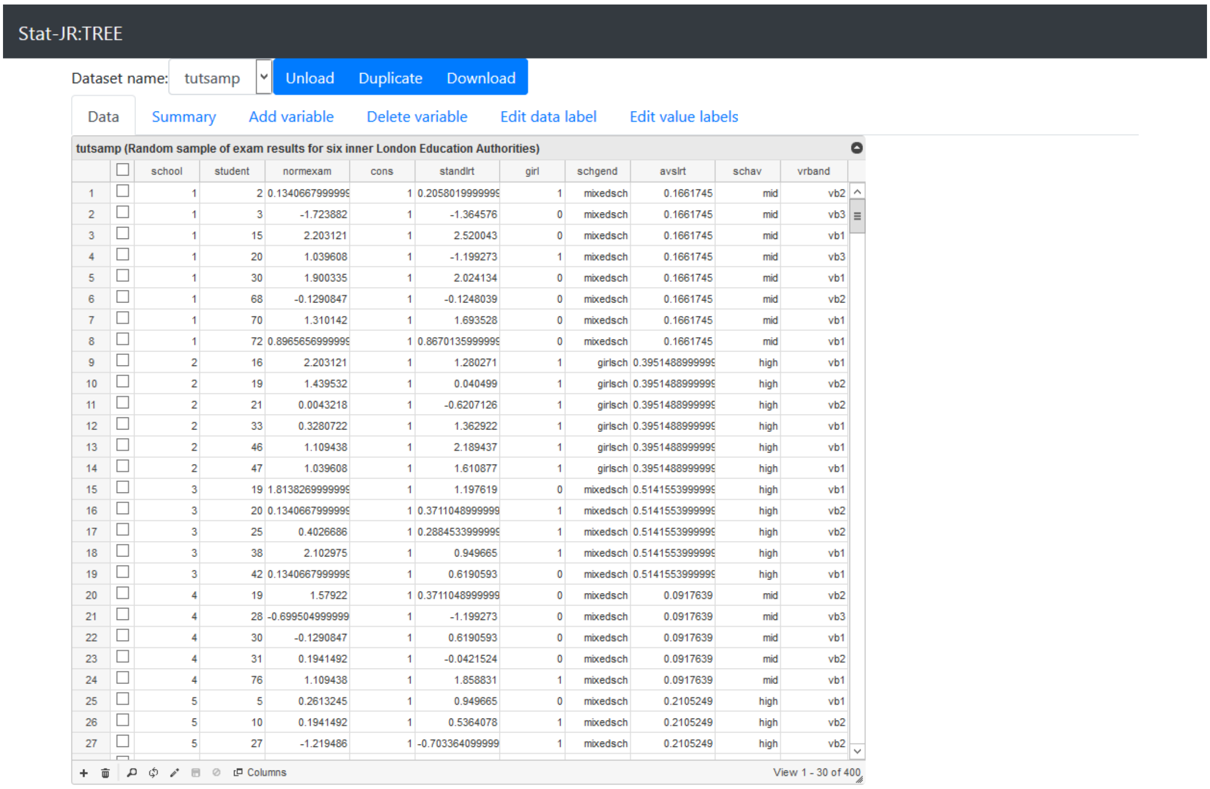
Name: tutsamp

Description: Random sample of exam results for six inner London Education Authorities

Current input string: {}

Command: RunStatJR(template='Regression1', dataset='tutorial', invars = {}, estoptions = {})

Now click on **Use** and this dataset will be selected. If we wish to view the dataset then choose **View** from the **Dataset** menu and we get the following in a fresh tab:



tutsamp (Random sample of exam results for six inner London Education Authorities)

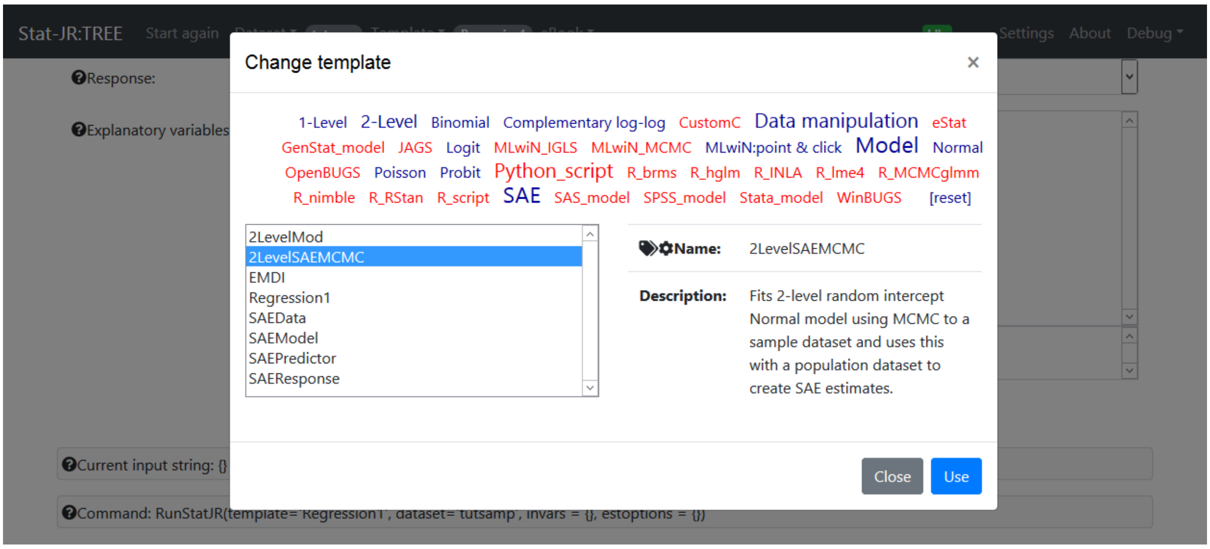
	□	school	student	normexam	cons	standlrt	girl	schgend	avslrt	schav	vrband
1	□	1	2	0.1340667999999	1	0.2058019999999	1	mixedsch	0.1661745	mid	vb2
2	□	1	3	-1.723882	1	-1.364576	0	mixedsch	0.1661745	mid	vb3
3	□	1	15	2.203121	1	2.520043	0	mixedsch	0.1661745	mid	vb1
4	□	1	20	1.039608	1	-1.199273	1	mixedsch	0.1661745	mid	vb3
5	□	1	30	1.900335	1	2.024134	0	mixedsch	0.1661745	mid	vb1
6	□	1	68	-0.1290847	1	-0.1248039	0	mixedsch	0.1661745	mid	vb2
7	□	1	70	1.310142	1	1.693528	0	mixedsch	0.1661745	mid	vb1
8	□	1	72	0.8965656999999	1	0.8670135999999	0	mixedsch	0.1661745	mid	vb1
9	□	2	16	2.203121	1	1.280271	1	girlsch	0.3951488999999	high	vb1
10	□	2	19	1.439532	1	0.040499	1	girlsch	0.3951488999999	high	vb2
11	□	2	21	0.0043218	1	-0.6207126	1	girlsch	0.3951488999999	high	vb2
12	□	2	33	0.3280722	1	1.362922	1	girlsch	0.3951488999999	high	vb1
13	□	2	46	1.109438	1	2.189437	1	girlsch	0.3951488999999	high	vb1
14	□	2	47	1.039608	1	1.610877	1	girlsch	0.3951488999999	high	vb1
15	□	3	19	1.8138269999999	1	1.197619	0	mixedsch	0.5141553999999	high	vb1
16	□	3	20	0.1340667999999	1	0.3711048999999	1	mixedsch	0.5141553999999	high	vb2
17	□	3	25	0.4026686	1	0.2884533999999	1	mixedsch	0.5141553999999	high	vb2
18	□	3	38	2.102975	1	0.949665	1	mixedsch	0.5141553999999	high	vb1
19	□	3	42	0.1340667999999	1	0.6190593	0	mixedsch	0.5141553999999	high	vb1
20	□	4	19	1.57922	1	0.3711048999999	0	mixedsch	0.0917639	mid	vb2
21	□	4	28	-0.699504999999	1	-1.199273	0	mixedsch	0.0917639	mid	vb3
22	□	4	30	-0.1290847	1	0.6190593	0	mixedsch	0.0917639	mid	vb1
23	□	4	31	0.1941492	1	-0.0421524	0	mixedsch	0.0917639	mid	vb2
24	□	4	76	1.109438	1	1.858831	1	mixedsch	0.0917639	mid	vb1
25	□	5	5	0.2613245	1	0.949665	0	mixedsch	0.2105249	high	vb1
26	□	5	10	0.1941492	1	0.5364078	1	mixedsch	0.2105249	high	vb2
27	□	5	27	-1.219486	1	-0.703364099999	1	mixedsch	0.2105249	high	vb2

View 1 - 30 of 400

Here there are a large number of variables including the variable that we are interested in which is called *normexam.* This is the total points score across the GCSE exams for these students but it has then been transformed by a normalising transformation i.e. the scores have been translated after sorting to the equivalent quantiles of a standard normal distribution which means it should follow a normal distribution with mean 0 and variance 1 (actually in practice this was done for the 4,059 students rather than the sample of 400).

Now we also see potential predictor variables for this outcome but here we will use just two: *girl* (which is a born-sex related binary variable taking values of 1 for girls and 0 for boys), and *standlrt* (which is an intake Reading test score for the students). The idea therefore would be to create a model that relates the *normexam* scores to these 2 predictor variables (as well as *school*) and then use this model to predict scores for the 3659 students who are not in the sample. In practice we often cannot identify which specific observations are in the sample so we actually predict scores for all 4,059 students.

To fit the model we can use the template *2levelSAEMCMC* which we can select by returning to the original tab and clicking on **Template** and **Choose** and selecting from the list thus:



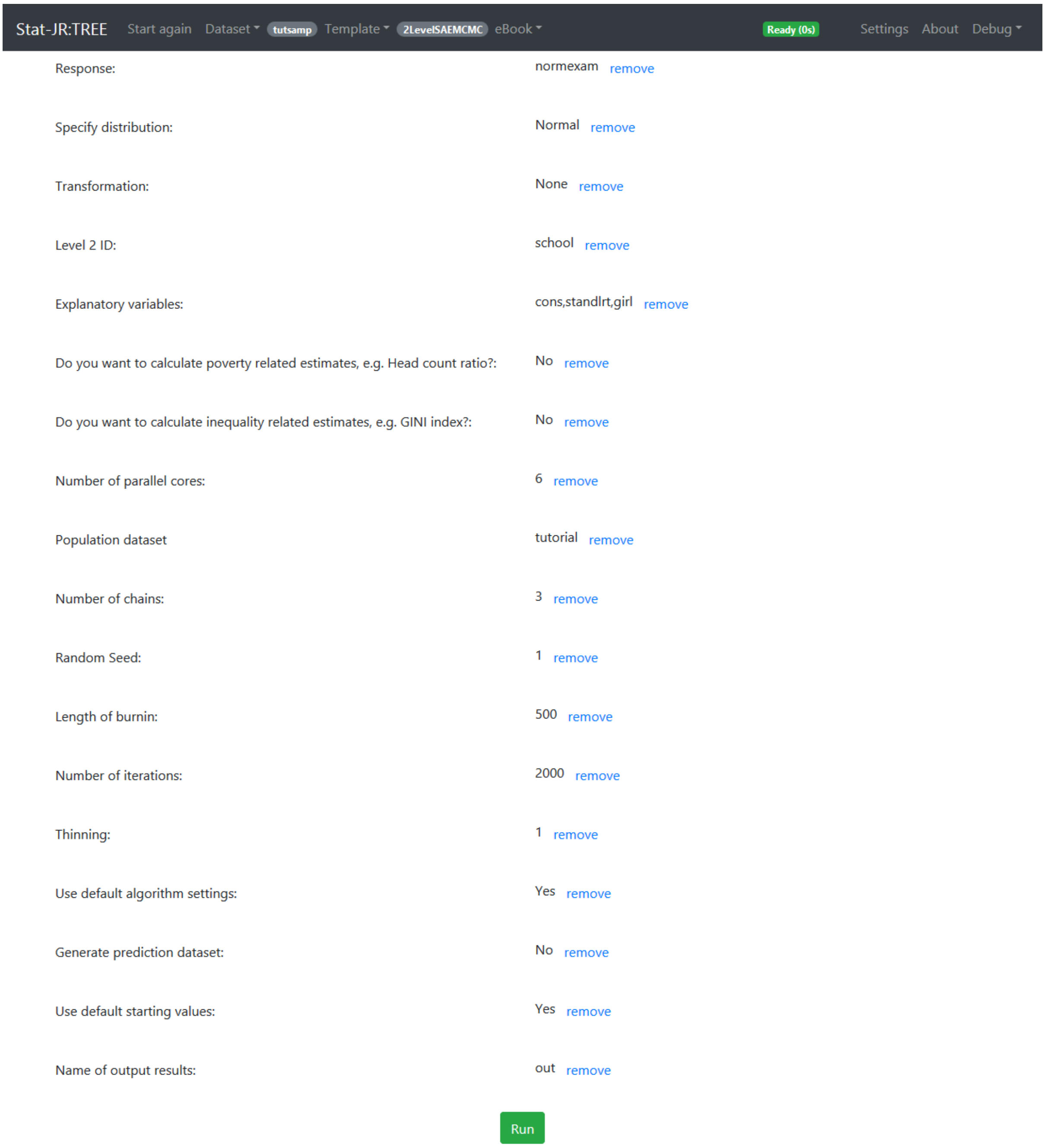
Name: 2LevelSAEMCMC

Description: Fits 2-level random intercept Normal model using MCMC to a sample dataset and uses this with a population dataset to create SAE estimates.

Current input string: {}

Command: RunStatJR(template='Regression1', dataset='tutsamp', invars = {}, estoptions = {})

Clicking **Use**, we can then fill the inputs in as follows:



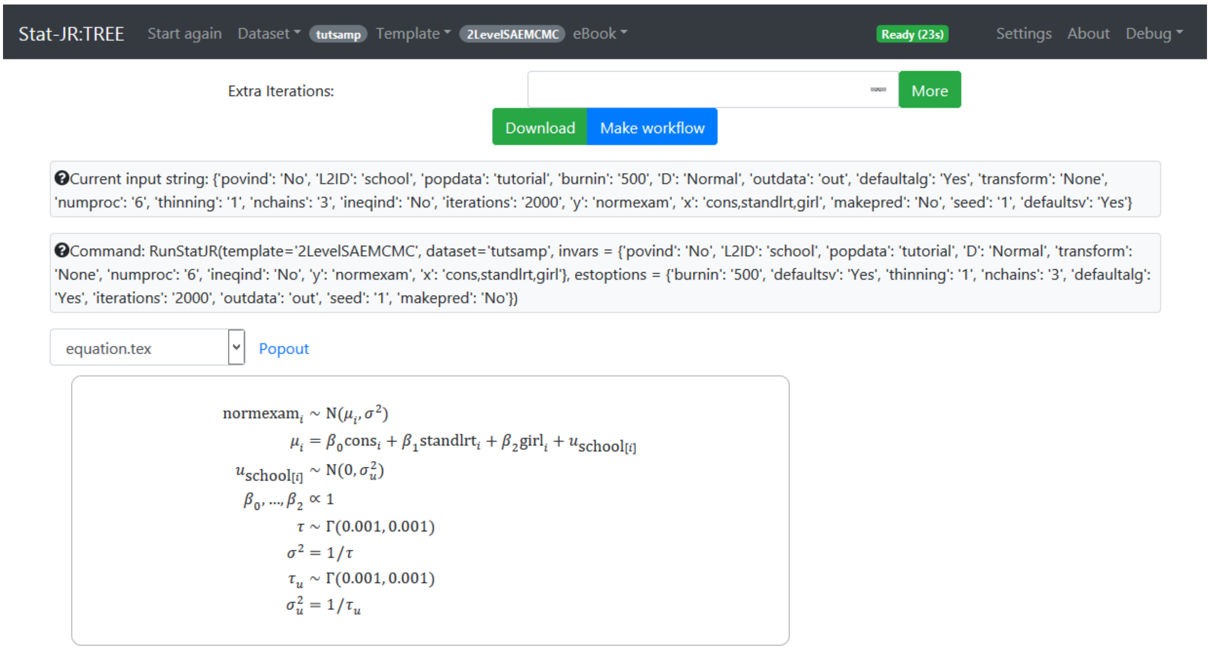
Response:	normexam	remove
Specify distribution:	Normal	remove
Transformation:	None	remove
Level 2 ID:	school	remove
Explanatory variables:	cons,standlrt,girl	remove
Do you want to calculate poverty related estimates, e.g. Head count ratio?:	No	remove
Do you want to calculate inequality related estimates, e.g. GINI index?:	No	remove
Number of parallel cores:	6	remove
Population dataset	tutorial	remove
Number of chains:	3	remove
Random Seed:	1	remove
Length of burnin:	500	remove
Number of iterations:	2000	remove
Thinning:	1	remove
Use default algorithm settings:	Yes	remove
Generate prediction dataset:	No	remove
Use default starting values:	Yes	remove
Name of output results:	out	remove

Here we see that we have chosen *normexam* as our response variable of interest and told Stat-JR that we wish to fit a *Normal* response variable to an untransformed (choice *None*) response. We indicate that the small area indicator (Level 2 ID) in this case is *school.* We then tell Stat-JR which predictor variables to use in our small area estimation model which here is *cons*, *standlrt* and *girl.* Here *cons* is a column of 1s and is used to indicate we wish an intercept in the multilevel regression model used within the SAE modelling. Small Area Estimation models are often used with salary data (see example 2) and so the template can construct various poverty/inequality indices but here as we have an education example we say not to calculate them. The template in Stat-JR allows parallel processing to speed up the MCMC estimation so here we will indicate 6 for the number of cores to use which will be the default in other templates we use later. There are then several estimation inputs which for now we will use the default values for by clicking on **Next** twice and typing *out* for the name of the output file to use.

All being well the input string given after the inputs in the current input string box will be as follows:

Current input string: {'povind': 'No', 'L2ID': 'school', 'popdata': 'tutorial', 'burnin': '500', 'D': 'Normal', 'outdata': 'out', 'defaultalg': 'Yes', 'transform': 'None', 'numproc': '6', 'thinning': '1', 'nchains': '3', 'ineqind': 'No', 'iterations': '2000', 'y': 'normexam', 'x': 'cons,standlrt,girl', 'makepred': 'No', 'seed': '1', 'defaultsv': 'Yes'}

If we then click on **Run** then Stat-JR will run the model with these inputs. This will involve compiling the code to fit the model, fitting it using MCMC estimation and producing small area estimates. When the model has finished running the screen should look something like the following (if you scroll to the bottom):

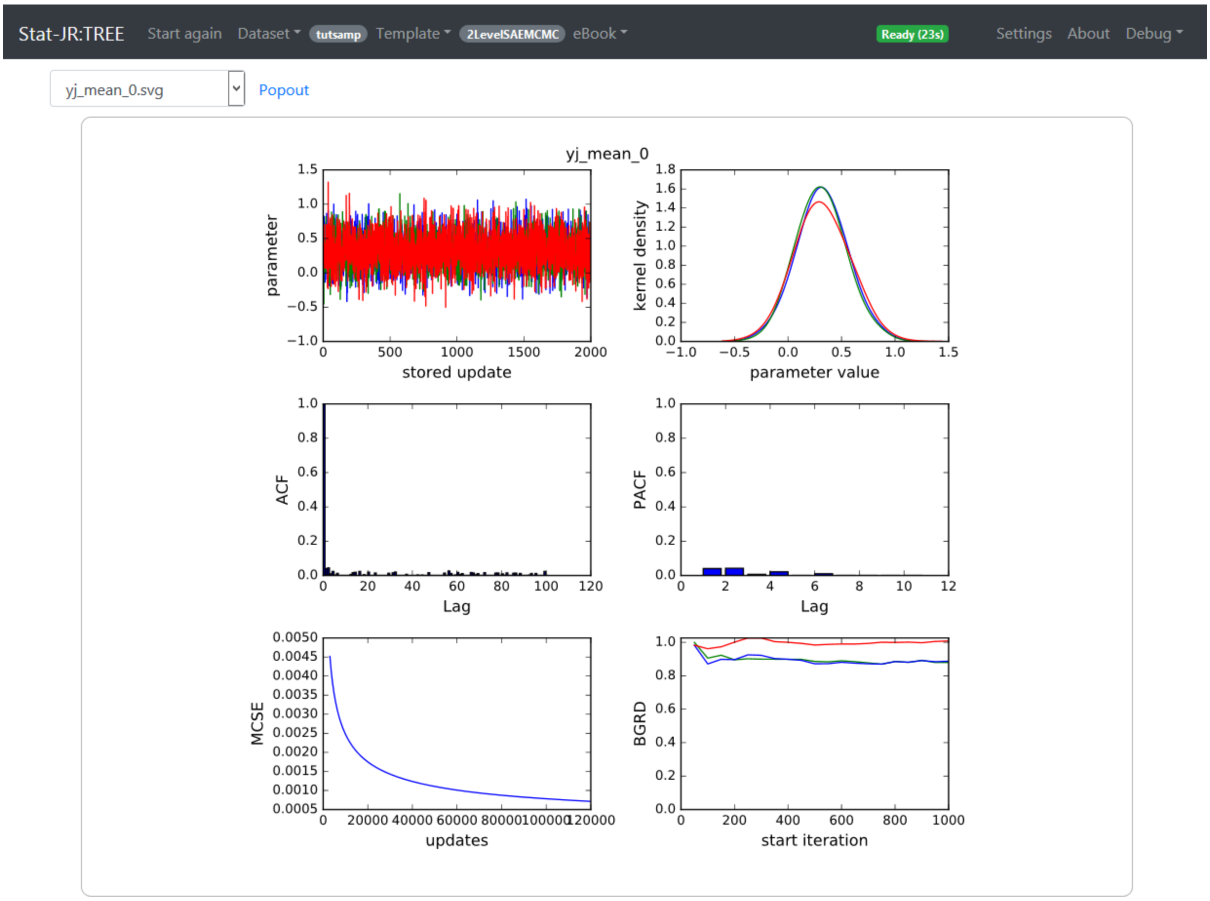


Current input string: {'povind': 'No', 'L2ID': 'school', 'popdata': 'tutorial', 'burnin': '500', 'D': 'Normal', 'outdata': 'out', 'defaultalg': 'Yes', 'transform': 'None', 'numproc': '6', 'thinning': '1', 'nchains': '3', 'ineqind': 'No', 'iterations': '2000', 'y': 'normexam', 'x': 'cons,standlrt,girl', 'makepred': 'No', 'seed': '1', 'defaultsv': 'Yes'}

Command: RunStatJR(template='2LevelSAEMCMC', dataset='tutsamp', invars = {'povind': 'No', 'L2ID': 'school', 'popdata': 'tutorial', 'D': 'Normal', 'transform': 'None', 'numproc': '6', 'ineqind': 'No', 'y': 'normexam', 'x': 'cons,standlrt,girl'}, estoptions = {'burnin': '500', 'defaultsv': 'Yes', 'thinning': '1', 'nchains': '3', 'defaultalg': 'Yes', 'iterations': '2000', 'outdata': 'out', 'seed': '1', 'makepred': 'No'})

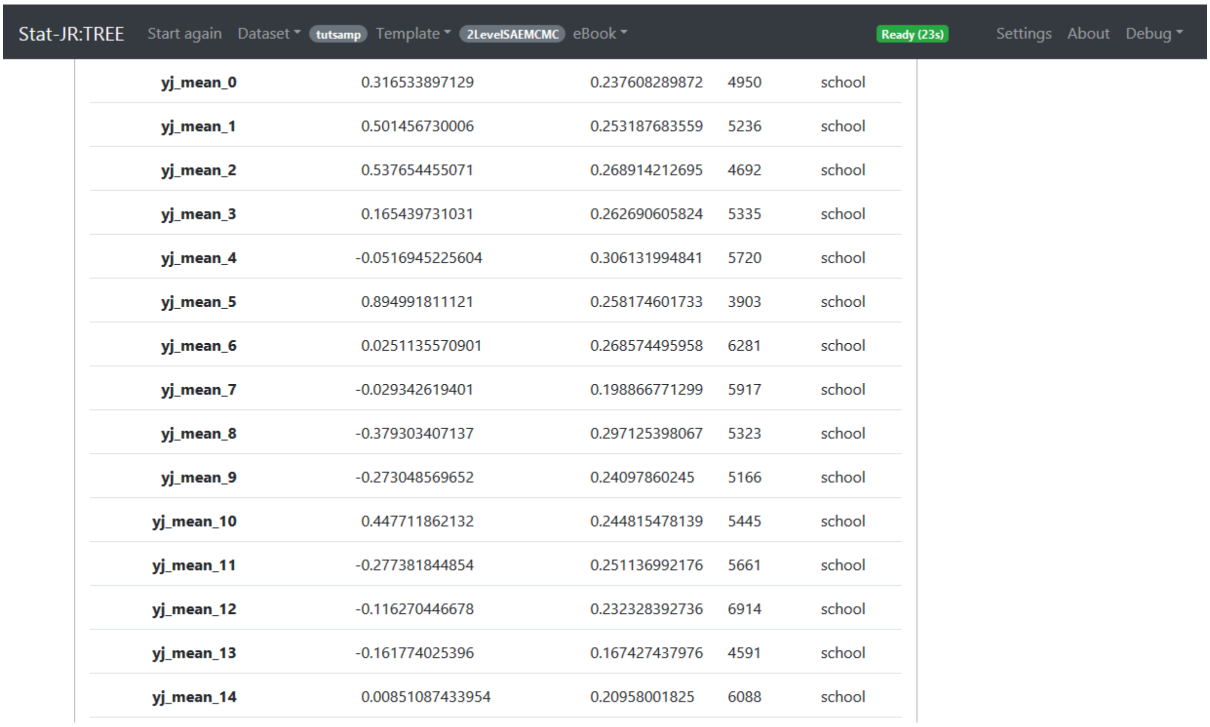
$$\begin{aligned}
\mathrm{normexam}_i &\sim \mathrm{N}(\mu_i, \sigma^2)\\
\mu_i &= \beta_0\mathrm{cons}_i + \beta_1\mathrm{standlrt}_i + \beta_2\mathrm{girl}_i + u_{\mathrm{school}[i]}\\
u_{\mathrm{school}[i]} &\sim \mathrm{N}(0, \sigma_u^2)\\
\beta_0, \ldots, \beta_2 &\propto 1\\
\tau &\sim \Gamma(0.001, 0.001)\\
\sigma^2 &= 1/\tau\\
\tau_u &\sim \Gamma(0.001, 0.001)\\
\sigma_u^2 &= 1/\tau_u
\end{aligned}$$

You will see that to the right at the top it says *Ready* with a time in seconds and this indicates that the model running has finished and how long it took. At the bottom of the screen is a pull-down list of the objects created by the template with one chosen – in this case *equation.tex* which shows the model that was fitted to the sample dataset. We can now view some of the other outputs to see what has actually happened in this template so if you click on the pull down list there are lots of outputs but if you choose *yj_mean_0.svg* you should (after a short pause for compilation) see the following:



Here we see some MCMC estimation diagnostics for the estimate of the mean of the first school (the template counts from 0 rather than 1 when numbering the schools). Essentially the template fits a model to the 400 pupils in the sample dataset and uses MCMC estimation which consists of a series of random draws for parameter estimates (from their appropriate posterior distributions) for each of a series of iterations. At each iteration the template then in addition draws an exam score for each of the 4,059 pupils in the larger population dataset. For this first school there are 8 pupils who happen to be in the sample dataset and 73 in the population dataset. So at each iteration the template is generating scores for the 73 pupils and then it uses these marks to create small area (school) estimates. So in the plots above we can see in the top left plot 2000x3 estimates (due to the MCMC running 3 chains in parallel each for 2,000 iterations) of the average score for the first school. We see that for this school the average score is more often positive although as shown in the Kernel density plots to the top right there are quite a few occasions when the average is predicted to be negative.

To better interpret this an estimate that is positive reflects a school where on average pupils do better than the average pupil in the population and negative means the school does on average worse than the average of the population. So for this school our estimate is positive meaning better than average but we are not 100% confident that this is true as for a percentage of iterations the estimate is negative. These MCMC diagnostics plots are perhaps not the best route to looking at the estimates and so if we scroll down the list (or type the first letters of the name into the pull down list) we will find the output **ModelParameters** which we can choose and which looks as follows (if we scroll down the list a bit):

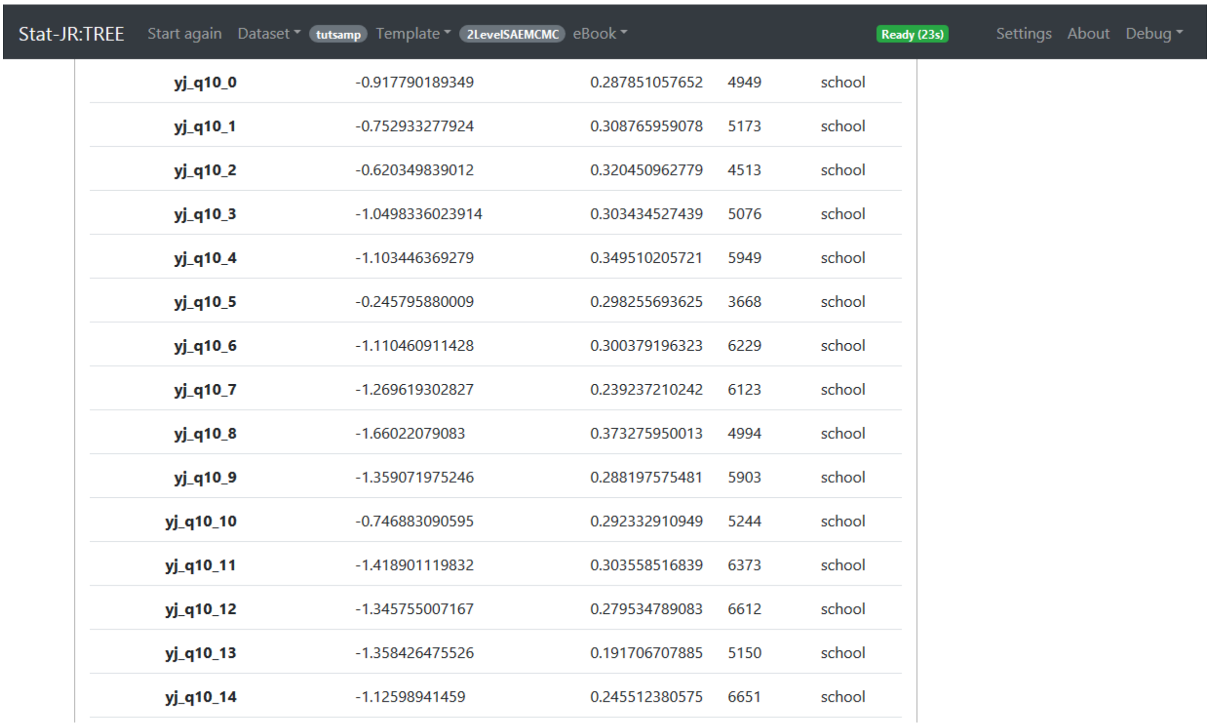


yj_mean_0	0.316533897129	0.237608289872	4950	school
yj_mean_1	0.501456730006	0.253187683559	5236	school
yj_mean_2	0.537654455071	0.268914212695	4692	school
yj_mean_3	0.165439731031	0.262690605824	5335	school
yj_mean_4	-0.0516945225604	0.306131994841	5720	school
yj_mean_5	0.894991811121	0.258174601733	3903	school
yj_mean_6	0.0251135570901	0.268574495958	6281	school
yj_mean_7	-0.029342619401	0.198866771299	5917	school
yj_mean_8	-0.379303407137	0.297125398067	5323	school
yj_mean_9	-0.273048569652	0.24097860245	5166	school
yj_mean_10	0.447711862132	0.244815478139	5445	school
yj_mean_11	-0.277381844854	0.251136992176	5661	school
yj_mean_12	-0.116270446678	0.232328392736	6914	school
yj_mean_13	-0.161774025396	0.167427437976	4591	school
yj_mean_14	0.00851087433954	0.20958001825	6088	school

Here we can see the long list of model parameters and for each 3 summary columns, the *mean* which is a point estimate for the parameter, the *sd* which is the posterior sd, a measure of the variability in the point estimate and an estimate of the standard error of the parameter and the *ESS* which is a diagnostic indicating whether the MCMC estimation has been run for long enough (with larger values being better).

Here we see that the estimate for the first school (*yj_mean_0*) is 0.317 with an SD of 0.238 i.e. we have the positive effect we saw in the diagnostic plot but as the estimate is not bigger than 1.96 times the SD then this effect is not significantly different from 0. The second school (*yj_mean_1*) has an estimate of 0.501 with an SD of 0.253 so again the estimate is above average but just about significant in this case. As we look down the first 12 schools we see that 7 have positive effects, 5 have negative but only school 6 *(yj_mean_5*) has an effect that is clearly statistically different from 0. This is perhaps not so surprising as we have data on only 400 children overall in our sample so we might expect quite a bit of uncertainty.

Within this **ModelParameters** output we can scroll down and find other small area statistics for our schools. So here we have scrolled down as shown in the screen shot next until we have come across the estimates for the 10th percentile for each school. These are estimates of the mark below which we would expect to find only 10% of children in the school and above which we would find 90%. The template constructs these estimates by at each iteration taking the estimated scores for all the 4,059 children and for each school taking its children's score sorting them and finding (often with some interpolation) the score that is in the 10th percentile position.



yj_q10_0	-0.917790189349	0.287851057652	4949	school
yj_q10_1	-0.752933277924	0.308765959078	5173	school
yj_q10_2	-0.620349839012	0.320450962779	4513	school
yj_q10_3	-1.0498336023914	0.303434527439	5076	school
yj_q10_4	-1.103446369279	0.349510205721	5949	school
yj_q10_5	-0.245795880009	0.298255693625	3668	school
yj_q10_6	-1.110460911428	0.300379196323	6229	school
yj_q10_7	-1.269619302827	0.239237210242	6123	school
yj_q10_8	-1.66022079083	0.373275950013	4994	school
yj_q10_9	-1.359071975246	0.288197575481	5903	school
yj_q10_10	-0.746883090595	0.292332910949	5244	school
yj_q10_11	-1.418901119832	0.303558516839	6373	school
yj_q10_12	-1.345755007167	0.279534789083	6612	school
yj_q10_13	-1.358426475526	0.191706707885	5150	school
yj_q10_14	-1.12598941459	0.245512380575	6651	school

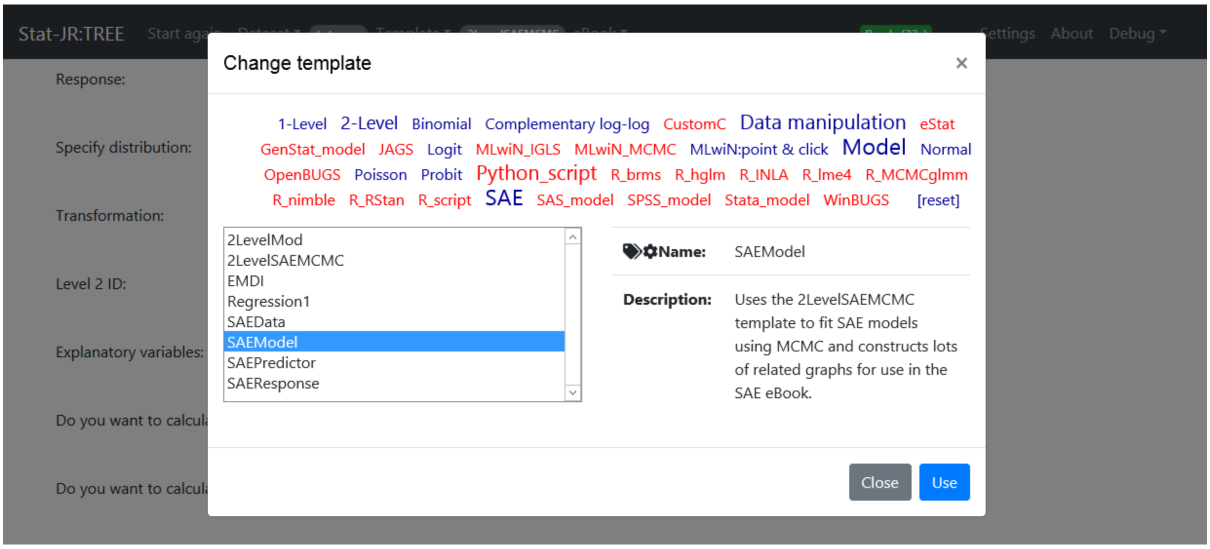
Here perhaps unsurprisingly we see negative values for all schools as it would be an unusual school that had 90% of its children doing better than the national average. The closest of the schools here is school 6 (*yj_q10_5*) which has a 10th quantile estimate of -0.246 which is only slightly negative.

There are many other small area statistics that are calculated but at this point you are perhaps wondering why you are having to look at such an untidy output object and why we have not improved on this output. In fact this template, *2LevelSAEMCMC*, is something of a building block and we have improved outputs in a second template called *SAEModel*.

Using the SAEModel template

One feature of Stat-JR is that we can use templates as building blocks to produce further templates. The *2LevelSAEMCMC* template is a powerful template in terms of flexible model fitting but is primarily designed for just the model fitting. We are therefore now going to move onto a second template called *SAEModel* which is a type of template that we call a “super” template in that it calls other templates from within its code and pieces the outputs together. In this case *SAEModel* will call *2LevelSAEMCMC* to do the model fitting but then will do some work to make the outputs more intelligible. This template has primarily been developed to be used within the SAE eBook that we will come on to later when we discuss the DEEP interface to Stat-JR.

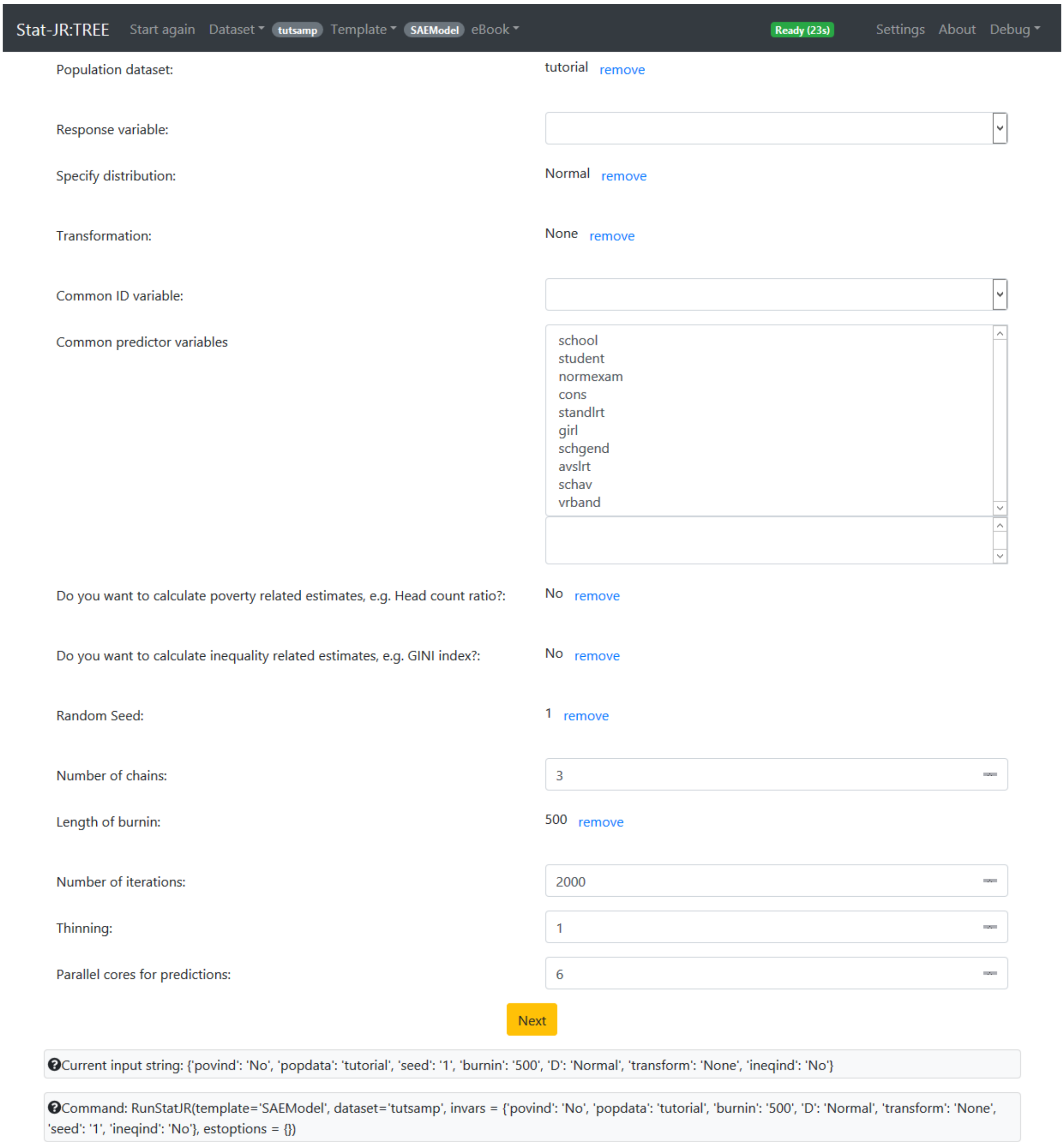
To get started we need to switch templates so click on **Templates** and **Choose** and select *SAEModel* from the list



Name: SAEModel

Description: Uses the 2LevelSAEMCMC template to fit SAE models using MCMC and constructs lots of related graphs for use in the SAE eBook.

Clicking on **Use** will allow us to now use this template and specify inputs for it. The screen will initially look as follows:



Population dataset: tutorial

Response variable:

Specify distribution: Normal

Transformation: None

Common ID variable:

Common predictor variables
school
student
normexam
cons
standlrt
girl
schgend
avslrt
schav
vrband

Do you want to calculate poverty related estimates, e.g. Head count ratio?: No

Do you want to calculate inequality related estimates, e.g. GINI index?: No

Random Seed: 1

Number of chains: 3

Length of burnin: 500

Number of iterations: 2000

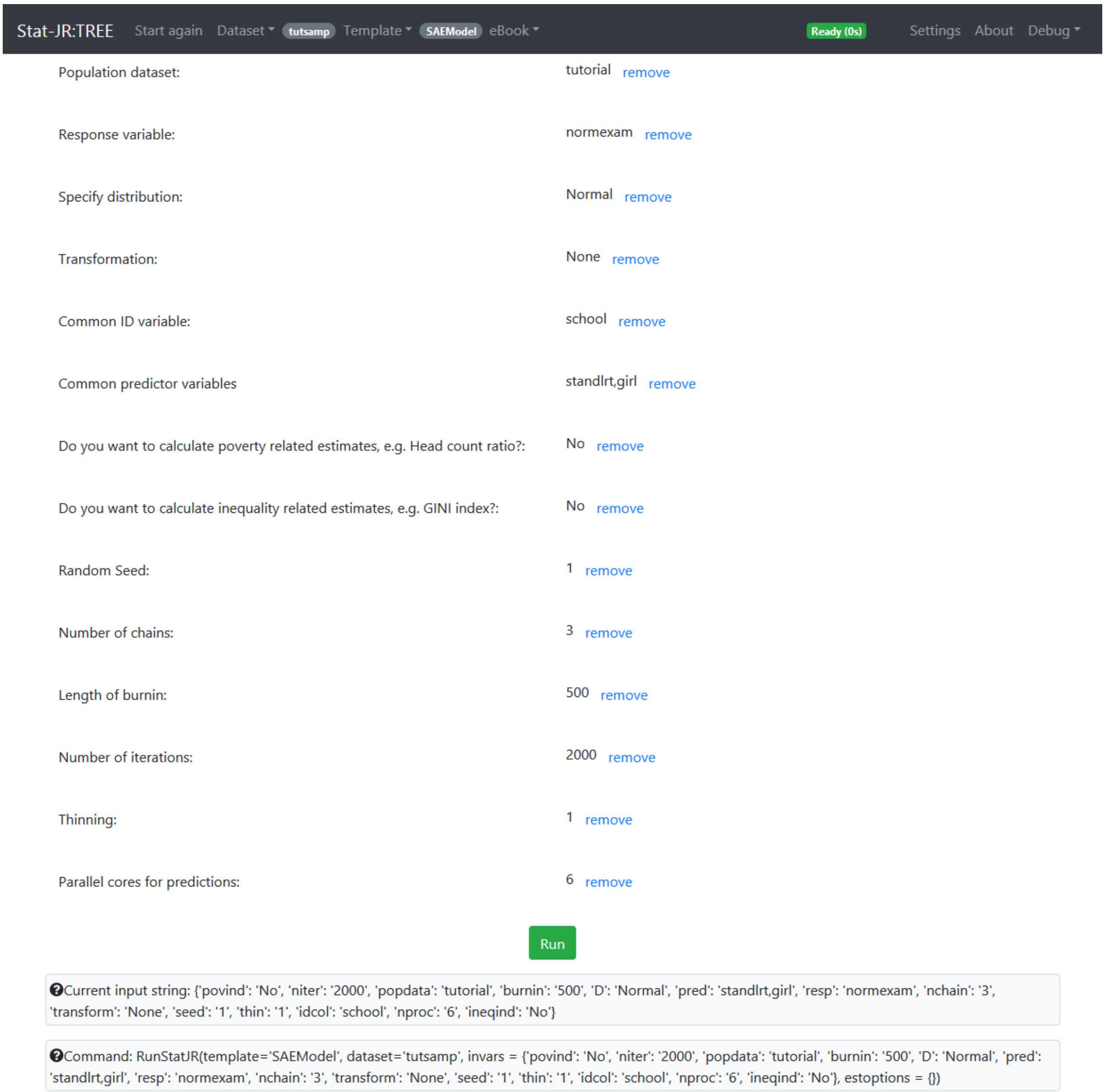
Thinning: 1

Parallel cores for predictions: 6

```
Current input string: {'povind': 'No', 'popdata': 'tutorial', 'seed': '1', 'burnin': '500', 'D': 'Normal', 'transform': 'None', 'ineqind': 'No'}
```

```
Command: RunStatJR(template='SAEModel', dataset='tutsamp', invars = {'povind': 'No', 'popdata': 'tutorial', 'burnin': '500', 'D': 'Normal', 'transform': 'None', 'seed': '1', 'ineqind': 'No'}, estoptions = {})
```

Here you will see that some inputs have been filled in for us and it is the case when templates share the same input names as each other that they are transferred across. We will here have to fill in the ones that are missing and one distinction here from the last template is that this template will automatically add an intercept into the model so we do NOT add *cons* in the predictor list. So now we fill in the missing inputs as follows:



Population dataset: tutorial remove

Response variable: normexam remove

Specify distribution: Normal remove

Transformation: None remove

Common ID variable: school remove

Common predictor variables standlrt,girl remove

Do you want to calculate poverty related estimates, e.g. Head count ratio?: No remove

Do you want to calculate inequality related estimates, e.g. GINI index?: No remove

Random Seed: 1 remove

Number of chains: 3 remove

Length of burnin: 500 remove

Number of iterations: 2000 remove

Thinning: 1 remove

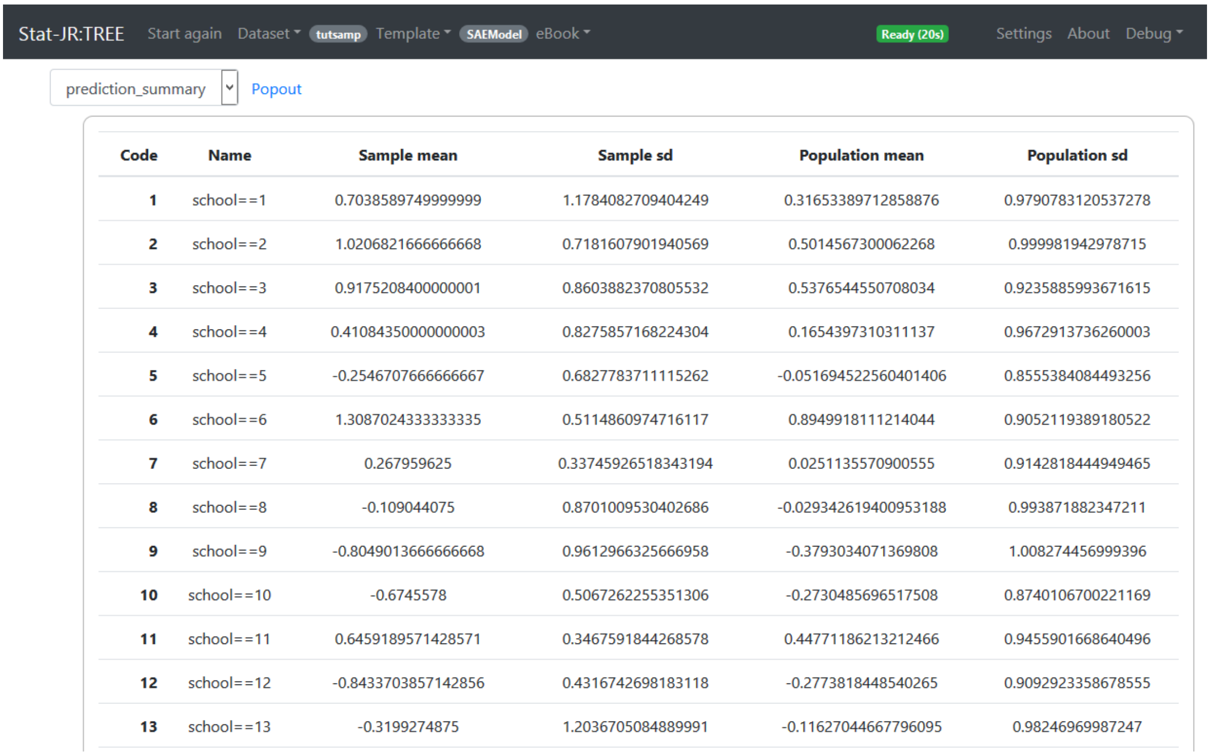
Parallel cores for predictions: 6 remove

```
?Current input string: {'povind': 'No', 'niter': '2000', 'popdata': 'tutorial', 'burnin': '500', 'D': 'Normal', 'pred': 'standlrt,girl', 'resp': 'normexam', 'nchain': '3', 'transform': 'None', 'seed': '1', 'thin': '1', 'idcol': 'school', 'nproc': '6', 'ineqind': 'No'}
```

```
?Command: RunStatJR(template='SAEModel', dataset='tutsamp', invars = {'povind': 'No', 'niter': '2000', 'popdata': 'tutorial', 'burnin': '500', 'D': 'Normal', 'pred': 'standlrt,girl', 'resp': 'normexam', 'nchain': '3', 'transform': 'None', 'seed': '1', 'thin': '1', 'idcol': 'school', 'nproc': '6', 'ineqind': 'No'}, estoptions = {})
```

When this is done we click on **Run** and we can wait for the model fitting to be executed. This will finish when the counter to the top right changes to *Ready* with an execution time. We can next scroll to the bottom of the screen and see what outputs this template produces.

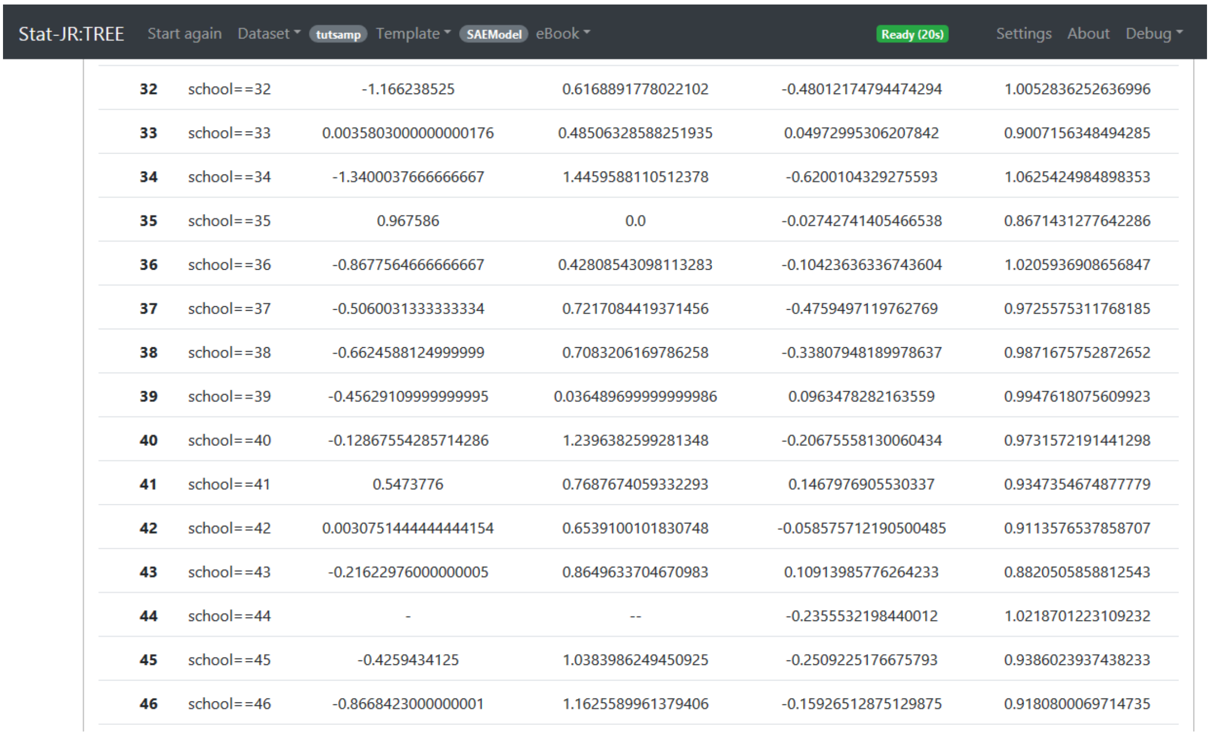
First if we select *prediction_summary* from the list we will see the following:



Code	Name	Sample mean	Sample sd	Population mean	Population sd
1	school==1	0.7038589749999999	1.1784082709404249	0.31653389712858876	0.9790783120537278
2	school==2	1.0206821666666668	0.7181607901940569	0.5014567300062268	0.999981942978715
3	school==3	0.9175208400000001	0.8603882370805532	0.5376544550708034	0.9235885993671615
4	school==4	0.41084350000000003	0.8275857168224304	0.1654397310311137	0.9672913736260003
5	school==5	-0.2546707666666667	0.6827783711115262	-0.051694522560401406	0.8555384084493256
6	school==6	1.3087024333333335	0.5114860974716117	0.8949918111214044	0.9052119389180522
7	school==7	0.267959625	0.33745926518343194	0.0251135570900555	0.9142818444949465
8	school==8	-0.109044075	0.8701009530402686	-0.029342619400953188	0.993871882347211
9	school==9	-0.8049013666666668	0.9612966325666958	-0.3793034071369808	1.008274456999396
10	school==10	-0.6745578	0.5067262255351306	-0.2730485696517508	0.8740106700221169
11	school==11	0.6459189571428571	0.3467591844268578	0.44771186213212466	0.9455901668640496
12	school==12	-0.8433703857142856	0.4316742698183118	-0.2773818448540265	0.9092923358678555
13	school==13	-0.3199274875	1.2036705084889991	-0.11627044667796095	0.98246969987247

Here we see the results for the different schools (and this list now uses the actual codes of the schools rather than a list starting at 0). In the **Name** column if the dataset had names we would see them here (see the second example later) but here we simply have the numbers converted to names e.g. *school==1.* There are then 4 columns of small area statistics, the third column population mean is one that we have seen already and happily we see the estimated value of 0.317 for school 1 we have seen earlier. The column to the right, *population sd* is as it suggests an estimate of the standard deviation of scores in school 1. Note this is different from the posterior sd for the mean we saw earlier which is in fact an estimate of the standard error of the mean and NOT the standard deviation for the school.

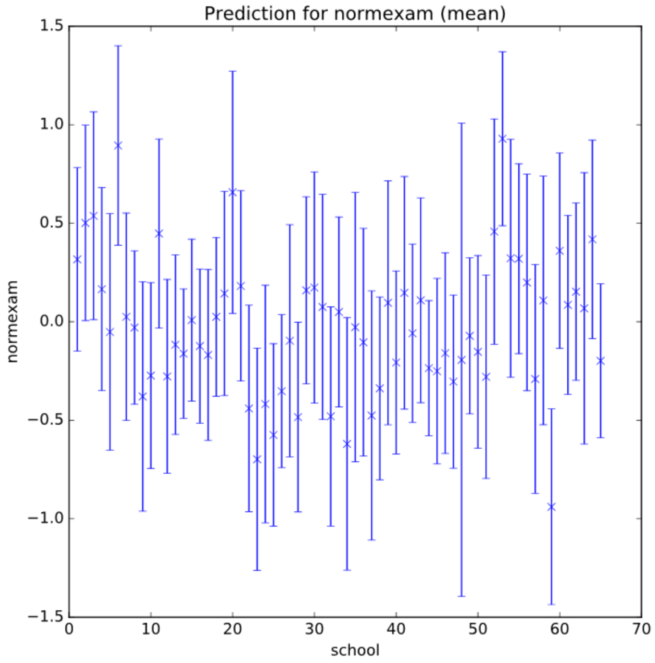
The two columns on the left are useful comparisons and are alternative estimates that are derived direct from the sample data. In other words the sample mean is simply the mean of the data **observed** in the sample and the sample sd is the standard deviation of the data observed in the sample. To see this more clearly we can scroll down and look at some of the later schools:



32	school==32	-1.166238525	0.6168891778022102	-0.48012174794474294	1.0052836252636996
33	school==33	0.0035803000000000176	0.48506328588251935	0.04972995306207842	0.9007156348494285
34	school==34	-1.3400037666666667	1.4459588110512378	-0.6200104329275593	1.0625424984898353
35	school==35	0.967586	0.0	-0.02742741405466538	0.8671431277642286
36	school==36	-0.8677564666666667	0.42808543098113283	-0.10423636336743604	1.0205936908656847
37	school==37	-0.5060031333333334	0.7217084419371456	-0.4759497119762769	0.9725575311768185
38	school==38	-0.6624588124999999	0.7083206169786258	-0.33807948189978637	0.9871675752872652
39	school==39	-0.45629109999999995	0.036489699999999986	0.0963478282163559	0.9947618075609923
40	school==40	-0.12867554285714286	1.2396382599281348	-0.20675558130060434	0.9731572191441298
41	school==41	0.5473776	0.7687674059332293	0.1467976905530337	0.9347354674877779
42	school==42	0.0030751444444444154	0.6539100101830748	-0.058575712190500485	0.9113576537858707
43	school==43	-0.21622976000000005	0.8649633704670983	0.10913985776264233	0.8820505858812543
44	school==44	-	--	-0.2355532198440012	1.0218701223109232
45	school==45	-0.4259434125	1.0383986249450925	-0.2509225176675793	0.9386023937438233
46	school==46	-0.8668423000000001	1.1625589961379406	-0.15926512875129875	0.918080069714735

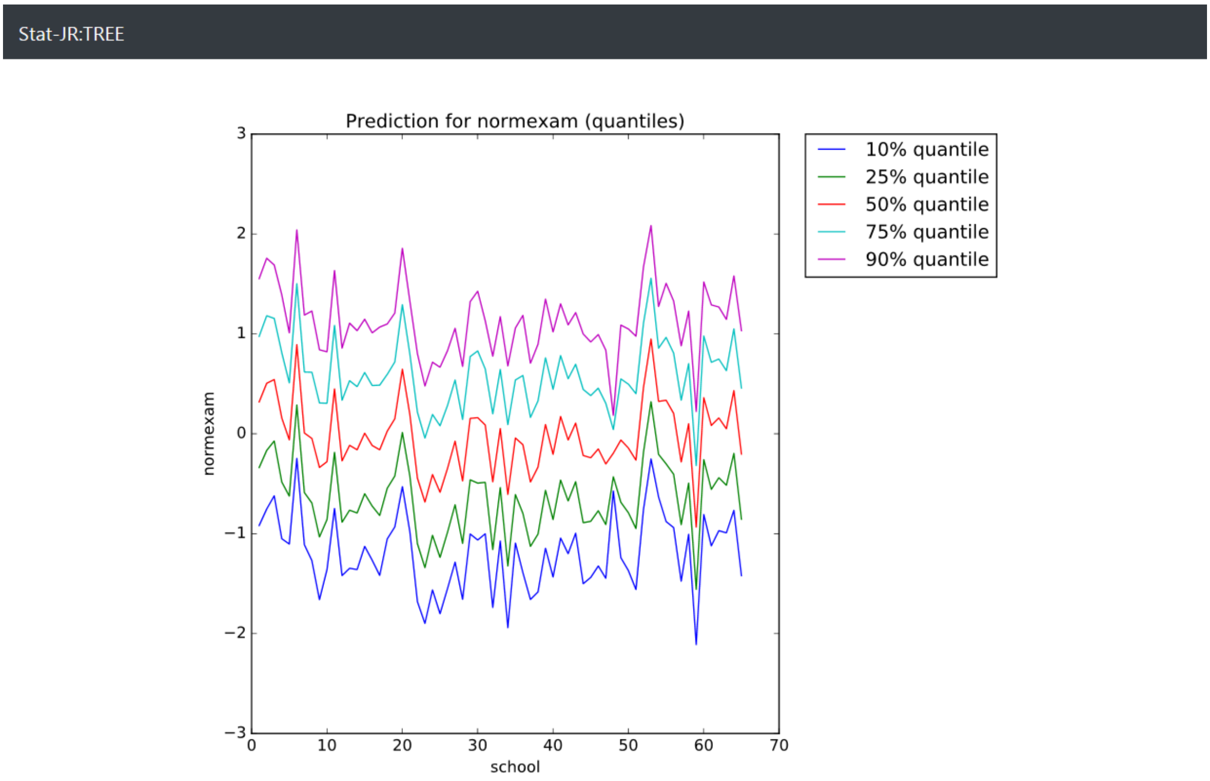
Here we see for school 35 that the estimated SD is 0 and this occurs as we only have 1 pupil in the sample for this school. Even worse we see no estimates for school 44 and this is because there are no pupils at all in the sample for this school. Here we see a clear advantage therefore of the model-based small area estimation approach in that we can borrow strength from the relationship between predictors and the response in the sample as a whole to construct small area estimates for schools that are not in the sample. So in this case although we have no responses for school 44 we do have predictor variables (*gender* and *standlrt*) for this school and these allow us to predict that the school is likely to have an average score that is below average. We can also look at schools where the sample and population estimates are very different for example school 34 where the sample data gives a large negative estimate of -1.346 whilst the model gives an estimate of -0.620 which is closer to 0. In fact only 3 of the 26 pupils in this school are in the sample and these 3 have scored less on average than the other 23 (and if we had access to the scores for all 26 pupils the average is -0.371). We'll look at this in more detail later.

The **SAEModel** template has a small group of outputs which are constructed from the parameter outputs in the **2LevelSAEMCMC** template. If we next look from the pulldown list at *sae_mean.svg* and click on the **Popout** button to put it in a new tab we will see the following:



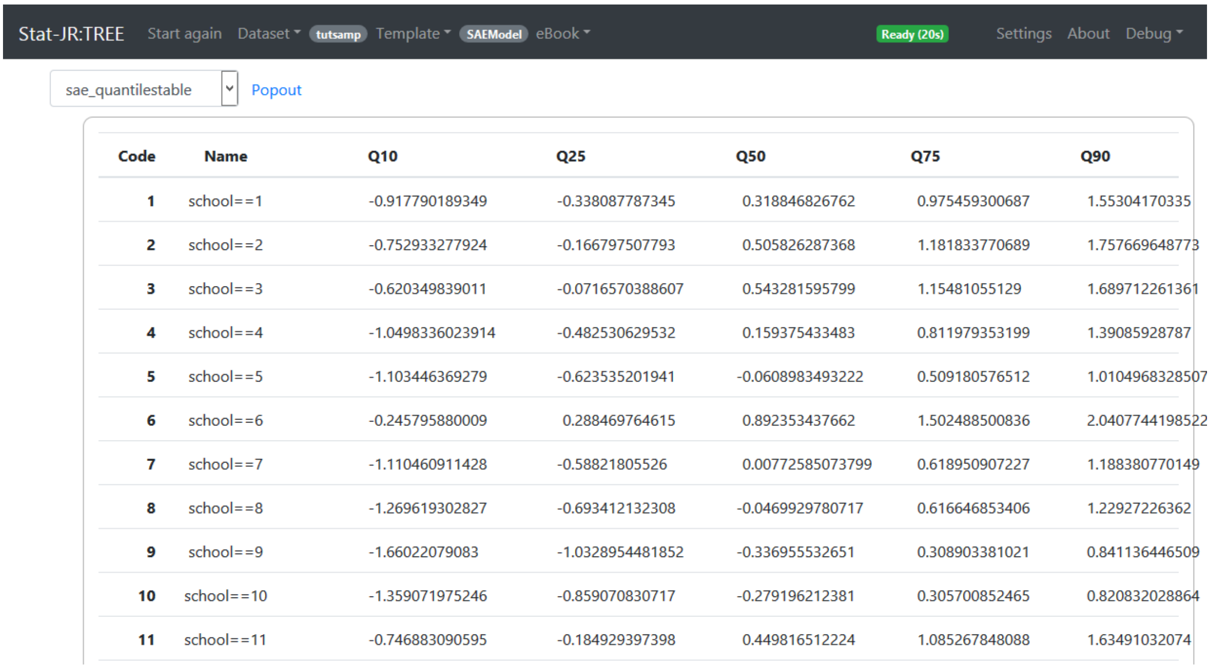
Here we see the (mean) estimates for all 65 schools along with confidence intervals for each estimate. For the first school we can see the mean estimate is 0.316 as in the earlier template with an SD that results in a confidence interval that overlaps with 0 as expected and as earlier the second school confidence interval doesn't quite overlap with 0.

We can also see visualisations of other small area estimations, for example the quantiles that we looked at earlier can be visualised in *sae_quantiles.svg* (and popped out) as shown below:



Here we see a whole selection of quantiles in different colours for the different schools. Note we previously looked at the 10th quantile which is to the bottom of the screen in dark blue.

Although the graph is pretty it is perhaps hard to focus on the results for a specific school so the plot has a complementary table with the specific numbers, *sae_quantilestable*



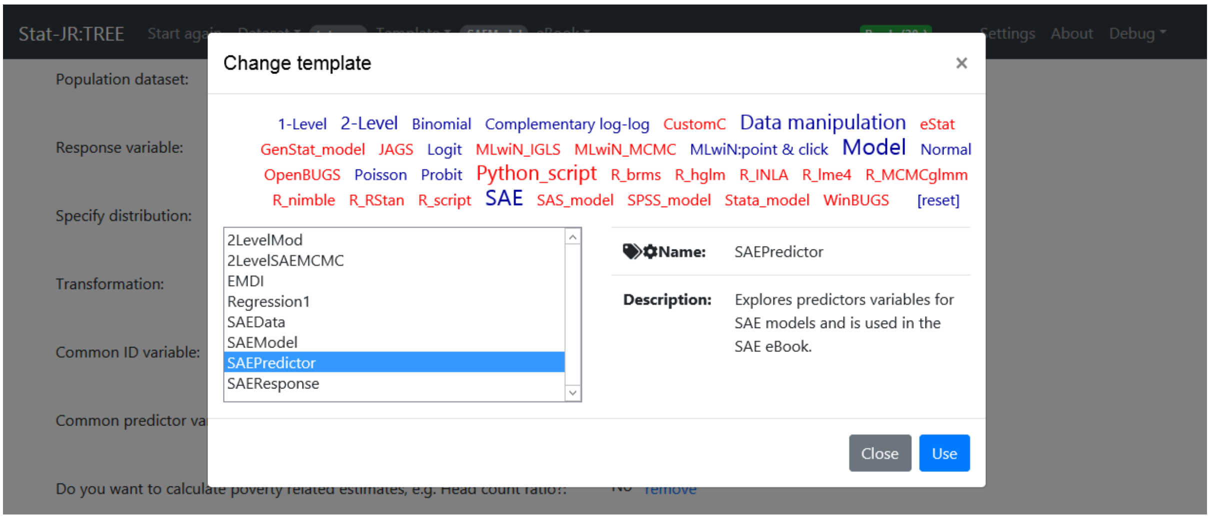
Code	Name	Q10	Q25	Q50	Q75	Q90
1	school==1	-0.917790189349	-0.338087787345	0.318846826762	0.975459300687	1.55304170335
2	school==2	-0.752933277924	-0.166797507793	0.505826287368	1.181833770689	1.757669648773
3	school==3	-0.620349839011	-0.0716570388607	0.543281595799	1.15481055129	1.689712261361
4	school==4	-1.0498336023914	-0.482530629532	0.159375433483	0.811979353199	1.39085928787
5	school==5	-1.103446369279	-0.623535201941	-0.0608983493222	0.509180576512	1.0104968328507
6	school==6	-0.245795880009	0.288469764615	0.892353437662	1.502488500836	2.0407744198522
7	school==7	-1.110460911428	-0.58821805526	0.00772585073799	0.618950907227	1.188380770149
8	school==8	-1.269619302827	-0.693412132308	-0.0469929780717	0.616646853406	1.22927226362
9	school==9	-1.66022079083	-1.0328954481852	-0.336955532651	0.308903381021	0.841136446509
10	school==10	-1.359071975246	-0.859070830717	-0.279196212381	0.305700852465	0.820832028864
11	school==11	-0.746883090595	-0.184929397398	0.449816512224	1.085267848088	1.63491032074

Here it is easier to focus in on specific schools which each have a row in the table so for example school 1 has an inter quartile range from -0.338 to 0.975. This template can produce several other visualisations and tables constructed from the small area estimates but we will discuss these when we move on to the eBook interface later.

The SAEPredictor template

Before moving on to the eBook interface we will first look at one of the other SAE templates, that is in fact used in the SAE eBook. If you recall when we looked at the *SAEModel* template we observed that for some schools the small area estimate from the model is rather different from that from the direct estimate from the data. One possible reason for this may be that the children from the school in the sample are not typical children from the school as a whole. We can investigate this by looking at the data we have both the sample and population i.e. the predictors. To do this we use the *SAEPredictor* template.

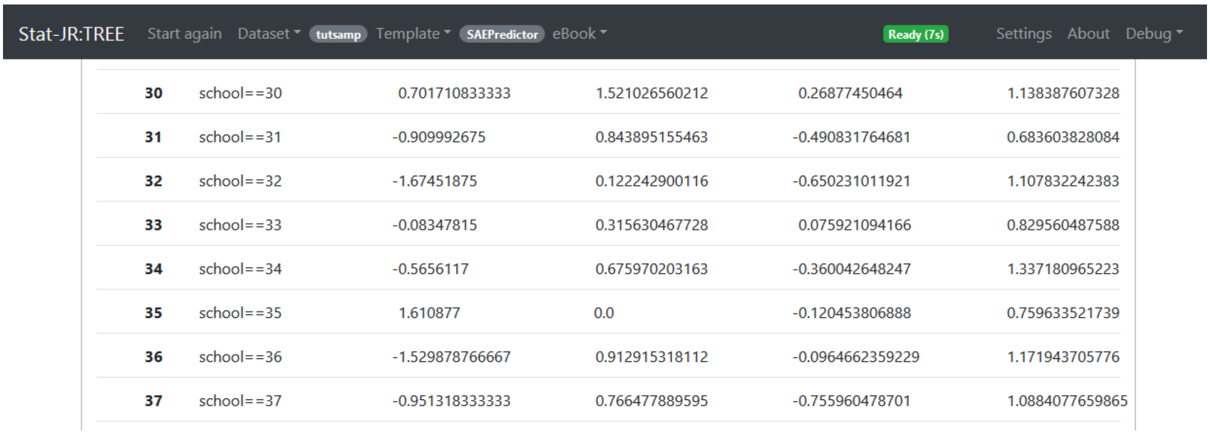
First therefore **Choose** this template from the *Template list* as shown:


Stat-JR:TREE
Start again
Dataset
Template
eBook
Settings
About
Debug
Population dataset:
Response variable:
Specify distribution:
Transformation:
Common ID variable:
Common predictor va
Do you want to calculate poverty related estimates, e.g. Head count ratio?:
remove
Change template
×
1-Level 2-Level Binomial Complementary log-log CustomC Data manipulation eStat GenStat_model JAGS Logit MLwiN_IGLS MLwiN_MCMC MLwiN:point & click Model Normal OpenBUGS Poisson Probit Python_script R_brms R_hglm R_INLA R_lme4 R_MCMCglmm R_nimble R_RStan R_script SAE SAS_model SPSS_model Stata_model WinBUGS [reset]
2LevelMod
2LevelSAEMCMC
EMDI
Regression1
SAEData
SAEModel
SAEPredictor
SAEResponse
Close
Use

Name: SAEPredictor

Description: Explores predictors variables for SAE models and is used in the SAE eBook.

Now clicking on **Use** we see that the template inputs are all filled in apart from the predictor to choose so select *standlrt* and press **Next** and **Run.** This template produces several objects related to the predictor we choose but for now we will focus on the *summary* object. Choosing this gives us an output list of the sample and populations statistics for the predictor. Previously we had identified school 34 as one where the estimates from the modelling approach and direct approach were rather different. If we scroll down to see this school we see the following:



30	school==30	0.701710833333	1.521026560212	0.26877450464	1.138387607328
31	school==31	-0.909992675	0.843895155463	-0.490831764681	0.683603828084
32	school==32	-1.67451875	0.122242900116	-0.650231011921	1.107832242383
33	school==33	-0.08347815	0.315630467728	0.075921094166	0.829560487588
34	school==34	-0.5656117	0.675970203163	-0.360042648247	1.337180965223
35	school==35	1.610877	0.0	-0.120453806888	0.759633521739
36	school==36	-1.529878766667	0.912915318112	-0.0964662359229	1.171943705776
37	school==37	-0.951318333333	0.766477889595	-0.755960478701	1.0884077659865

Here we see that the sample of pupils for school 34 had average *standlrt* score of -0.566 whilst the larger population had average score -0.360. The sample is therefore of lower average intake score than the population more generally. However this is actually not as dramatically different as many of the other schools. In fact it might also be that the sample simply have unusually low scores in the response compared to the population regardless of the predictor.

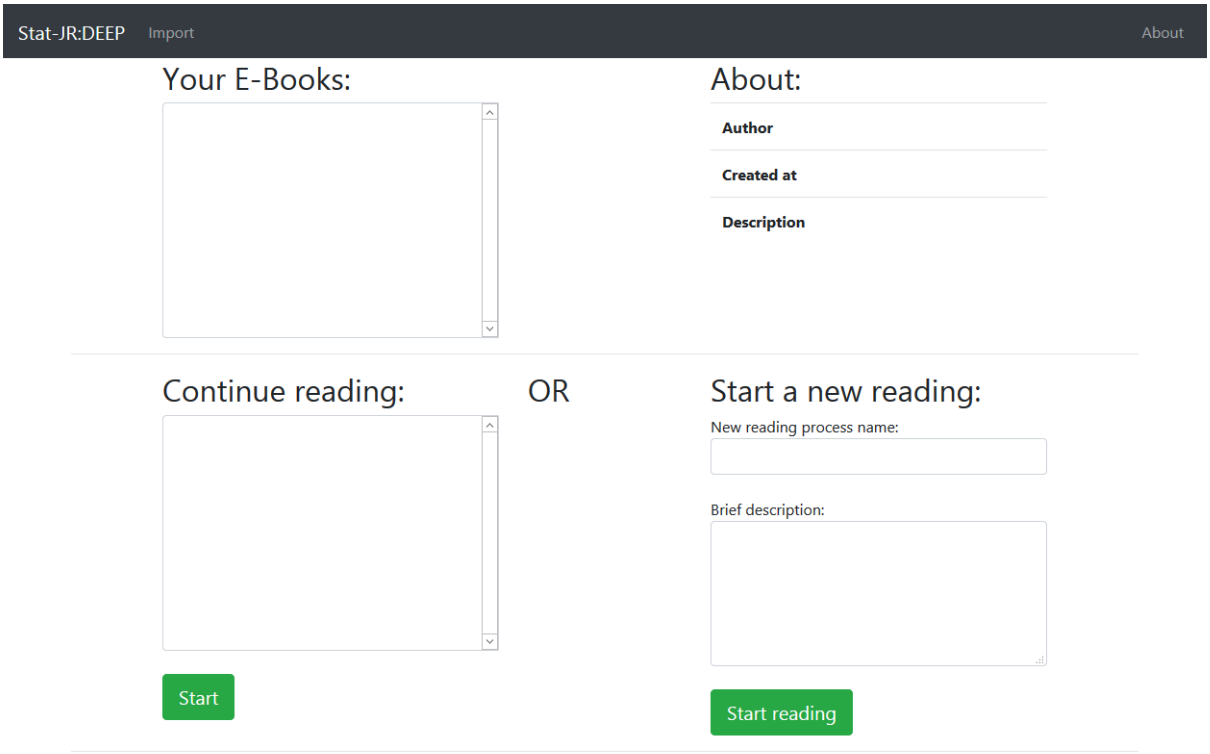
We will end our exploration of the TREE interface here and now move onto the DEEP eBook interface to Stat-JR next. In reality the eBook interface is meant to be self-contained and not need instructions but here we will simply link what it shows to what we have learnt in this earlier section by looking at templates directly in TREE.

The SAE eBook in DEEP

The DEEP eBook interface is a second interface into the Stat-JR package which shares much of the underlying functionality in terms of Stat-JR templates that perform specific functions. It however uses eBooks that the user can write which embed the templates and the output objects they produce into an electronic book which wraps them in explanatory text and positions them in a sensible way.

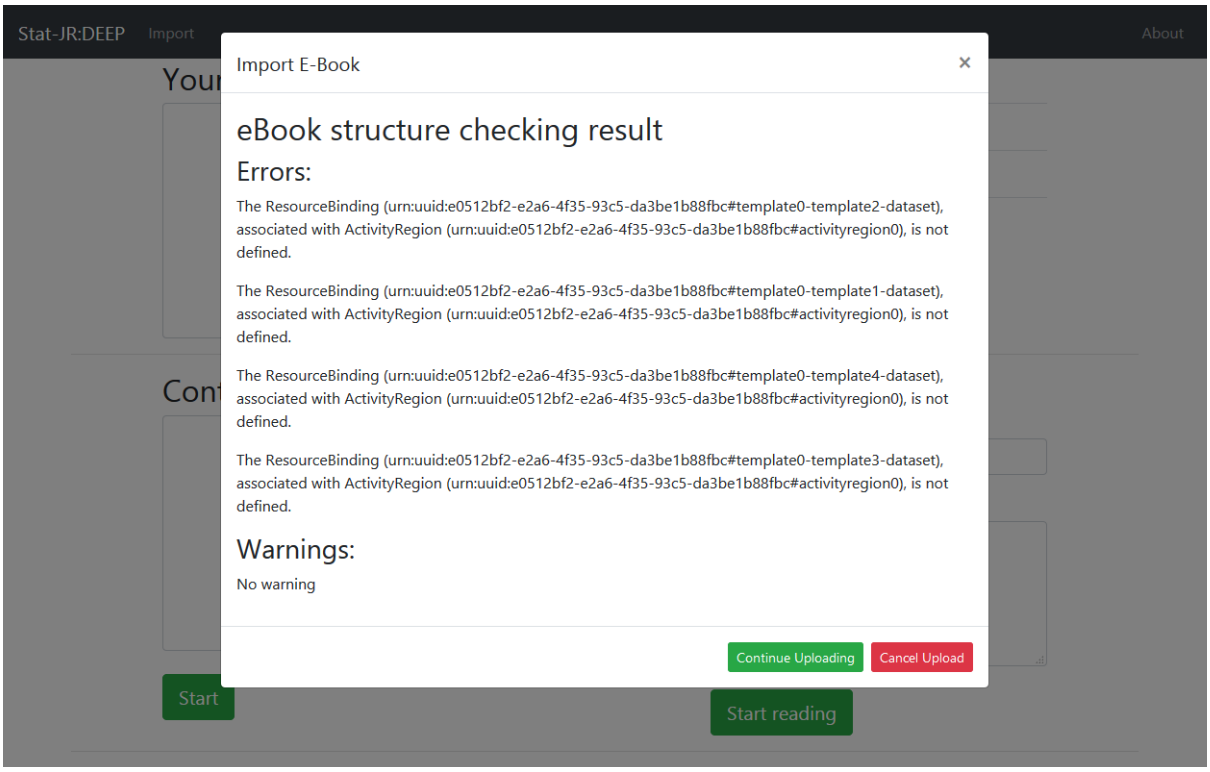
To run DEEP look in the home directory of the memory stick and double click on the shortcut for the file *DEEP.exe*

The DEEP starting screen looks as follows:



To begin with we need to Import the eBook so to do this we click on **Import** and then on the green **Select an E-Book file** button. In the window that appears find the file *ExploreSAE.zip* (from the ebooks subdirectory of the home directory of the memory stick) and click on **Open**.

The screen should then look as follows:



eBook structure checking result

Errors:

The ResourceBinding (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#template0-template2-dataset), associated with ActivityRegion (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#activityregion0), is not defined.

The ResourceBinding (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#template0-template1-dataset), associated with ActivityRegion (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#activityregion0), is not defined.

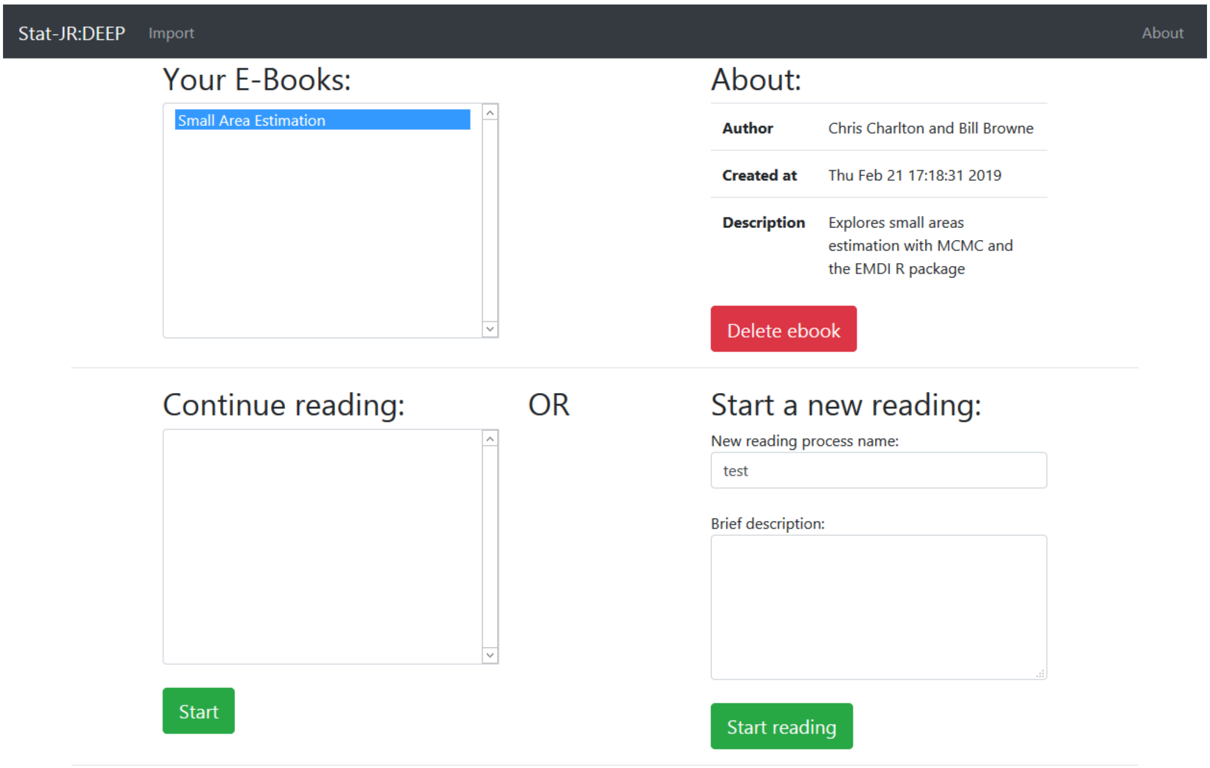
The ResourceBinding (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#template0-template4-dataset), associated with ActivityRegion (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#activityregion0), is not defined.

The ResourceBinding (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#template0-template3-dataset), associated with ActivityRegion (urn:uuid:e0512bf2-e2a6-4f35-93c5-da3be1b88fbc#activityregion0), is not defined.

Warnings:

No warning

These error messages are nothing to worry about as it is simply the software telling us we haven't yet loaded up datasets on each page of the eBook (which is done when the eBook is run) so click on **Continue Uploading** and **Continue**. The eBook will then be loaded and appear in the list of eBooks at the top left. If we click on it then information about it will appear in the top right and we can then type a (reading process) name (e.g. test) in the bottom right as shown:



Your E-Books:

Small Area Estimation

About:

Author	Chris Charlton and Bill Browne
Created at	Thu Feb 21 17:18:31 2019
Description	Explores small areas estimation with MCMC and the EMDI R package

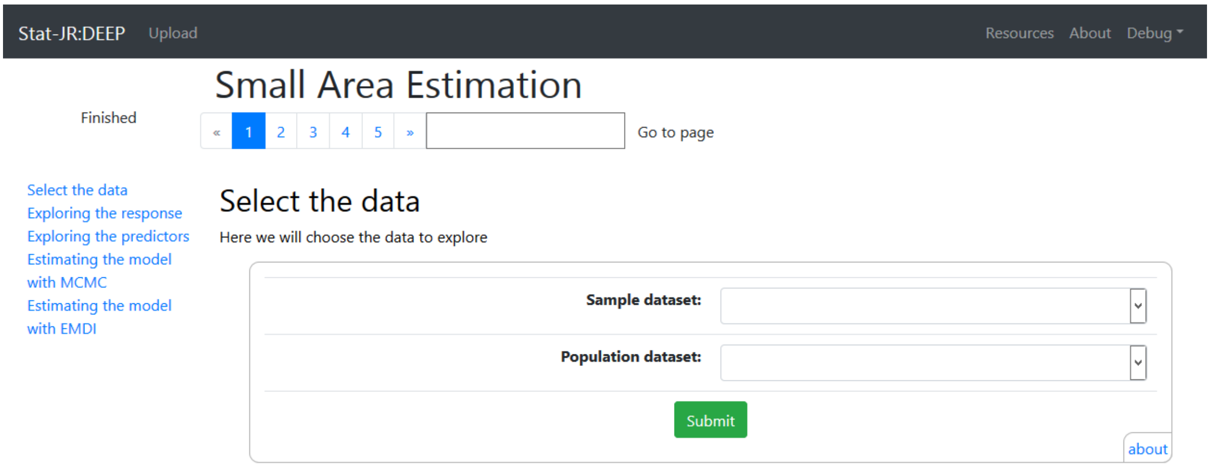
Continue reading:

Start a new reading:

New reading process name: test

Brief description:

Clicking on the green **Start reading** button next will allow us to start reading the eBook which will appear as follows:



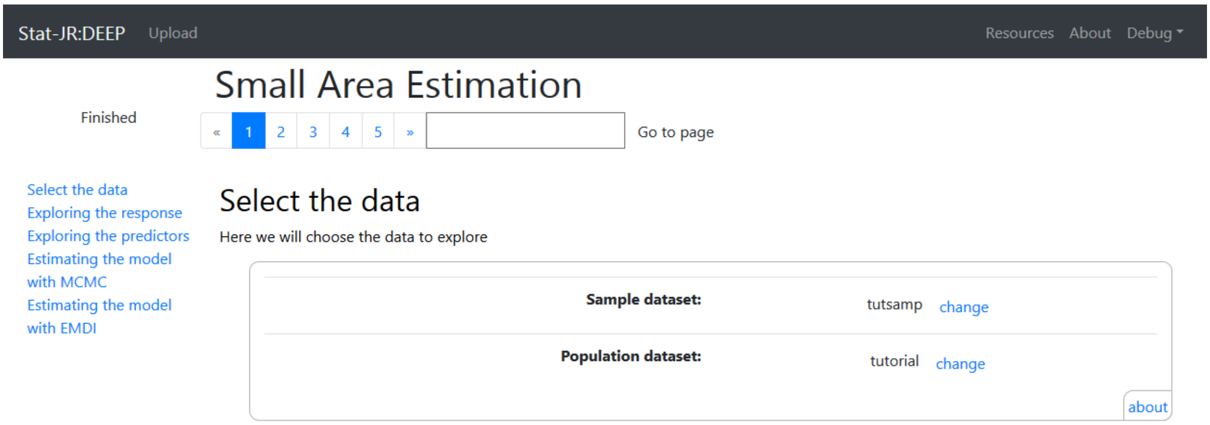
Select the data

Here we will choose the data to explore

Sample dataset:

Population dataset:

Here we see that the eBook has 5 pages. The first page which we see is simply used to identify the datasets that we will use in our SAE modelling. The other pages have a logical order in that pages 2 and 3 allow us to explore the dataset before we begin modelling. The modelling using MCMC is done on page 4 and then finally on page 5 we can do similar modelling using a different method using the EMDI package from R. We will begin by entering the datasets so as with our previous work in TREE our sample dataset is *tutsamp* whilst out population dataset is *tutorial*. If we input these options and press the green **Submit** button then the progress indicator (in the top left) will briefly indicate activity before returning to saying *Finished* and the screen will look as follows:



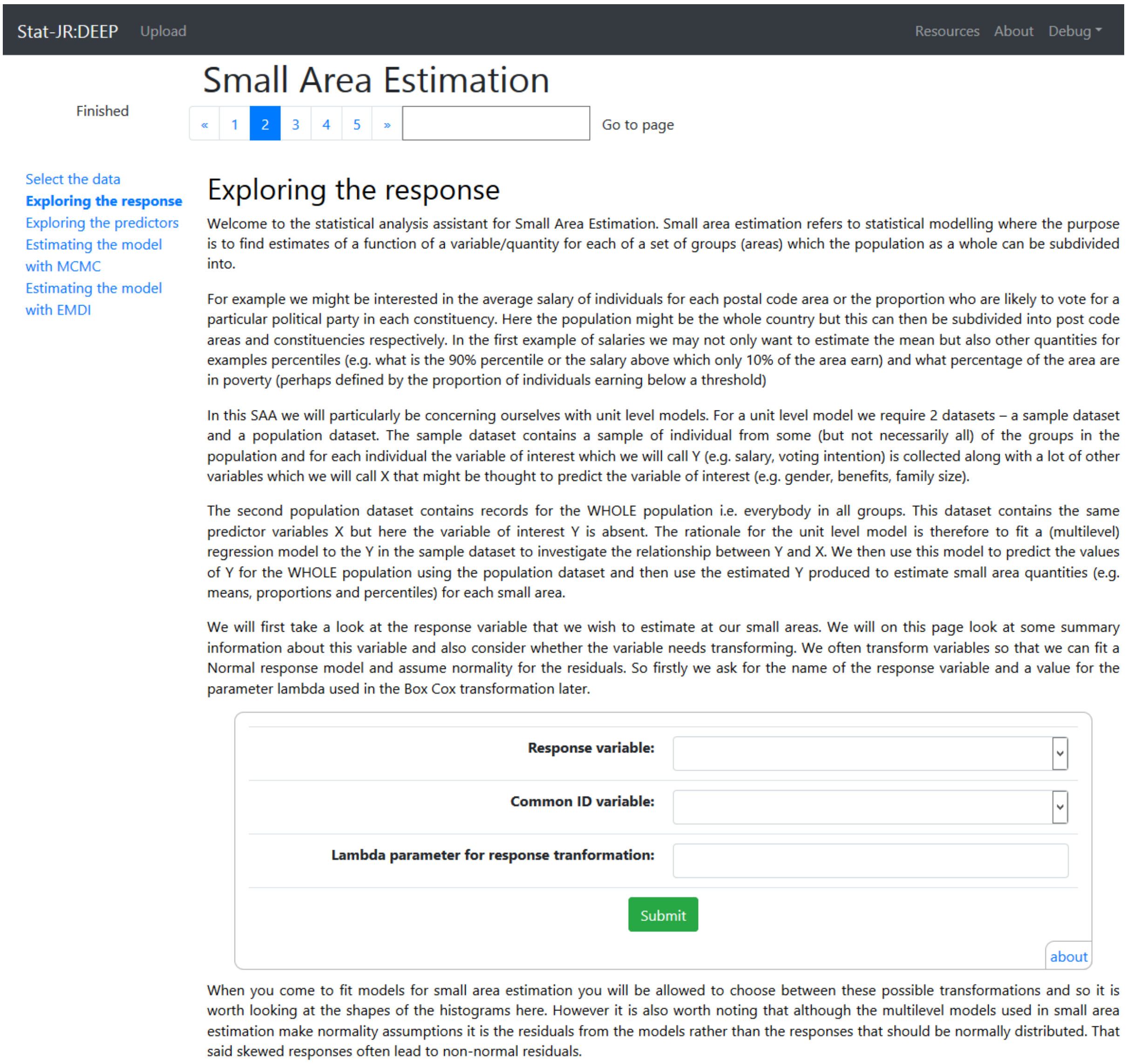
Select the data

Here we will choose the data to explore

Sample dataset: tutsamp change

Population dataset: tutorial change

We can next move onto page 2 and do some exploratory work on the response variable of interest. Clicking on the blue 2 will move us on to page 2 and we will see that this page first gives some background text as shown below.



Small Area Estimation

Exploring the response

Welcome to the statistical analysis assistant for Small Area Estimation. Small area estimation refers to statistical modelling where the purpose is to find estimates of a function of a variable/quantity for each of a set of groups (areas) which the population as a whole can be subdivided into.

For example we might be interested in the average salary of individuals for each postal code area or the proportion who are likely to vote for a particular political party in each constituency. Here the population might be the whole country but this can then be subdivided into post code areas and constituencies respectively. In the first example of salaries we may not only want to estimate the mean but also other quantities for examples percentiles (e.g. what is the 90% percentile or the salary above which only 10% of the area earn) and what percentage of the area are in poverty (perhaps defined by the proportion of individuals earning below a threshold)

In this SAA we will particularly be concerning ourselves with unit level models. For a unit level model we require 2 datasets – a sample dataset and a population dataset. The sample dataset contains a sample of individual from some (but not necessarily all) of the groups in the population and for each individual the variable of interest which we will call Y (e.g. salary, voting intention) is collected along with a lot of other variables which we will call X that might be thought to predict the variable of interest (e.g. gender, benefits, family size).

The second population dataset contains records for the WHOLE population i.e. everybody in all groups. This dataset contains the same predictor variables X but here the variable of interest Y is absent. The rationale for the unit level model is therefore to fit a (multilevel) regression model to the Y in the sample dataset to investigate the relationship between Y and X. We then use this model to predict the values of Y for the WHOLE population using the population dataset and then use the estimated Y produced to estimate small area quantities (e.g. means, proportions and percentiles) for each small area.

We will first take a look at the response variable that we wish to estimate at our small areas. We will on this page look at some summary information about this variable and also consider whether the variable needs transforming. We often transform variables so that we can fit a Normal response model and assume normality for the residuals. So firstly we ask for the name of the response variable and a value for the parameter lambda used in the Box Cox transformation later.

Response variable:

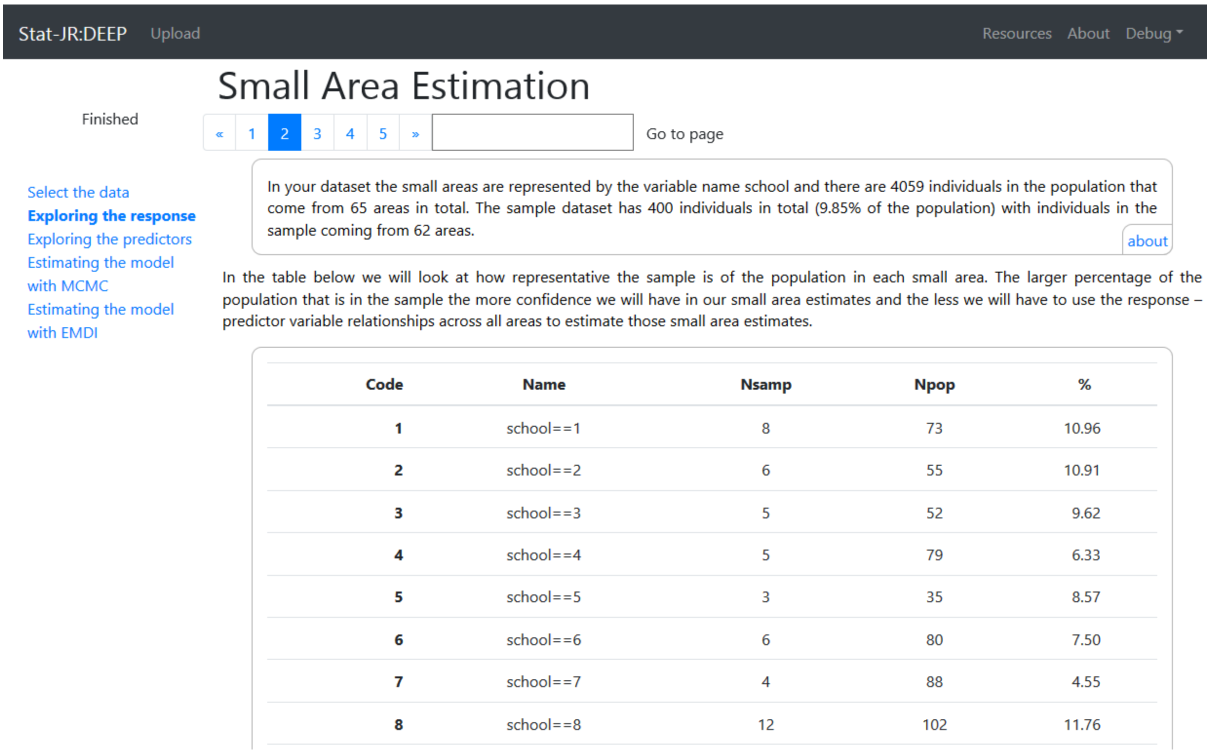
Common ID variable:

Lambda parameter for response tranformation:

When you come to fit models for small area estimation you will be allowed to choose between these possible transformations and so it is worth looking at the shapes of the histograms here. However it is also worth noting that although the multilevel models used in small area estimation make normality assumptions it is the residuals from the models rather than the responses that should be normally distributed. That said skewed responses often lead to non-normal residuals.

We then have some boxes to fill in and we will choose *normexam* as the **Response variable**, *school* as the **Common ID variable** and for now 0.5 for the **Lambda parameter** value which is used to show a particular transformation later. Clicking on **Submit** and the rest of the page is updated to take account for the inputs.

The page first looks at what proportion of the population is contained in the sample:



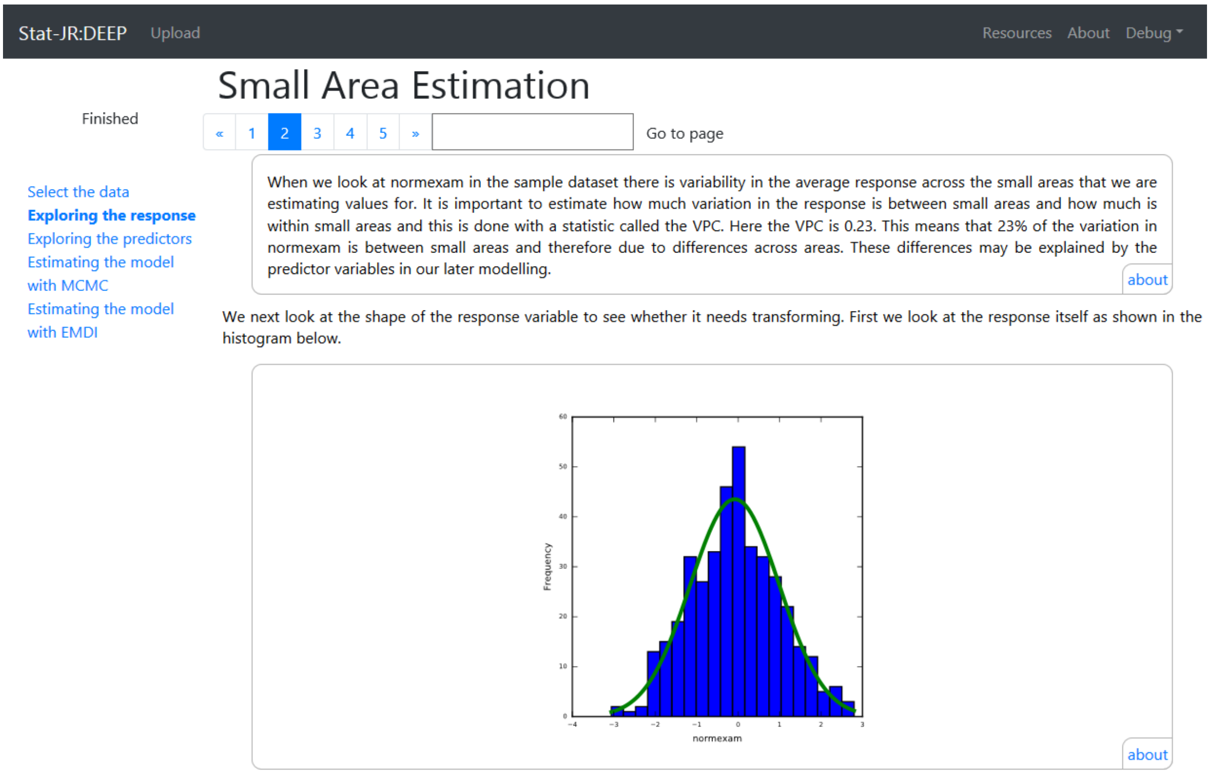
Small Area Estimation

In your dataset the small areas are represented by the variable name school and there are 4059 individuals in the population that come from 65 areas in total. The sample dataset has 400 individuals in total (9.85% of the population) with individuals in the sample coming from 62 areas.

In the table below we will look at how representative the sample is of the population in each small area. The larger percentage of the population that is in the sample the more confidence we will have in our small area estimates and the less we will have to use the response – predictor variable relationships across all areas to estimate those small area estimates.

Code	Name	Nsamp	Npop	%
1	school==1	8	73	10.96
2	school==2	6	55	10.91
3	school==3	5	52	9.62
4	school==4	5	79	6.33
5	school==5	3	35	8.57
6	school==6	6	80	7.50
7	school==7	4	88	4.55
8	school==8	12	102	11.76

Here we see that the sample contains nearly 10% of the population. We also see that in fact 3 of the schools in the population have no data in the sample. This is expanded upon in the table which shows the relative sizes of the sample and population for each school. Scrolling down we next look at how important the clustering is in the response and as we see below it reports the variance partitioning coefficient (VPC):



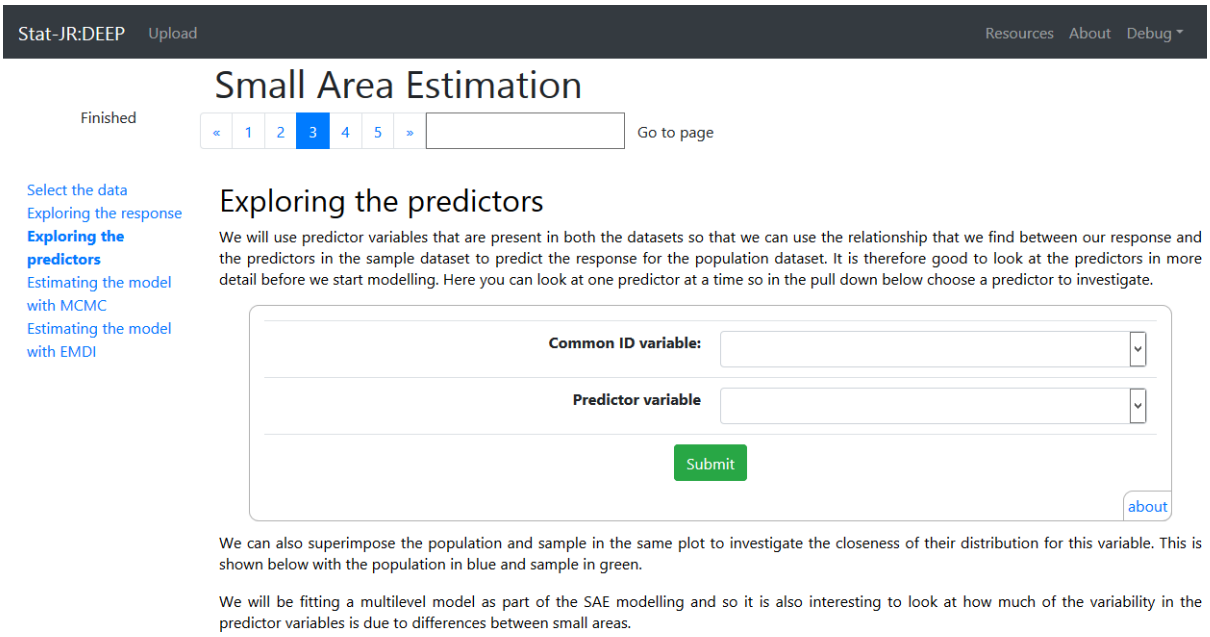
Small Area Estimation

When we look at normexam in the sample dataset there is variability in the average response across the small areas that we are estimating values for. It is important to estimate how much variation in the response is between small areas and how much is within small areas and this is done with a statistic called the VPC. Here the VPC is 0.23. This means that 23% of the variation in normexam is between small areas and therefore due to differences across areas. These differences may be explained by the predictor variables in our later modelling.

We next look at the shape of the response variable to see whether it needs transforming. First we look at the response itself as shown in the histogram below.

Here we see that 23% of the variability in exam score is due to school differences in the sample suggesting that multilevel modelling is important in the modelling we do later. We next look at the shape of the response which as the data has been normalised looks unsurprisingly a good fit to a normal distribution. This is confirmed by a skewness statistic and if you look further down you will see similar plots for three possible transformations of this variable – a log transform and Box-Cox and Dual Power transforms with a specific value of the lambda parameter in each case (0.5 which we input). We will omit showing these here for brevity in particular as the data on the original scale looks reasonably normal.

We will next move onto page 3 of the eBook which allows us to explore the potential predictor variables in more detail. So click on the blue 3 and you will be taken to the following screen:



Exploring the predictors

We will use predictor variables that are present in both the datasets so that we can use the relationship that we find between our response and the predictors in the sample dataset to predict the response for the population dataset. It is therefore good to look at the predictors in more detail before we start modelling. Here you can look at one predictor at a time so in the pull down below choose a predictor to investigate.

Common ID variable:

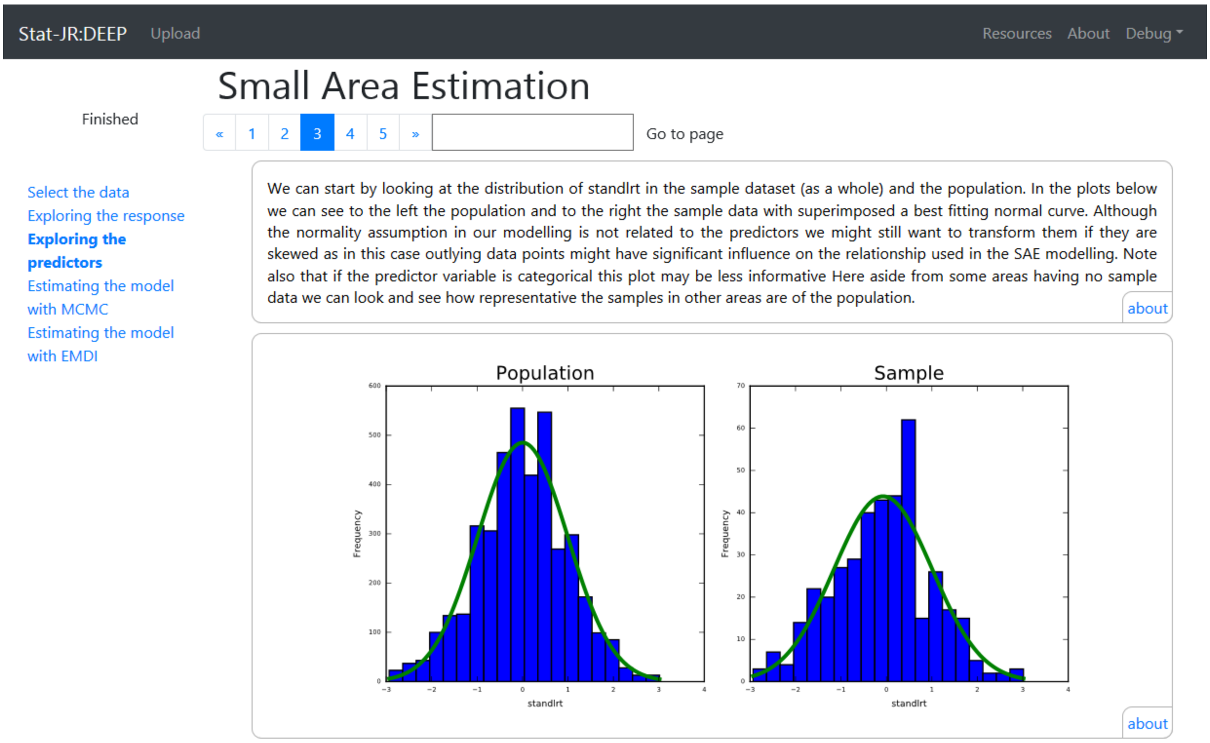
Predictor variable

We can also superimpose the population and sample in the same plot to investigate the closeness of their distribution for this variable. This is shown below with the population in blue and sample in green.

We will be fitting a multilevel model as part of the SAE modelling and so it is also interesting to look at how much of the variability in the predictor variables is due to differences between small areas.

Here we get some instructions and we need to again input the **Common ID variable** (*school*) as well as this time a chosen **Predictor variable** for which we will start with *standlrt*. This page actually sits on top of the *SAEPredictor* template we discussed earlier and thus the outputs on this page come from that template. If we click on the green **Submit** button the page will now be populated with outputs that refer to these inputs.

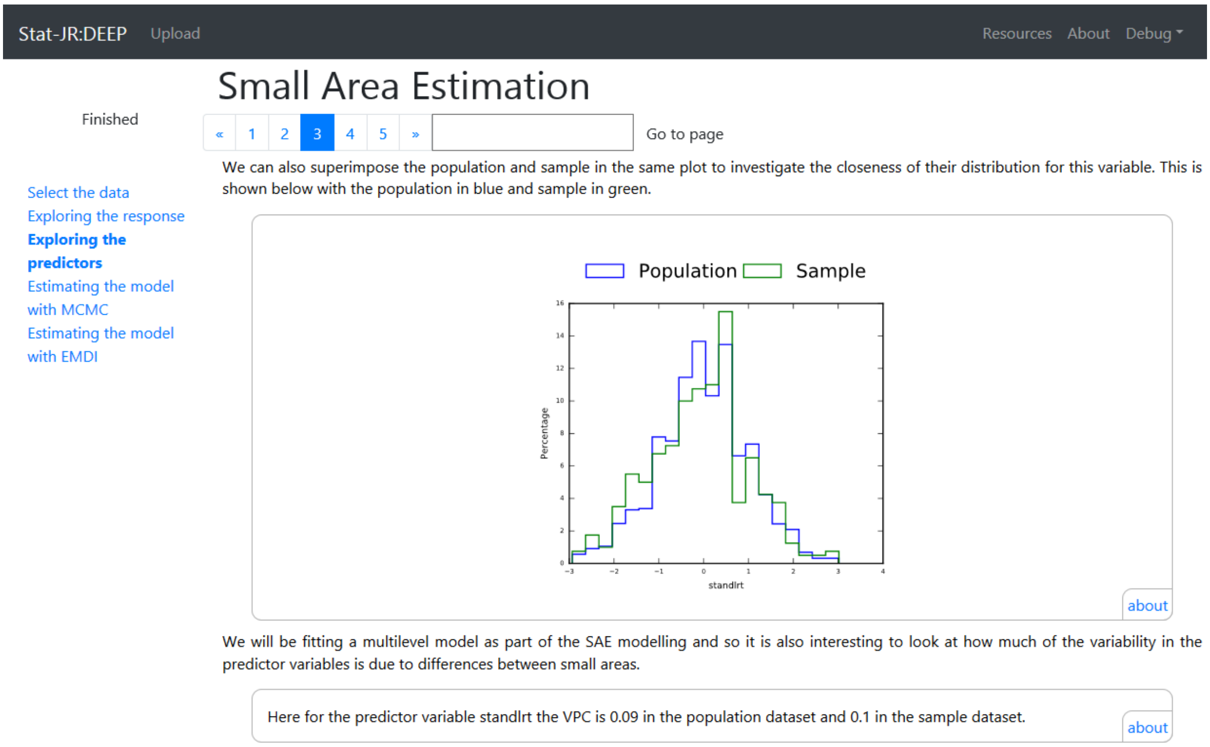
For predictor variables we have data for both the sample and population so we can look at how representative the sample is in several ways. To start with we look at the whole datasets as shown below:



Small Area Estimation

We can start by looking at the distribution of standlrt in the sample dataset (as a whole) and the population. In the plots below we can see to the left the population and to the right the sample data with superimposed a best fitting normal curve. Although the normality assumption in our modelling is not related to the predictors we might still want to transform them if they are skewed as in this case outlying data points might have significant influence on the relationship used in the SAE modelling. Note also that if the predictor variable is categorical this plot may be less informative Here aside from some areas having no sample data we can look and see how representative the samples in other areas are of the population.

Here we see two histograms, one for the sample and one for the population which don’t look wildly different. To aid in comparison we next superimpose one on the other:



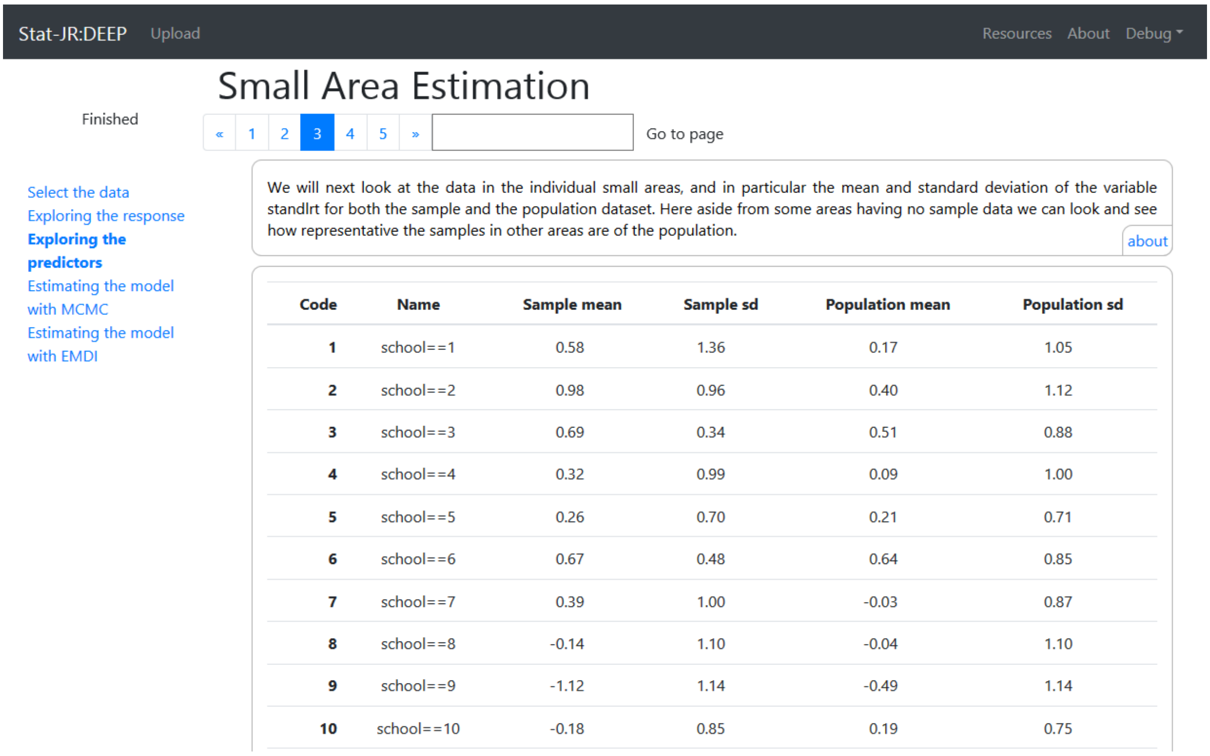
Small Area Estimation

We can also superimpose the population and sample in the same plot to investigate the closeness of their distribution for this variable. This is shown below with the population in blue and sample in green.

We will be fitting a multilevel model as part of the SAE modelling and so it is also interesting to look at how much of the variability in the predictor variables is due to differences between small areas.

Here for the predictor variable standlrt the VPC is 0.09 in the population dataset and 0.1 in the sample dataset.

Here we can see strong overlap between the sample and population we also see in the box below that in both the sample and population there is clustering of the predictor within schools with schools explaining 9% of the variability in each case. We next look at summary statistics for each area:

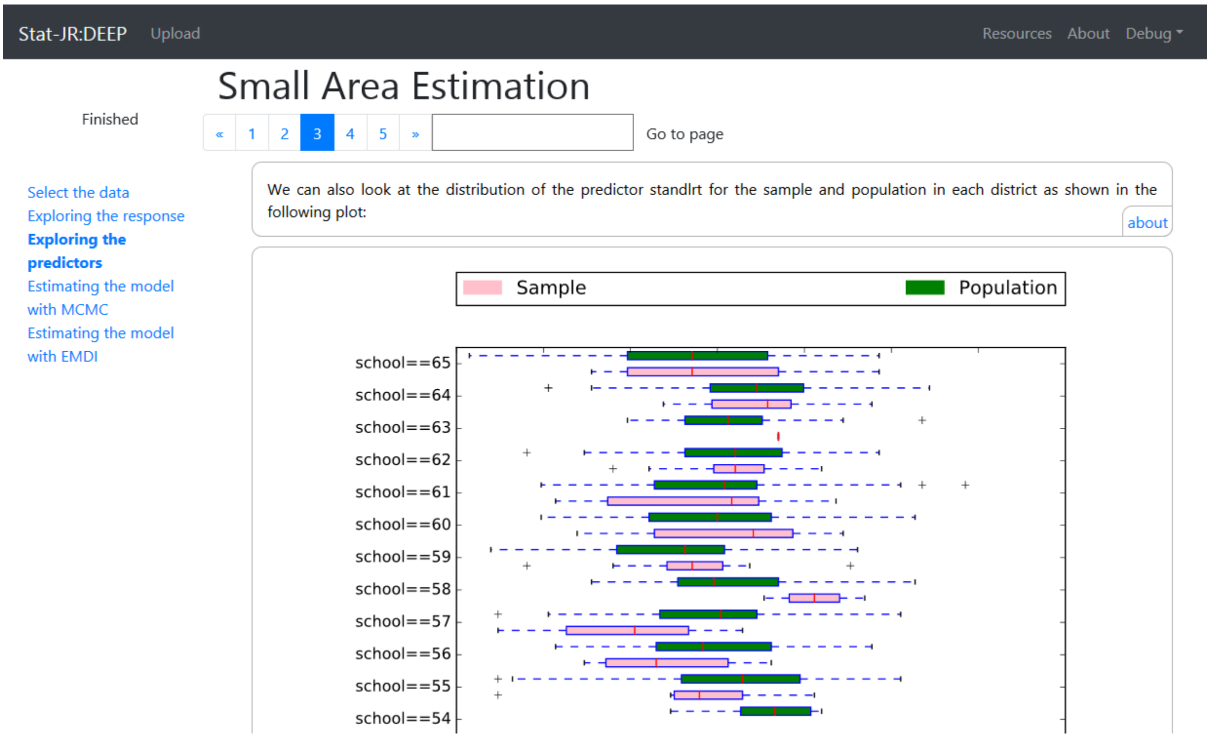


Small Area Estimation

We will next look at the data in the individual small areas, and in particular the mean and standard deviation of the variable standlrt for both the sample and the population dataset. Here aside from some areas having no sample data we can look and see how representative the samples in other areas are of the population.

Code	Name	Sample mean	Sample sd	Population mean	Population sd
1	school==1	0.58	1.36	0.17	1.05
2	school==2	0.98	0.96	0.40	1.12
3	school==3	0.69	0.34	0.51	0.88
4	school==4	0.32	0.99	0.09	1.00
5	school==5	0.26	0.70	0.21	0.71
6	school==6	0.67	0.48	0.64	0.85
7	school==7	0.39	1.00	-0.03	0.87
8	school==8	-0.14	1.10	-0.04	1.10
9	school==9	-1.12	1.14	-0.49	1.14
10	school==10	-0.18	0.85	0.19	0.75

Here we see the differences in mean and sd between the sample and population as we saw earlier only now the data has been tidied up and rounded to 2 decimal places. Finally we also show the distribution of the data for each school both in the sample and population in a stacked boxplot shown below:



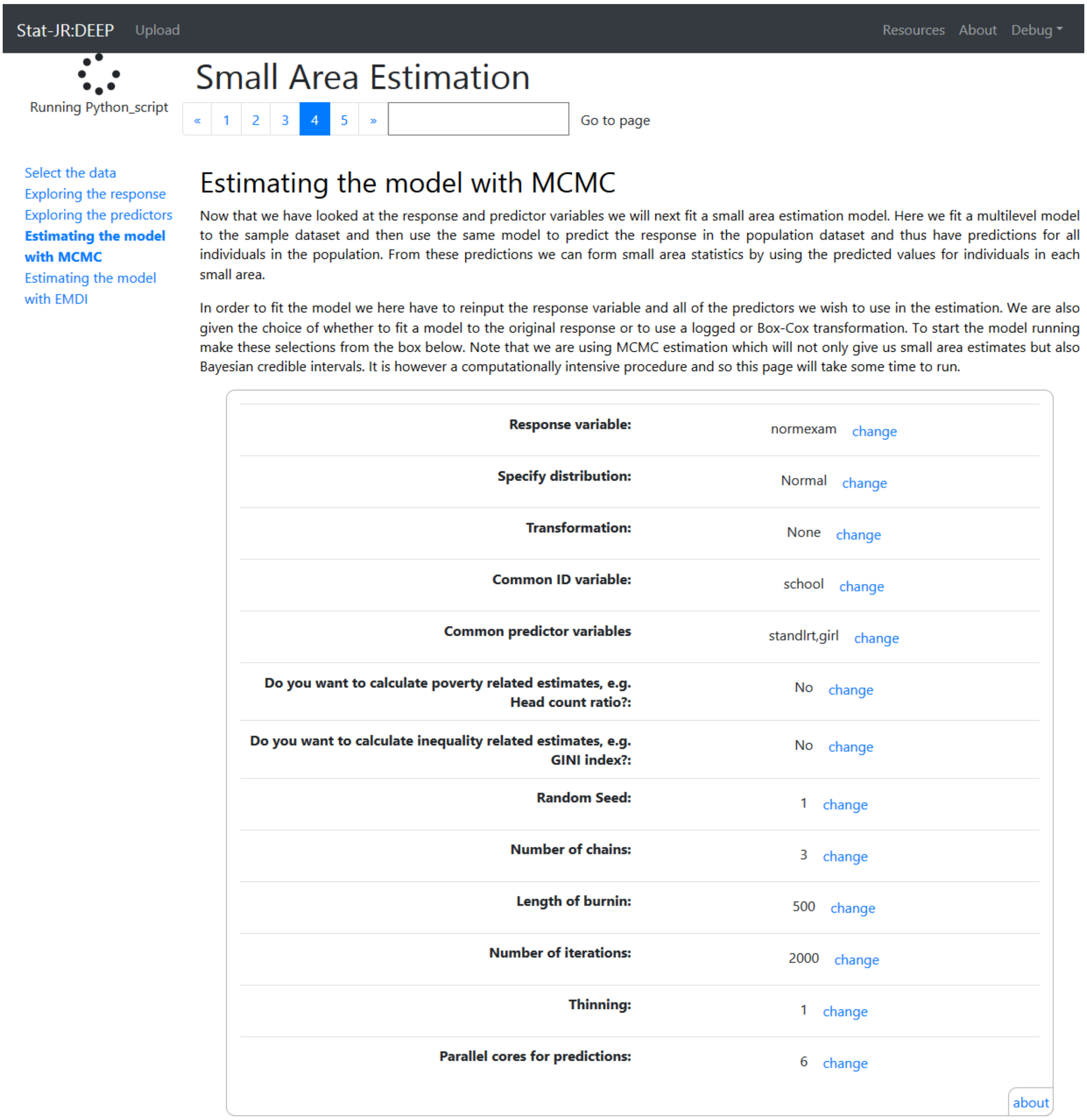
Small Area Estimation

We can also look at the distribution of the predictor standlrt for the sample and population in each district as shown in the following plot:

Here we see the degree of overlap for each school with sample data in pink and population data in green. We can also see for example that school 63 only has 1 data point in the sample while school 54 doesn’t have any.

MCMC Modelling in the eBook

If we now move onto page 4 by clicking on the blue 4 at the top of the screen we finally begin the SAE modelling properly. This page in effect sits on top of the *SAEModel* template we looked at earlier and all outputs will come from that template. The page initially requires a series of inputs and these should be input as follows:



Small Area Estimation

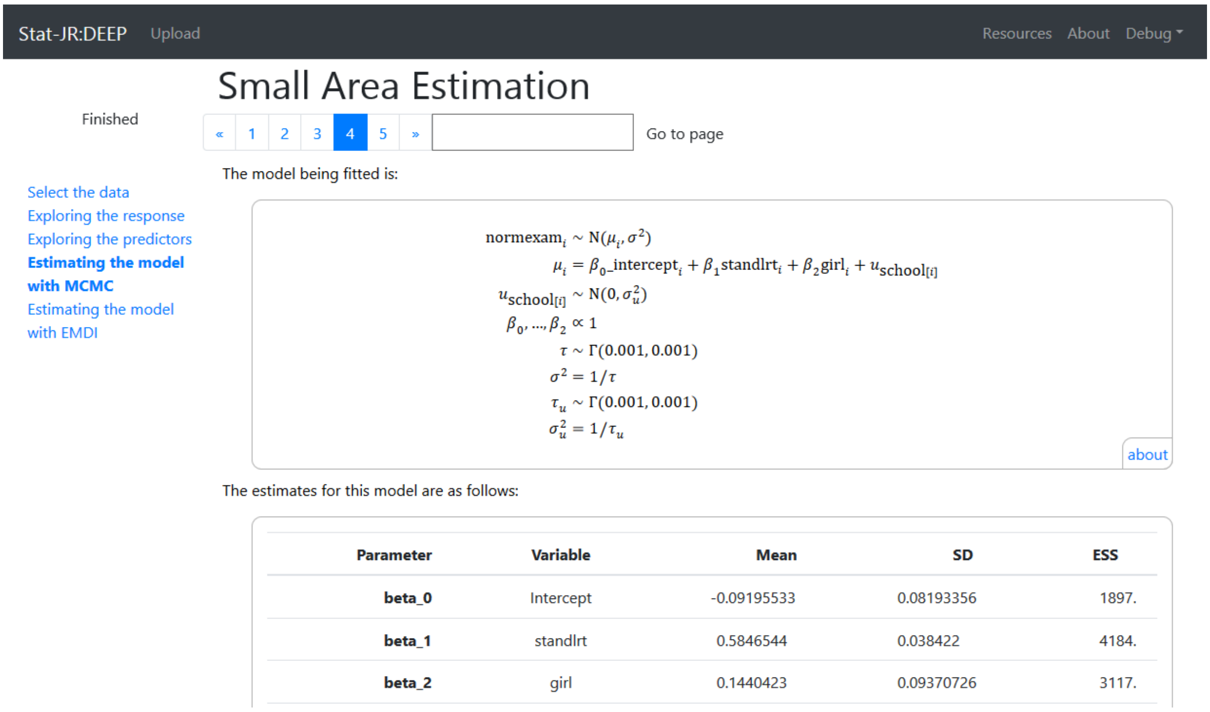
Estimating the model with MCMC

Now that we have looked at the response and predictor variables we will next fit a small area estimation model. Here we fit a multilevel model to the sample dataset and then use the same model to predict the response in the population dataset and thus have predictions for all individuals in the population. From these predictions we can form small area statistics by using the predicted values for individuals in each small area.

In order to fit the model we here have to reinput the response variable and all of the predictors we wish to use in the estimation. We are also given the choice of whether to fit a model to the original response or to use a logged or Box-Cox transformation. To start the model running make these selections from the box below. Note that we are using MCMC estimation which will not only give us small area estimates but also Bayesian credible intervals. It is however a computationally intensive procedure and so this page will take some time to run.

Response variable:	normexam	change
Specify distribution:	Normal	change
Transformation:	None	change
Common ID variable:	school	change
Common predictor variables	standlrt,girl	change
Do you want to calculate poverty related estimates, e.g. Head count ratio?:	No	change
Do you want to calculate inequality related estimates, e.g. GINI index?:	No	change
Random Seed:	1	change
Number of chains:	3	change
Length of burnin:	500	change
Number of iterations:	2000	change
Thinning:	1	change
Parallel cores for predictions:	6	change

Here as you can see to the top left on clicking submit for the last input the eBook begins running the code (*Python_script*) to do the SAE modelling. This will take a while compared to the computation on earlier pages and you will need to be patient at this point. When estimation is complete the indicator in the top left will change to indicate this though because the *SAEModel* template does several operations including the model fitting you will observe the page will update in stages while the script is still executing so for example the equations and estimates of the model being fitted to the small dataset are shown first:



Small Area Estimation

The model being fitted is:

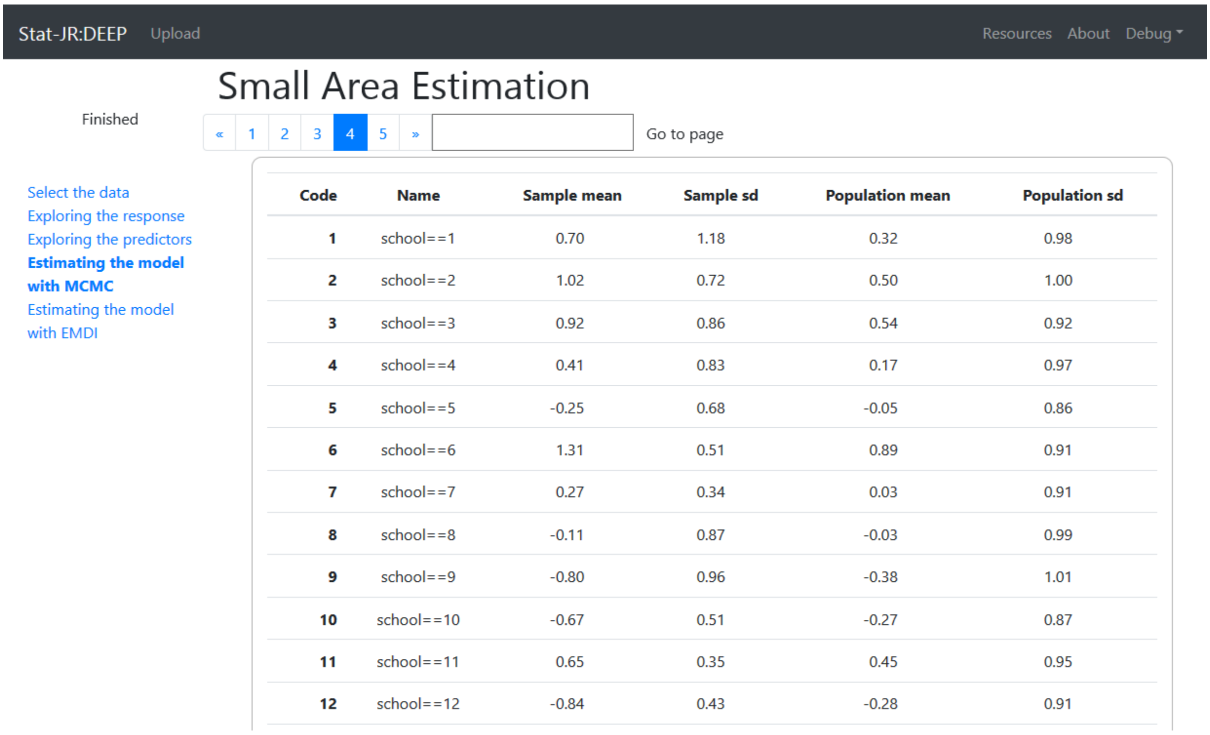
$$
\begin{aligned}
\text{normexam}_i &\sim \mathrm{N}(\mu_i, \sigma^2)\\
\mu_i &= \beta_0\text{\_intercept}_i + \beta_1\text{standlrt}_i + \beta_2\text{girl}_i + u_{\text{school}[i]}\\
u_{\text{school}[i]} &\sim \mathrm{N}(0, \sigma_u^2)\\
\beta_0, \ldots, \beta_2 &\propto 1\\
\tau &\sim \Gamma(0.001, 0.001)\\
\sigma^2 &= 1/\tau\\
\tau_u &\sim \Gamma(0.001, 0.001)\\
\sigma_u^2 &= 1/\tau_u
\end{aligned}
$$

The estimates for this model are as follows:

Parameter	Variable	Mean	SD	ESS
beta_0	Intercept	-0.09195533	0.08193356	1897.
beta_1	standlrt	0.5846544	0.038422	4184.
beta_2	girl	0.1440423	0.09370726	3117.

Here we see a positive effect for *standlrt* (0.585, with a very small SD (0.038)) indicating that the earlier London Reading test is a very important predictor of exam score. We also see a positive effect for *girl* which indicates that girls do on average 0.144 points better than boys. This model will then be used to predict exam scores for all 4,059 pupils for each iteration. When the model finishes running we also observe some residual plots to assess model fit but we ignore these for now and scroll down to look at the small area estimates.

We can first see the mean and sd estimates, both directly from the samples and based on the MCMC modelling as shown below:

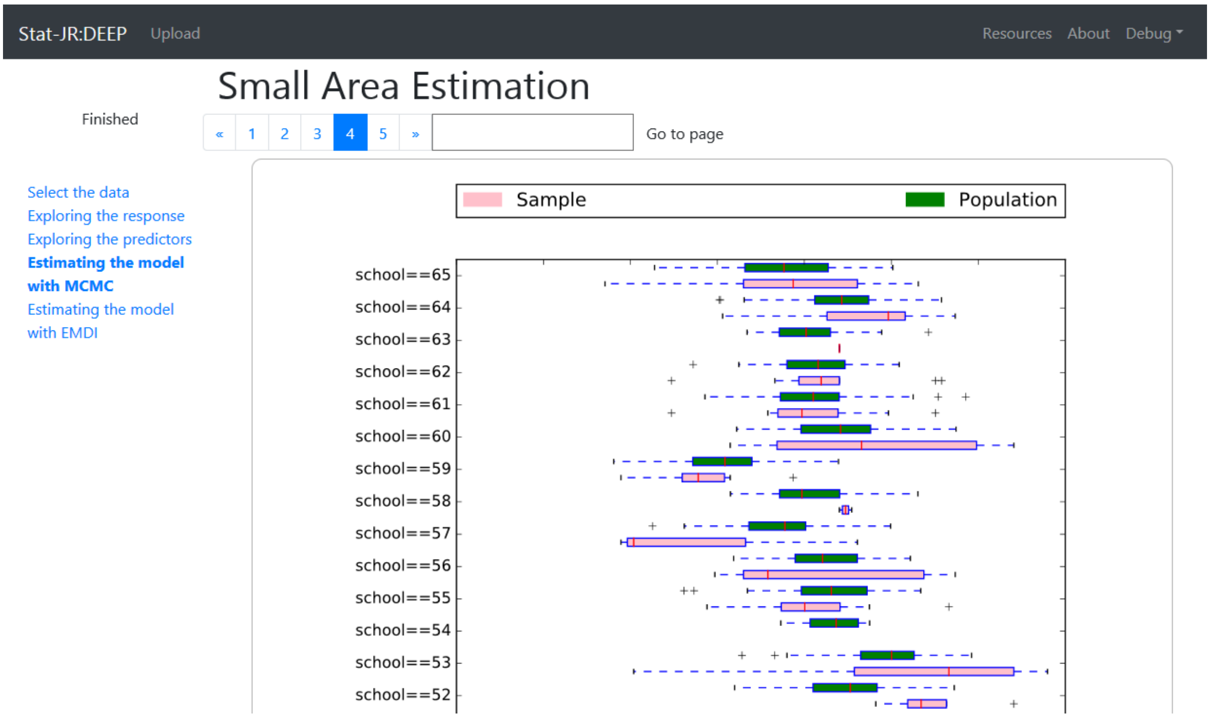


Small Area Estimation

Code	Name	Sample mean	Sample sd	Population mean	Population sd
1	school==1	0.70	1.18	0.32	0.98
2	school==2	1.02	0.72	0.50	1.00
3	school==3	0.92	0.86	0.54	0.92
4	school==4	0.41	0.83	0.17	0.97
5	school==5	-0.25	0.68	-0.05	0.86
6	school==6	1.31	0.51	0.89	0.91
7	school==7	0.27	0.34	0.03	0.91
8	school==8	-0.11	0.87	-0.03	0.99
9	school==9	-0.80	0.96	-0.38	1.01
10	school==10	-0.67	0.51	-0.27	0.87
11	school==11	0.65	0.35	0.45	0.95
12	school==12	-0.84	0.43	-0.28	0.91

These estimates we have already seen earlier when we looked at the *SAEModel* template and the only difference here is that the table has been tidied up by reducing the number of decimal places.

We next look at boxplots produced by each method which allows us to compare estimates from the modelling to the samples we had originally.

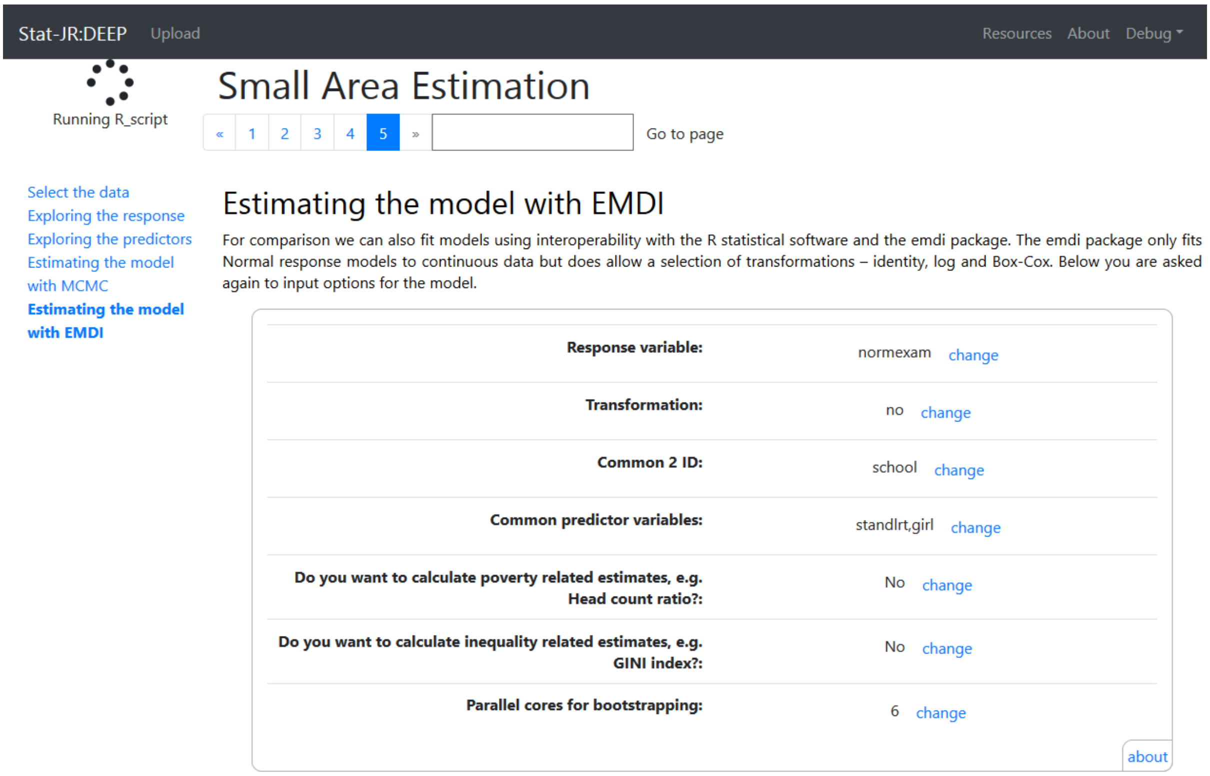


The page continues with several more plots and tables including the error bar plots for the means and then the plot and table for the quantiles all of which we saw earlier when we looked at the *SAEModel* template.

You might like to think about other possible predictors e.g. school gender, VR band that are available and what impact they have on the modelling we have done here though you will need to specify that they are categorical when you add them as predictors.

Modelling using EMDI

On page 5 we can look at an alternative modelling approach using a software package called EMDI which is run within the R package. Stat-JR has interoperability functionality which allows it to directly call the EMDI package in R and bring back the outputs. We will do this next by clicking on the 5 to turn to page 5 and to specify the inputs as follows:



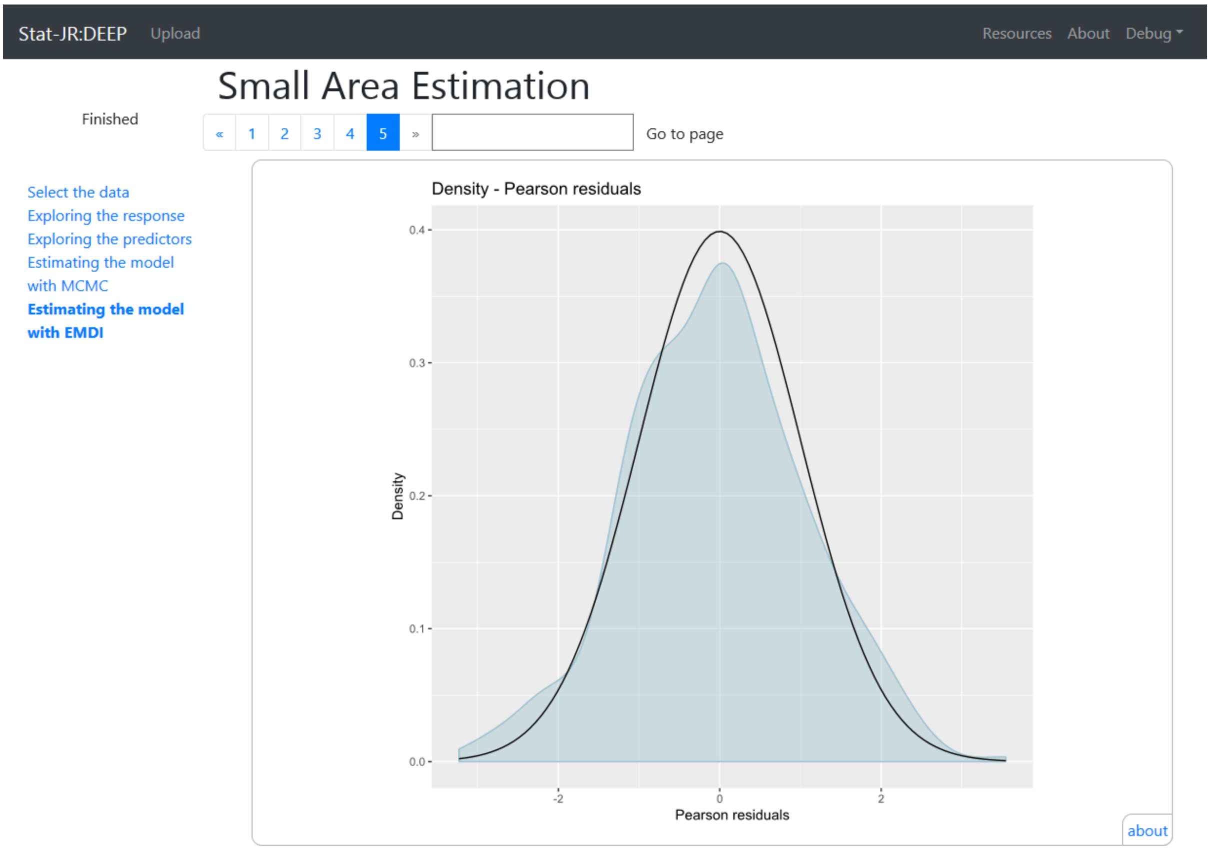
Small Area Estimation

Estimating the model with EMDI

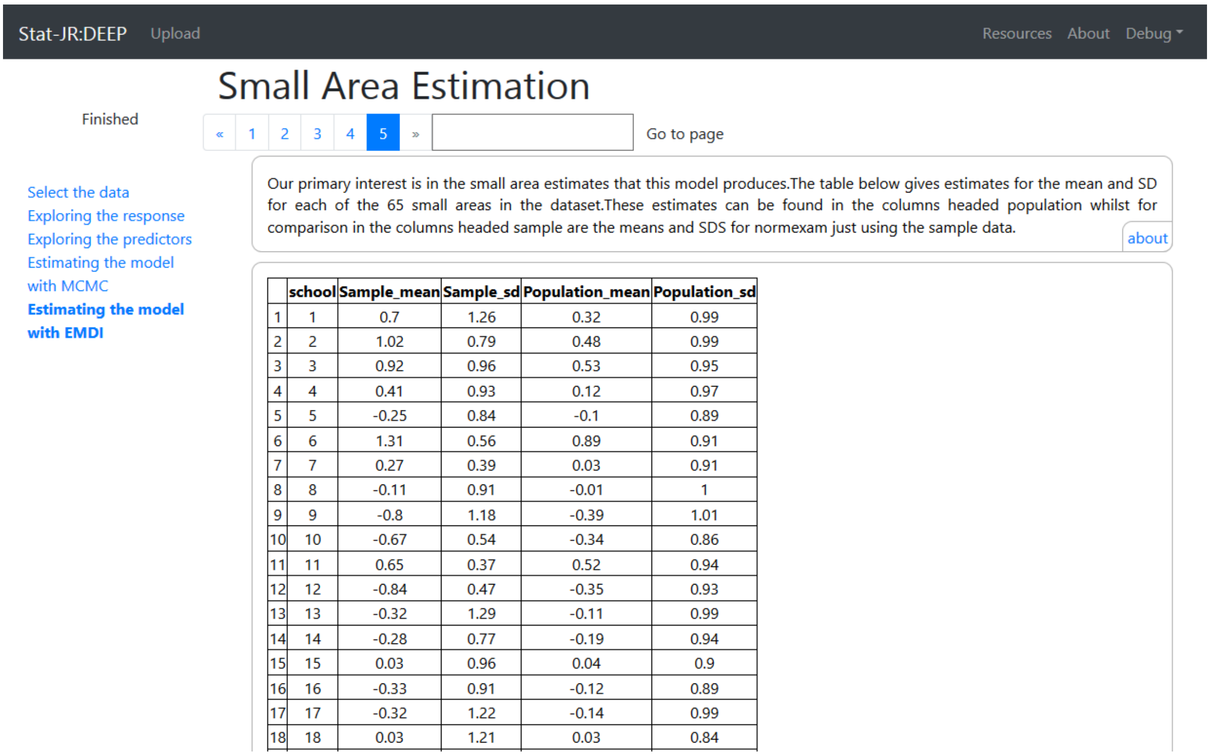
For comparison we can also fit models using interoperability with the R statistical software and the emdi package. The emdi package only fits Normal response models to continuous data but does allow a selection of transformations – identity, log and Box-Cox. Below you are asked again to input options for the model.

Response variable:	normexam	change
Transformation:	no	change
Common 2 ID:	school	change
Common predictor variables:	standlrt,girl	change
Do you want to calculate poverty related estimates, e.g. Head count ratio?:	No	change
Do you want to calculate inequality related estimates, e.g. GINI index?:	No	change
Parallel cores for bootstrapping:	6	change

As you can see we use the same inputs here although the names are sometimes slightly different. When we click on **Submit** you will see that the top left progress indicator now says *Running R_script* to explain that it is calling R in the background. If you haven't previously used emdi in the R installed on your machine then it may take a while to install all the required packages which Stat-JR does in the background. The outputs begin with some diagnostic plots:



Here you can see a plot of model fit. EMDI is showing how good a fit the normal model is to the data at both levels of the multilevel model in two separate plots. There are also some QQ plots to also look at model fit before some of the more familiar outputs like the table of sample of population estimates shown below:



Our primary interest is in the small area estimates that this model produces.The table below gives estimates for the mean and SD for each of the 65 small areas in the dataset.These estimates can be found in the columns headed population whilst for comparison in the columns headed sample are the means and SDS for normexam just using the sample data.

	school	Sample_mean	Sample_sd	Population_mean	Population_sd
1	1	0.7	1.26	0.32	0.99
2	2	1.02	0.79	0.48	0.99
3	3	0.92	0.96	0.53	0.95
4	4	0.41	0.93	0.12	0.97
5	5	-0.25	0.84	-0.1	0.89
6	6	1.31	0.56	0.89	0.91
7	7	0.27	0.39	0.03	0.91
8	8	-0.11	0.91	-0.01	1
9	9	-0.8	1.18	-0.39	1.01
10	10	-0.67	0.54	-0.34	0.86
11	11	0.65	0.37	0.52	0.94
12	12	-0.84	0.47	-0.35	0.93
13	13	-0.32	1.29	-0.11	0.99
14	14	-0.28	0.77	-0.19	0.94
15	15	0.03	0.96	0.04	0.9
16	16	-0.33	0.91	-0.12	0.89
17	17	-0.32	1.22	-0.14	0.99
18	18	0.03	1.21	0.03	0.84

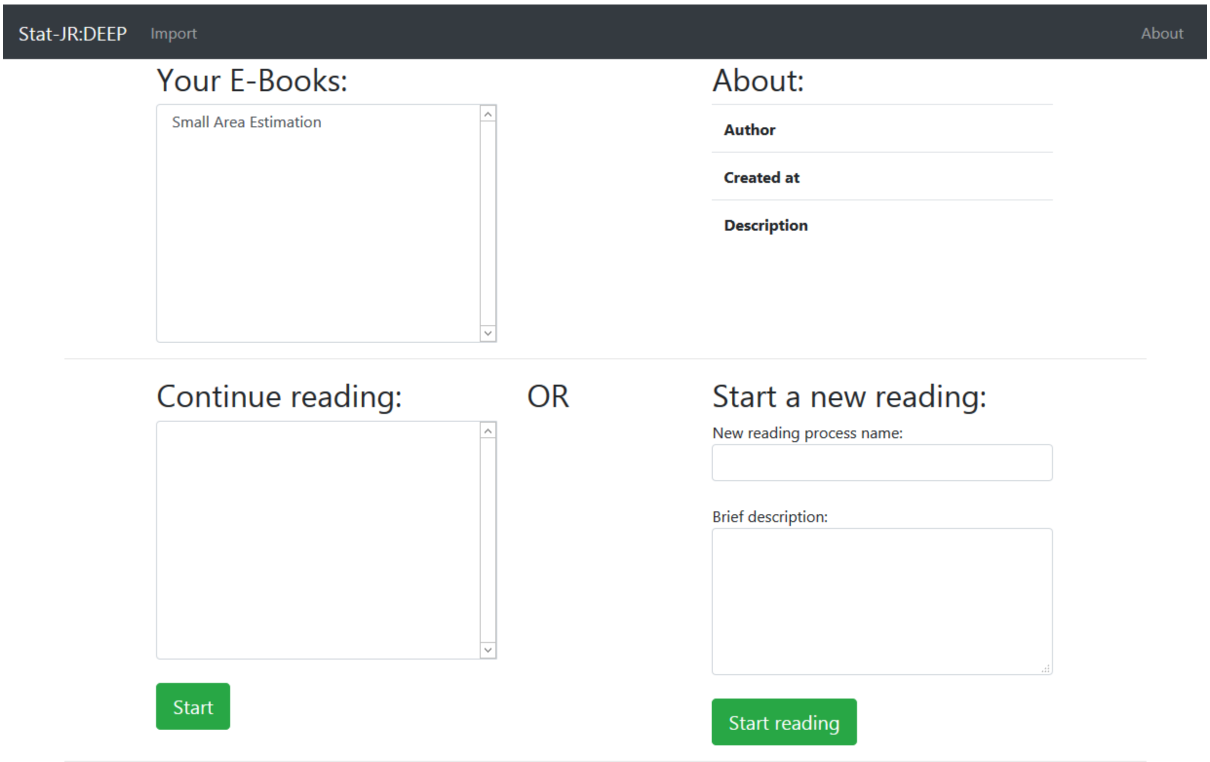
Here you might like to compare the (population) estimates given on this page with those from MCMC on page 4. They will not be identical as the two approaches used are different but there should be similarity in the patterns shown. This page then continues with various plots for the means and quantiles which again you can compare with the MCMC approach. We will leave you at this point to explore the eBook in more detail, after all the eBook is meant to be self-contained and instead we will move onto a second SAE example.

Example 2 - The European Union Statistics on Incom and Living Conditions – Austria (EU-SILC A) dataset

As mentioned previously our first education-based example was a slightly less common use of small area estimation. In most applications of SAE the areas are generally geographical areas (as opposed to schools) and the population is the whole population in those areas whilst the sample is some form of census for a smaller group of people from this whole population. The EU-SILC Austrian dataset is the example that is used in the vignette document (Kreutzmann et al. 2018) that was written as documentation for the emdi package. The original (full) EU-SILC data comes from a household survey and there are 8.8 Million people in Austria with Austria split into 94 districts which play the role of small areas. The data that we in fact use is simulated and originates from the R package simFrame (Alfons et al. 2010). Here we have a (reduced) population dataset, *eusilcA_pop* of 25,000 people and a smaller sample dataset, *eusilcA_smp* of 1,945 individuals. As with example 1 the sample dataset in EU-SILC contains individuals from some but not all of the small areas in the population. The motivation behind the EU-SILC example is to look at income across Austria and in particular income inequalities and poverty rates. This is an application area that often uses SAE methods and our variable of interest will be (equivalized) household income – a household income adjusted for size of household. The dataset contains several predictor variables like the household size, the gender of the person sampled and various benefit type measures like unemployment, old age, sickness etc. that can be used to predict the income.

There are two main motivations for using a second example here which cover features that were not present in the first example. First, the response variable, *normexam* that we used in example 1 had a symmetric distribution that was clearly normally distributed. In this example household income has by no means a symmetric distribution and so we will need to transform the variable of interest prior to fitting models to it. Second as we are interested in poverty and inequality there are a broader selection of summary measures that we can estimate for each small area.

We could of course repeat the structure of example 1 and work our way slowly through several templates using the TREE interface but here we will jump straight to the DEEP interface. So we start by bringing up the DEEP interface (if you are proceeding on from the last section then simply click on the white Stat-JR:DEEP in the top left to return to the opening screen). Assuming you have already looked at example 1 in DEEP then the opening screen should look as follows:



Your E-Books:

Small Area Estimation

About:

Author

Created at

Description

Continue reading:

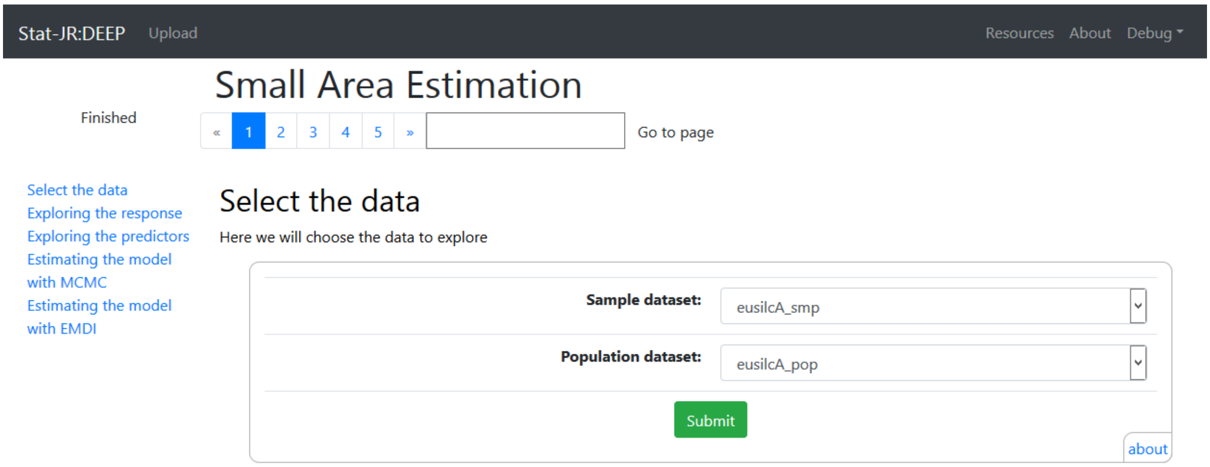
OR

Start a new reading:

New reading process name:

Brief description:

Here we see that the eBook for Small Area Estimation is still listed in the list of eBooks and we can click on it type in a new reading process name e.g. eusilc in the box towards the bottom right and click on the green **Start Reading** button. Here once again we are greeted with the 5 pages and we can on page 1 select the appropriate datasets as shown below:



Small Area Estimation

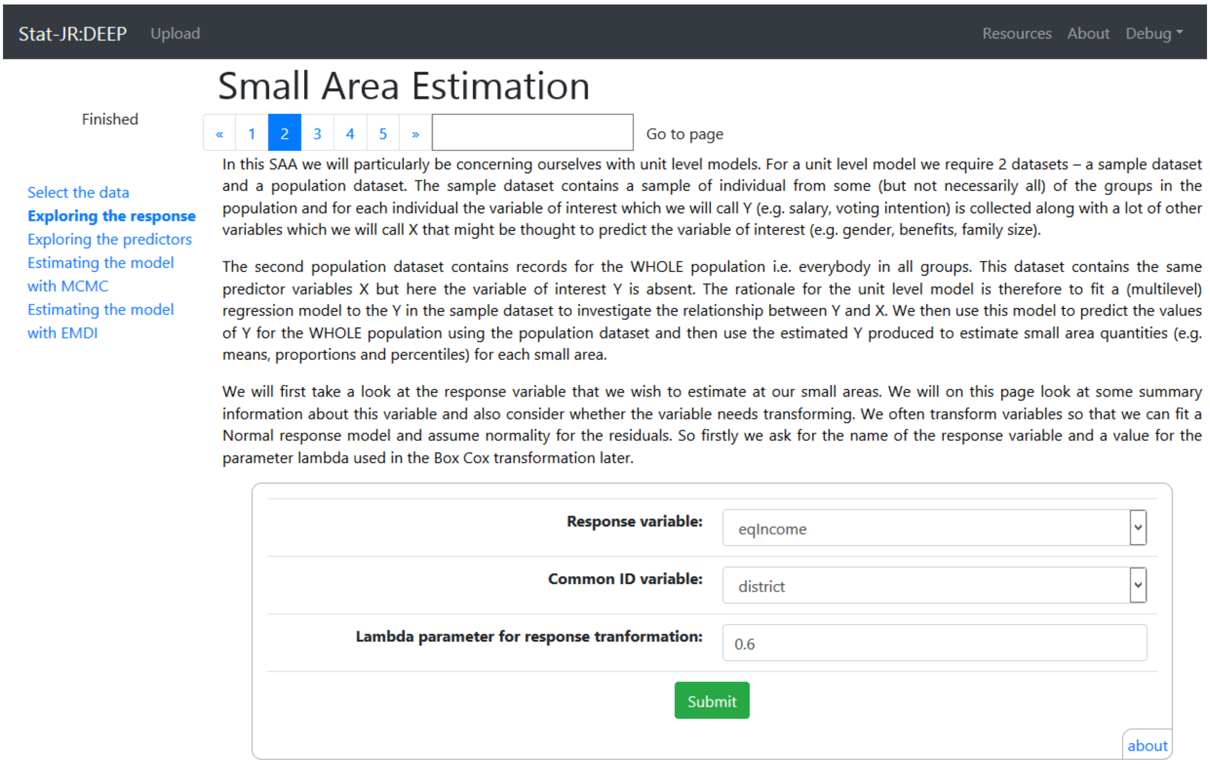
Select the data

Here we will choose the data to explore

Sample dataset: eusilcA_smp

Population dataset: eusilcA_pop

Now we click on the **Submit** button and move on to page 2 where we will explore the (response) variable of interest, *eqIncome.* On page 2 we can choose this response and note that the **Common ID variable** i.e. the variable that indicates the small areas is *district.* We also need to input a lambda value and here we will try 0.6.



Small Area Estimation

In this SAA we will particularly be concerning ourselves with unit level models. For a unit level model we require 2 datasets – a sample dataset and a population dataset. The sample dataset contains a sample of individual from some (but not necessarily all) of the groups in the population and for each individual the variable of interest which we will call Y (e.g. salary, voting intention) is collected along with a lot of other variables which we will call X that might be thought to predict the variable of interest (e.g. gender, benefits, family size).

The second population dataset contains records for the WHOLE population i.e. everybody in all groups. This dataset contains the same predictor variables X but here the variable of interest Y is absent. The rationale for the unit level model is therefore to fit a (multilevel) regression model to the Y in the sample dataset to investigate the relationship between Y and X. We then use this model to predict the values of Y for the WHOLE population using the population dataset and then use the estimated Y produced to estimate small area quantities (e.g. means, proportions and percentiles) for each small area.

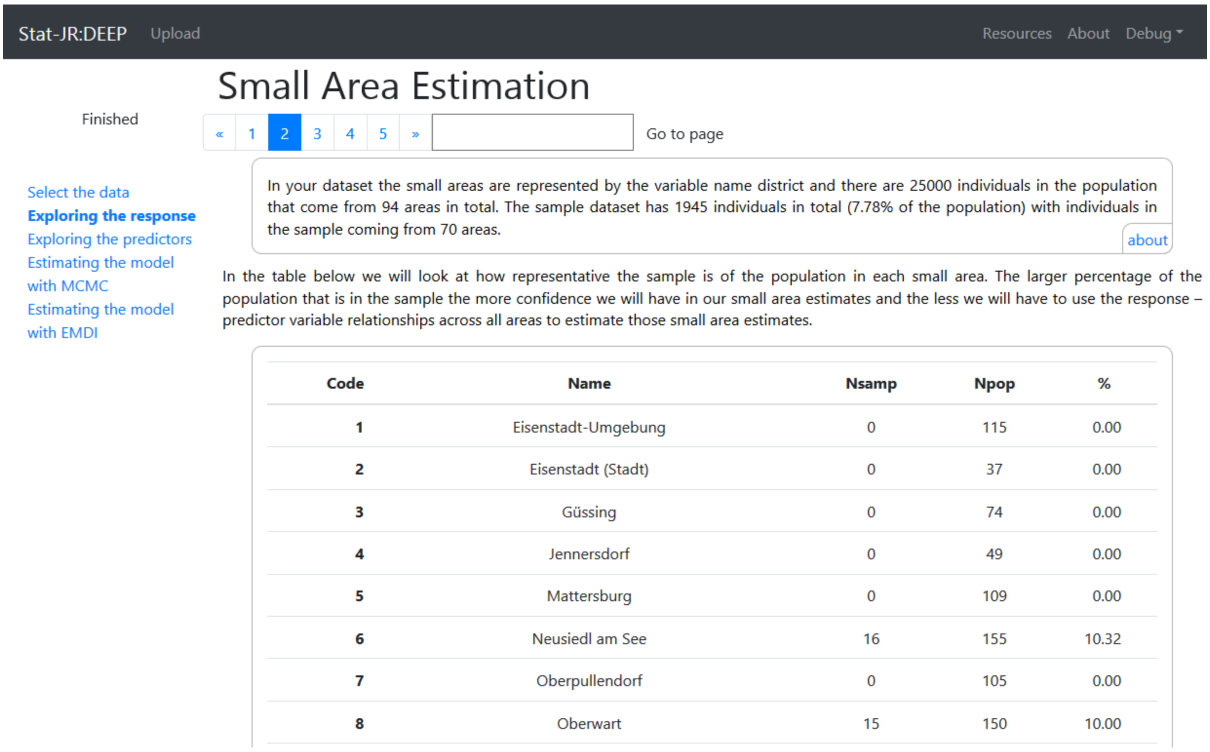
We will first take a look at the response variable that we wish to estimate at our small areas. We will on this page look at some summary information about this variable and also consider whether the variable needs transforming. We often transform variables so that we can fit a Normal response model and assume normality for the residuals. So firstly we ask for the name of the response variable and a value for the parameter lambda used in the Box Cox transformation later.

Response variable: eqIncome

Common ID variable: district

Lambda parameter for response tranformation: 0.6

We can now click on the green **Submit** button and observe the outputs that this will generate:



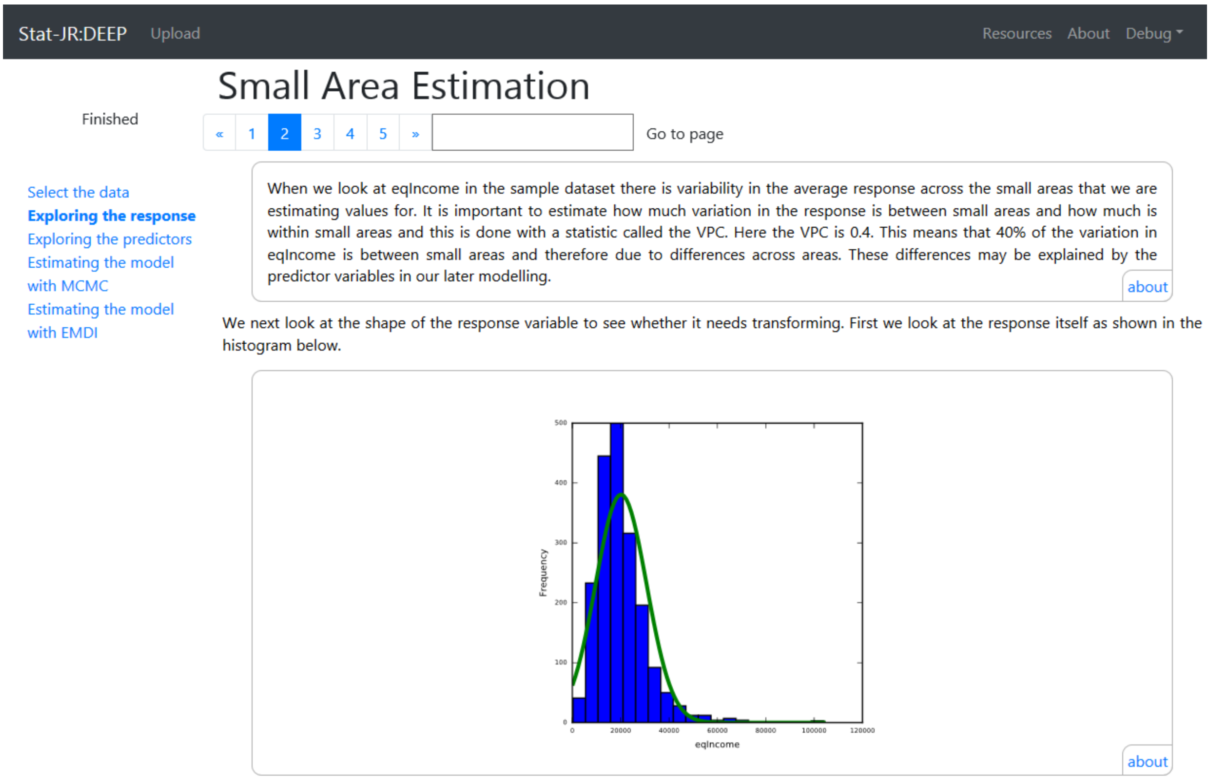
Small Area Estimation

In your dataset the small areas are represented by the variable name district and there are 25000 individuals in the population that come from 94 areas in total. The sample dataset has 1945 individuals in total (7.78% of the population) with individuals in the sample coming from 70 areas.

In the table below we will look at how representative the sample is of the population in each small area. The larger percentage of the population that is in the sample the more confidence we will have in our small area estimates and the less we will have to use the response – predictor variable relationships across all areas to estimate those small area estimates.

Code	Name	Nsamp	Npop	%
1	Eisenstadt-Umgebung	0	115	0.00
2	Eisenstadt (Stadt)	0	37	0.00
3	Güssing	0	74	0.00
4	Jennersdorf	0	49	0.00
5	Mattersburg	0	109	0.00
6	Neusiedl am See	16	155	10.32
7	Oberpullendorf	0	105	0.00
8	Oberwart	15	150	10.00

So initially we get confirmation of the population and sample dataset sizes and that we have slightly under 8% of the population in the sample with individuals from 70 of the 94 areas. Unlike the first example the areas have been given actual names so we can see them in the table above and in fact 6 of the first 8 areas have no individuals in the sample whilst the other 2 have approximately 10% in the sample. In fact the (simulated) sample design appears to be to choose a sample of areas (in this case 70 of the 94) and for each area chosen sample approximately 10% of the population. If we scroll lower down we see information about the VPC and a plot of the response:

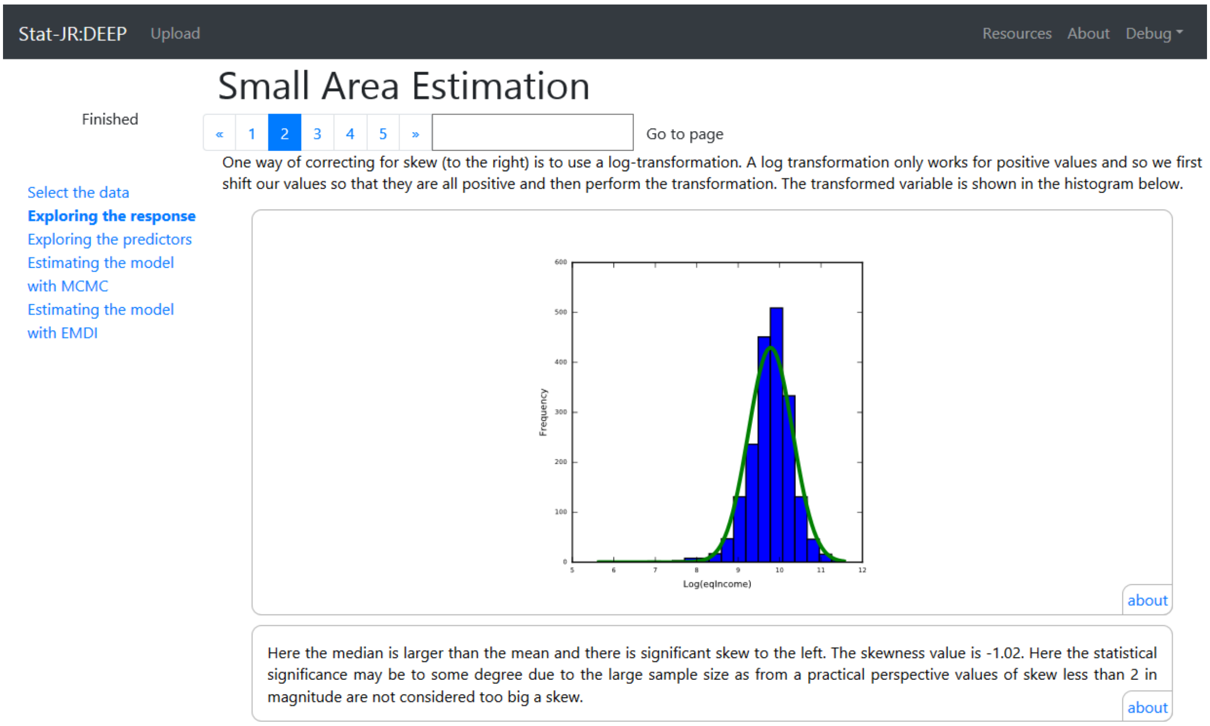


Small Area Estimation

When we look at eqIncome in the sample dataset there is variability in the average response across the small areas that we are estimating values for. It is important to estimate how much variation in the response is between small areas and how much is within small areas and this is done with a statistic called the VPC. Here the VPC is 0.4. This means that 40% of the variation in eqIncome is between small areas and therefore due to differences across areas. These differences may be explained by the predictor variables in our later modelling.

We next look at the shape of the response variable to see whether it needs transforming. First we look at the response itself as shown in the histogram below.

Here we see that the small areas in this example have greater differences than the schools in example 1 with 40% of the variability in *eqIncome* being attributed to small area differences. The graph below shows perhaps a typical income distribution with a skew to the right i.e. some households earning far more than the majority. This will suggest that we will need to transform the variable when we come to do our SAE modelling. The page next shows possible transforms so below we see the log transform:

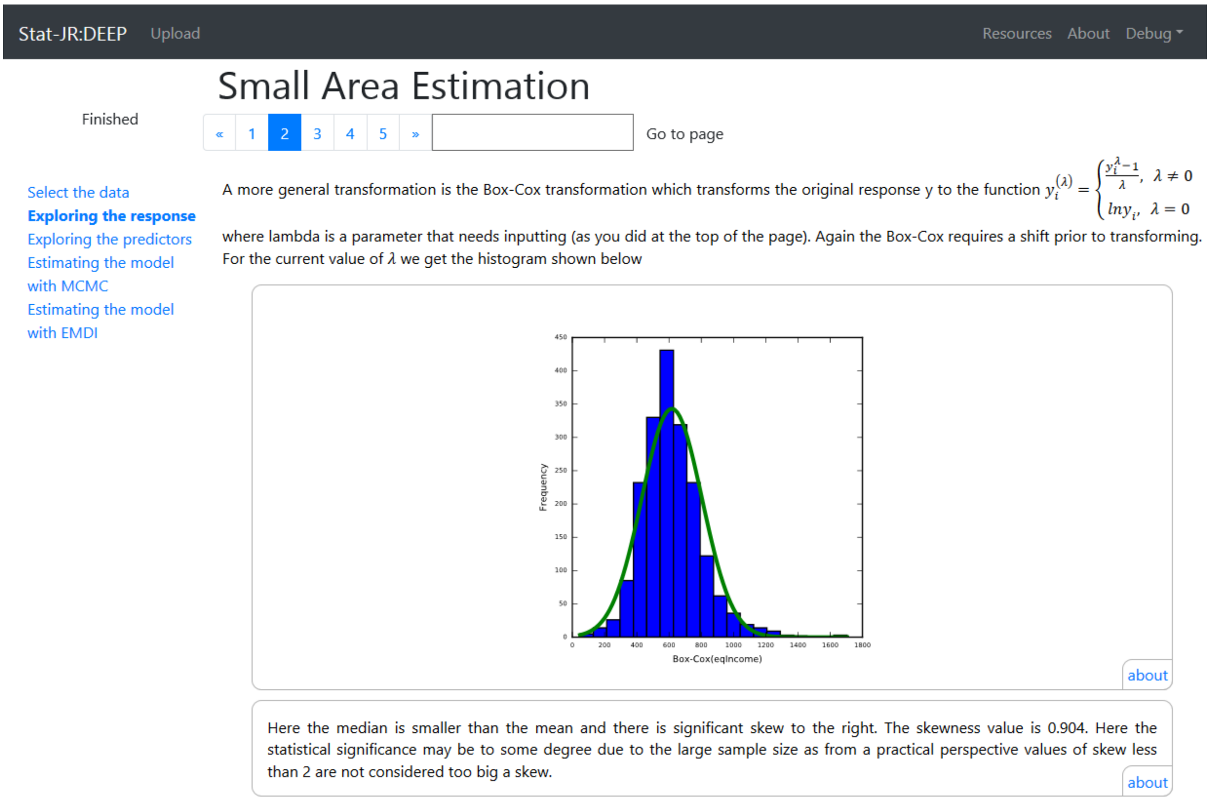


Small Area Estimation

One way of correcting for skew (to the right) is to use a log-transformation. A log transformation only works for positive values and so we first shift our values so that they are all positive and then perform the transformation. The transformed variable is shown in the histogram below.

Here the median is larger than the mean and there is significant skew to the left. The skewness value is -1.02. Here the statistical significance may be to some degree due to the large sample size as from a practical perspective values of skew less than 2 in magnitude are not considered too big a skew.

This does a reasonable job though seems to end up with a few low values now and thus a skew to the left. The second possible transform is the Box-Cox transform (with fixed parameter lambda that we have set to 0.6 and is shown as follows:



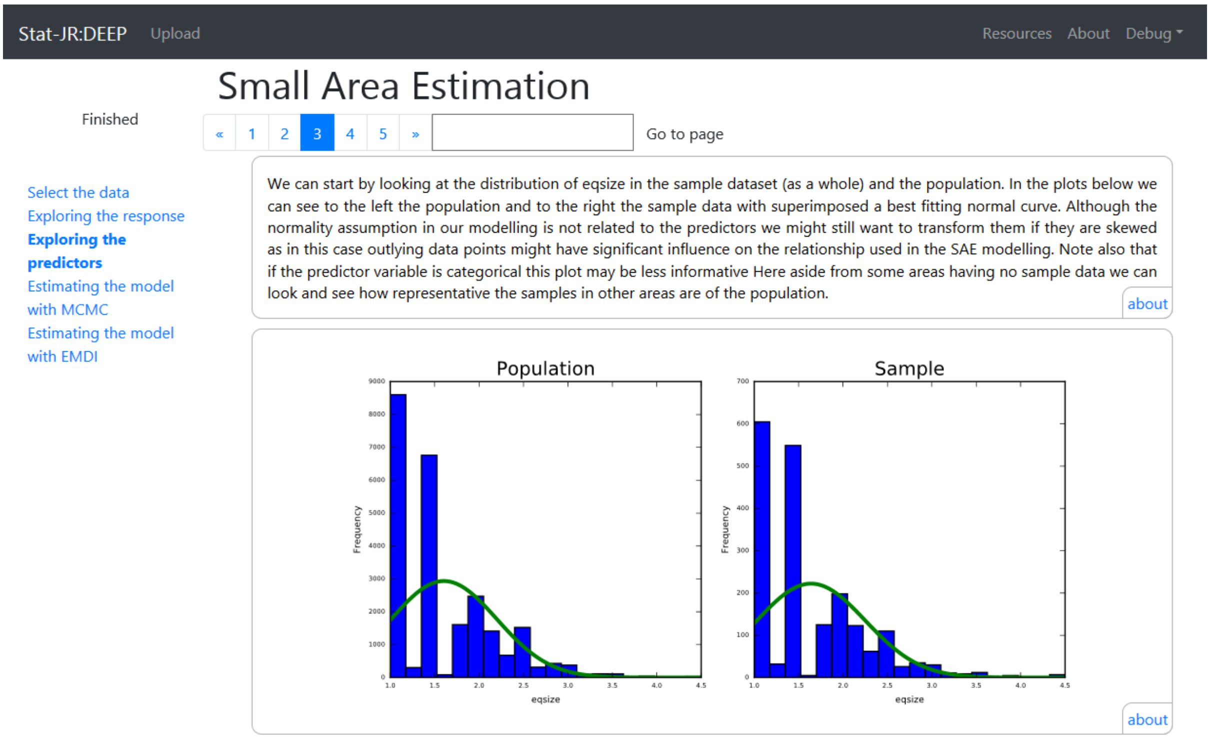
Small Area Estimation

A more general transformation is the Box-Cox transformation which transforms the original response y to the function $y_i^{(\lambda)} = \begin{cases} \frac{y_i^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ lny_i, & \lambda = 0 \end{cases}$ where lambda is a parameter that needs inputting (as you did at the top of the page). Again the Box-Cox requires a shift prior to transforming. For the current value of λ we get the histogram shown below

Here the median is smaller than the mean and there is significant skew to the right. The skewness value is 0.904. Here the statistical significance may be to some degree due to the large sample size as from a practical perspective values of skew less than 2 are not considered too big a skew.

Here we see that the transform has reduced the skew though the histogram is not perfect. It is worth remembering when we come to modelling that it is the residuals after fitting models that should be normally distributed rather than the response per se though of course having a symmetric response is helpful. We will in a little while select the Box Cox transform as our choice but when we do that the value of lambda will be modelled rather than fixed to 0.6 as we see here. The page also gives a third transform for completeness, the Dual Power transform but the modelling on pages 4 and 5 does not allow us to choose this transform so we will ignore it here.

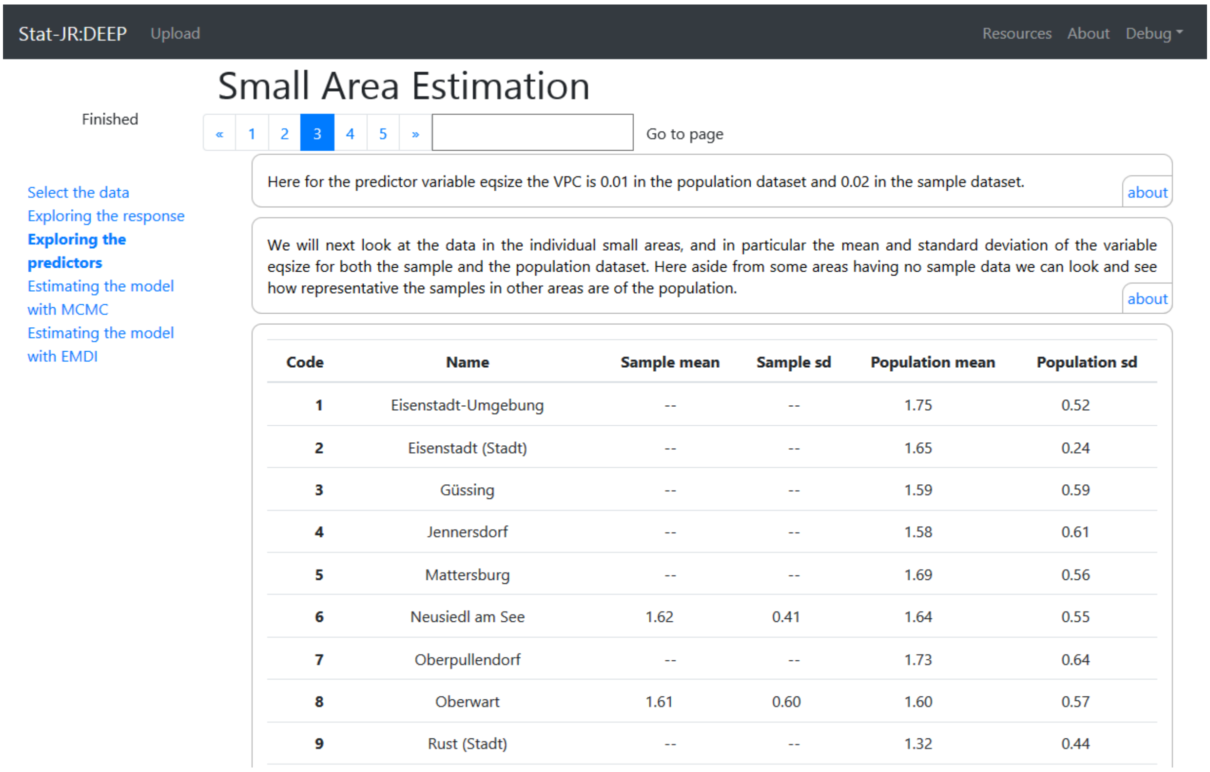
If we move on to page 3 we can again look at the distribution of the possible predictor variables within both the population and sample. Feel free to test this out for various predictors as in our modelling we will include ALL predictors from *gender* through to *tax_adj.* As an example we have chosen the variable, *eqsize* (with **Common ID variable** *district*) and we can see the distribution as follows:



Small Area Estimation

We can start by looking at the distribution of eqsize in the sample dataset (as a whole) and the population. In the plots below we can see to the left the population and to the right the sample data with superimposed a best fitting normal curve. Although the normality assumption in our modelling is not related to the predictors we might still want to transform them if they are skewed as in this case outlying data points might have significant influence on the relationship used in the SAE modelling. Note also that if the predictor variable is categorical this plot may be less informative Here aside from some areas having no sample data we can look and see how representative the samples in other areas are of the population.

The main things to note here are the somewhat unusual shape of the distribution but that in practice the shape is very similar for both the population and the sample and this can be seen again in the later plot where both histograms are superimposed on each other. There are various plots and tables later in the page and we show a screenshot below:



Small Area Estimation

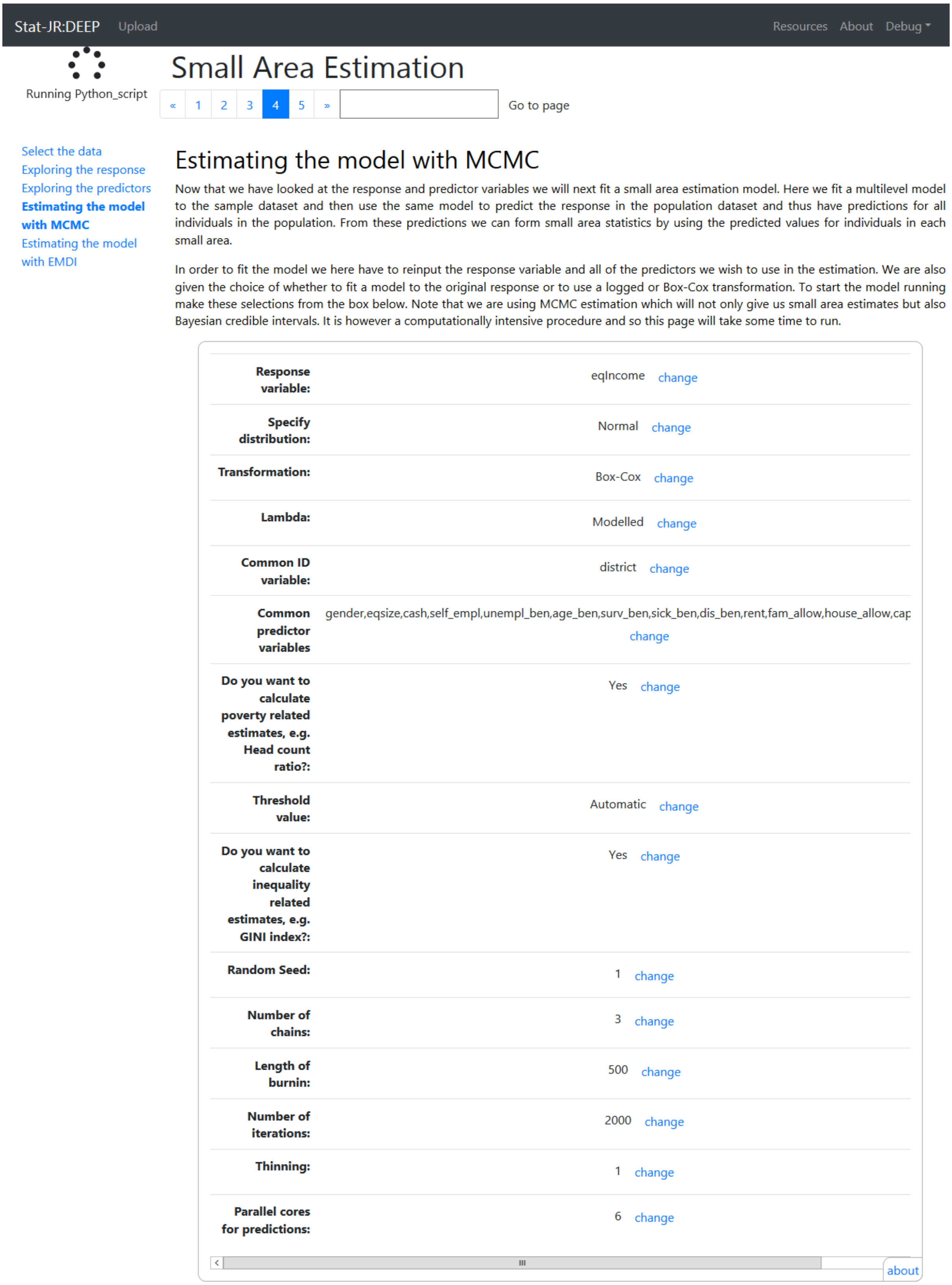
Here for the predictor variable eqsize the VPC is 0.01 in the population dataset and 0.02 in the sample dataset.

We will next look at the data in the individual small areas, and in particular the mean and standard deviation of the variable eqsize for both the sample and the population dataset. Here aside from some areas having no sample data we can look and see how representative the samples in other areas are of the population.

Code	Name	Sample mean	Sample sd	Population mean	Population sd
1	Eisenstadt-Umgebung	--	--	1.75	0.52
2	Eisenstadt (Stadt)	--	--	1.65	0.24
3	Güssing	--	--	1.59	0.59
4	Jennersdorf	--	--	1.58	0.61
5	Mattersburg	--	--	1.69	0.56
6	Neusiedl am See	1.62	0.41	1.64	0.55
7	Oberpullendorf	--	--	1.73	0.64
8	Oberwart	1.61	0.60	1.60	0.57
9	Rust (Stadt)	--	--	1.32	0.44

Here we can see that perhaps unsurprisingly district doesn’t explain much of the variation in household size with the VPC of 1% and 2% for population and sample respectively. Also for the 2 districts shown that have sample data, the summary statistics (mean and sd) are rather similar.

We will leave you to look at the other outputs and indeed the other possible predictors and now move on to page 4 and some modelling and set the inputs as shown:



Small Area Estimation

Estimating the model with MCMC

Now that we have looked at the response and predictor variables we will next fit a small area estimation model. Here we fit a multilevel model to the sample dataset and then use the same model to predict the response in the population dataset and thus have predictions for all individuals in the population. From these predictions we can form small area statistics by using the predicted values for individuals in each small area.

In order to fit the model we here have to reinput the response variable and all of the predictors we wish to use in the estimation. We are also given the choice of whether to fit a model to the original response or to use a logged or Box-Cox transformation. To start the model running make these selections from the box below. Note that we are using MCMC estimation which will not only give us small area estimates but also Bayesian credible intervals. It is however a computationally intensive procedure and so this page will take some time to run.

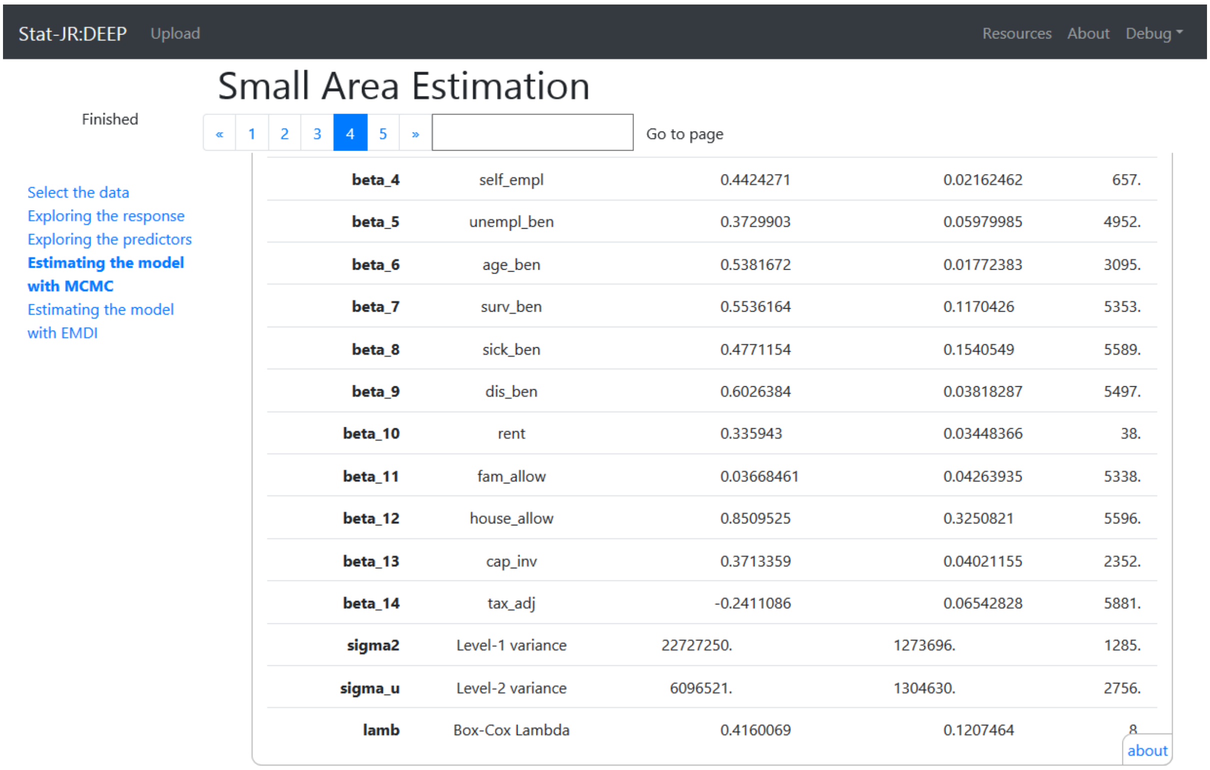
Response variable:	eqIncome change
Specify distribution:	Normal change
Transformation:	Box-Cox change
Lambda:	Modelled change
Common ID variable:	district change
Common predictor variables	gender,eqsize,cash,self_empl,unempl_ben,age_ben,surv_ben,sick_ben,dis_ben,rent,fam_allow,house_allow,cap change
Do you want to calculate poverty related estimates, e.g. Head count ratio?:	Yes change
Threshold value:	Automatic change
Do you want to calculate inequality related estimates, e.g. GINI index?:	Yes change
Random Seed:	1 change
Number of chains:	3 change
Length of burnin:	500 change
Number of iterations:	2000 change
Thinning:	1 change
Parallel cores for predictions:	6 change

Here we start with selecting the response, *eqIncome* and *Normal* distribution and then we chose the *Box-Cox* transform. Here we select that the **Lambda** parameter will be *Modelled* which means that it will be treated as a parameter in the MCMC algorithm and thus be updated at each MCMC iteration. We see the long list of predictor variables that we have chosen (from *gender* down to *tax_adj*) as well as that we have requested to calculate both poverty related and inequality related quantities.

You will also see that a threshold value is asked for with choices of *Automatic* or *Manual*. This threshold is used in the poverty related measures to represent a poverty line and work out statistics based around this. If *Automatic* is chosen then for this dataset the value 10885.33 is used which is 60% of the median equivalised income as is used in Kreutzmann et al. (2018). The MCMC inputs are finally asked for which we have been left at their defaults.

Having completed all the inputs and clicked on **Submit** we have to wait a little longer than example 1 as the dataset is bigger, has more predictor variables and modelling the lambda value and Box-Cox transformation adds to the computation.

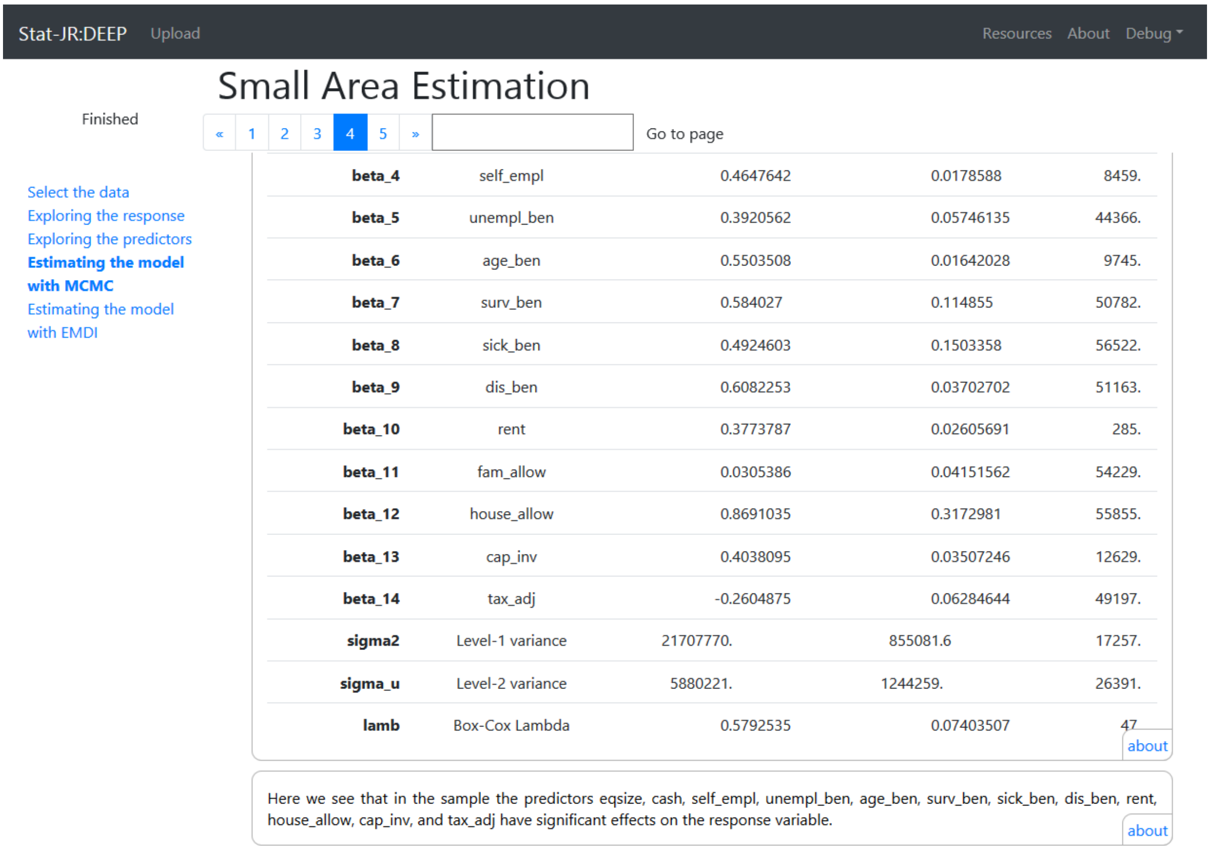
If we now look at the outputs on page 4 we can scroll past the model equations and next see the model parameters:



beta_4	self_empl	0.4424271	0.02162462	657.
beta_5	unempl_ben	0.3729903	0.05979985	4952.
beta_6	age_ben	0.5381672	0.01772383	3095.
beta_7	surv_ben	0.5536164	0.1170426	5353.
beta_8	sick_ben	0.4771154	0.1540549	5589.
beta_9	dis_ben	0.6026384	0.03818287	5497.
beta_10	rent	0.335943	0.03448366	38.
beta_11	fam_allow	0.03668461	0.04263935	5338.
beta_12	house_allow	0.8509525	0.3250821	5596.
beta_13	cap_inv	0.3713359	0.04021155	2352.
beta_14	tax_adj	-0.2411086	0.06542828	5881.
sigma2	Level-1 variance	22727250.	1273696.	1285.
sigma_u	Level-2 variance	6096521.	1304630.	2756.
lamb	Box-Cox Lambda	0.4160069	0.1207464	8.

Here you will see after the 15 beta coefficients for the intercept and predictors and the 2 variances then we have also the estimated lambda (*lamb*) which takes value of 0.42 but note that here we are looking at the best value when accounting for the predictor variables. One thing to watch is the effective sample size (ESS) for this parameter is rather low at only 8 as is the ESS for the intercept and so we should really run the MCMC for more iterations. Here we therefore click on the blue *change* next to *Number of iterations* and change the 2000 to 20000 before pressing the **Submit** button and waiting again this time slightly longer for estimation.

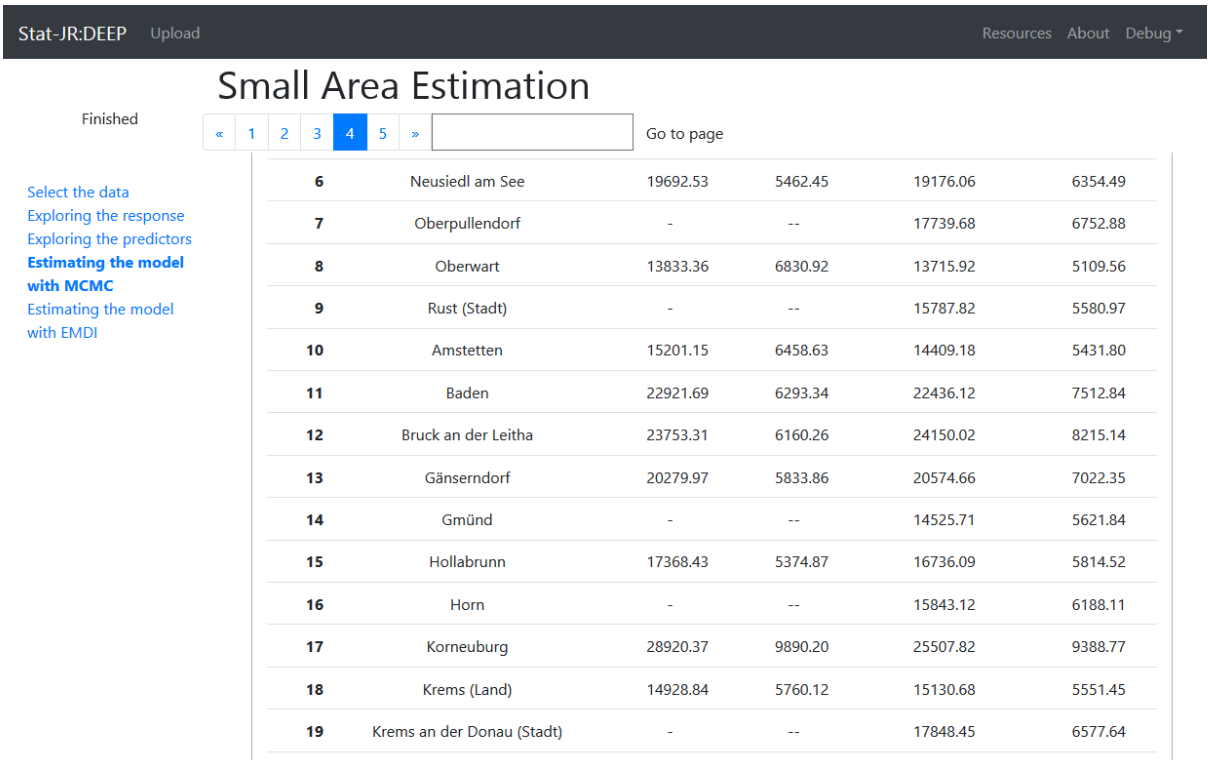
The estimates now should refresh and look as follows:



beta_4	self_empl	0.4647642	0.0178588	8459.
beta_5	unempl_ben	0.3920562	0.05746135	44366.
beta_6	age_ben	0.5503508	0.01642028	9745.
beta_7	surv_ben	0.584027	0.114855	50782.
beta_8	sick_ben	0.4924603	0.1503358	56522.
beta_9	dis_ben	0.6082253	0.03702702	51163.
beta_10	rent	0.3773787	0.02605691	285.
beta_11	fam_allow	0.0305386	0.04151562	54229.
beta_12	house_allow	0.8691035	0.3172981	55855.
beta_13	cap_inv	0.4038095	0.03507246	12629.
beta_14	tax_adj	-0.2604875	0.06284644	49197.
sigma2	Level-1 variance	21707770.	855081.6	17257.
sigma_u	Level-2 variance	5880221.	1244259.	26391.
lamb	Box-Cox Lambda	0.5792535	0.07403507	47.

Here we see that in the sample the predictors eqsize, cash, self_empl, unempl_ben, age_ben, surv_ben, sick_ben, dis_ben, rent, house_allow, cap_inv, and tax_adj have significant effects on the response variable.

If we scroll further we can see the sample (direct) estimates and population (model-based) estimates as shown below for a selection of districts:

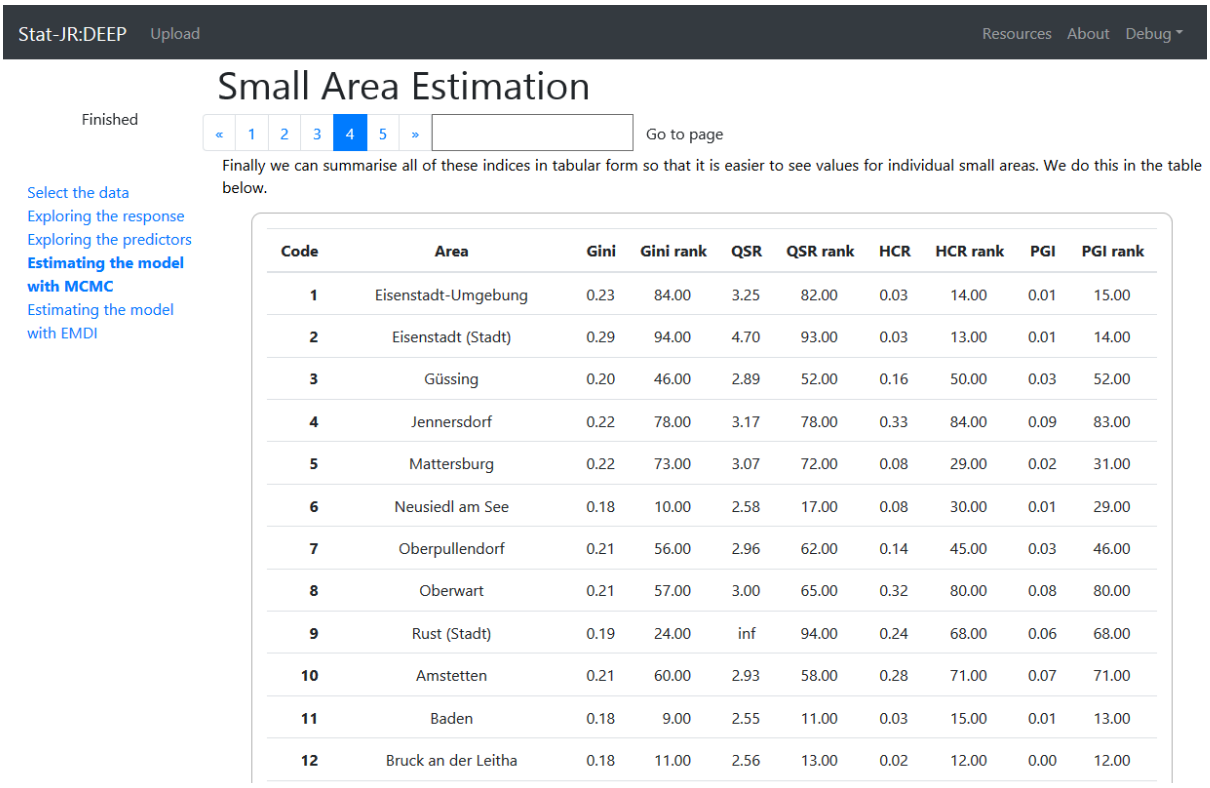


6	Neusiedl am See	19692.53	5462.45	19176.06	6354.49
7	Oberpullendorf	-	--	17739.68	6752.88
8	Oberwart	13833.36	6830.92	13715.92	5109.56
9	Rust (Stadt)	-	--	15787.82	5580.97
10	Amstetten	15201.15	6458.63	14409.18	5431.80
11	Baden	22921.69	6293.34	22436.12	7512.84
12	Bruck an der Leitha	23753.31	6160.26	24150.02	8215.14
13	Gänserndorf	20279.97	5833.86	20574.66	7022.35
14	Gmünd	-	--	14525.71	5621.84
15	Hollabrunn	17368.43	5374.87	16736.09	5814.52
16	Horn	-	--	15843.12	6188.11
17	Korneuburg	28920.37	9890.20	25507.82	9388.77
18	Krems (Land)	14928.84	5760.12	15130.68	5551.45
19	Krems an der Donau (Stadt)	-	--	17848.45	6577.64

It is worth observing that there are sometimes large discrepancies between the 2 estimates for a district of the order of thousands of Euro but in general the pattern across districts is similar for the

2 estimates i.e. those districts with larger sample estimates (eg. Korneuburg) have larger population estimates and one advantage of the model-based approach is we gain estimates for districts with no sample data.

The page contains many tables and graphs and these are explained in the text accompanying them so we will here simply show the final table which includes both the inequality and poverty indices:



Small Area Estimation

Finally we can summarise all of these indices in tabular form so that it is easier to see values for individual small areas. We do this in the table below.

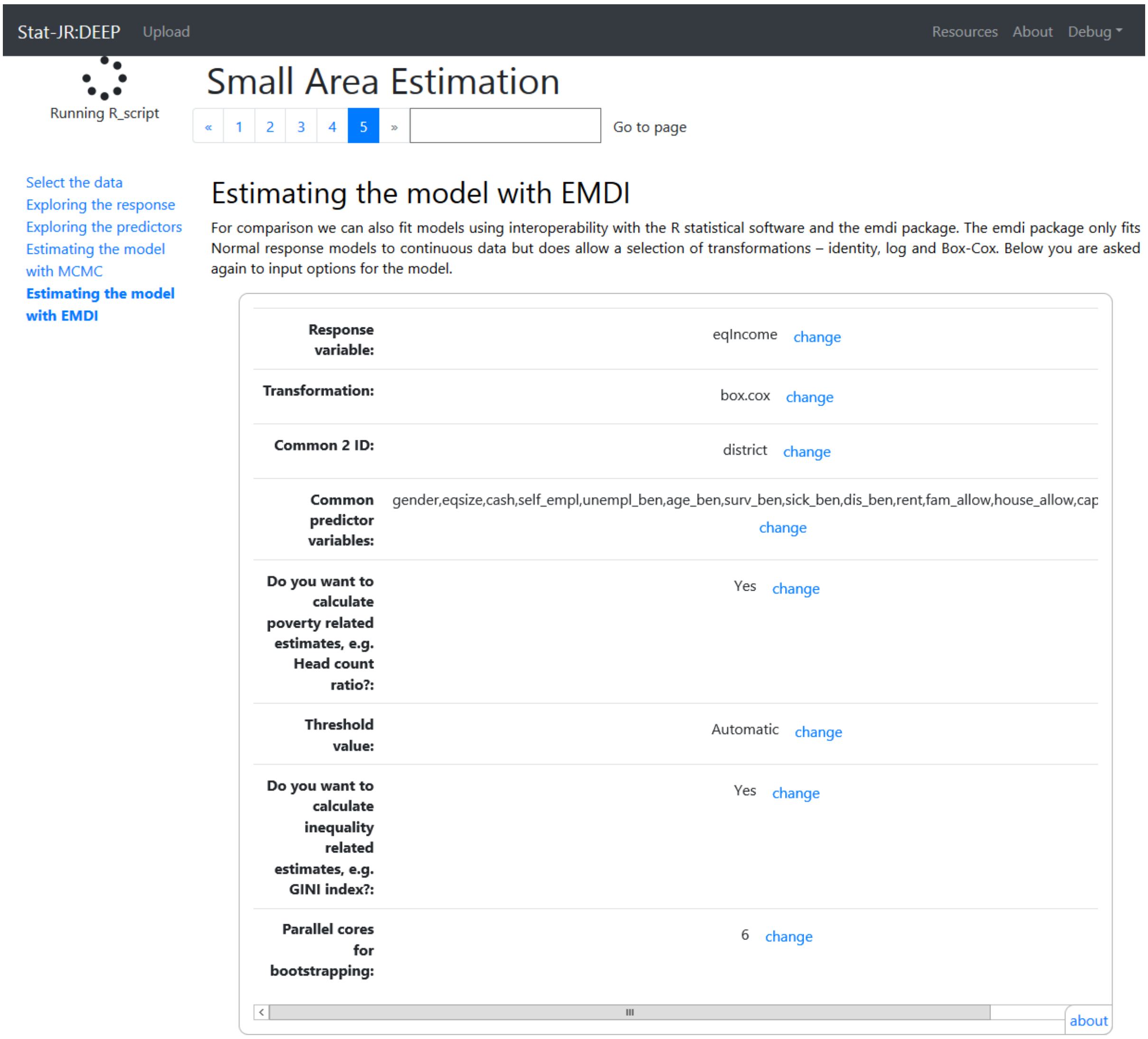
Code	Area	Gini	Gini rank	QSR	QSR rank	HCR	HCR rank	PGI	PGI rank
1	Eisenstadt-Umgebung	0.23	84.00	3.25	82.00	0.03	14.00	0.01	15.00
2	Eisenstadt (Stadt)	0.29	94.00	4.70	93.00	0.03	13.00	0.01	14.00
3	Güssing	0.20	46.00	2.89	52.00	0.16	50.00	0.03	52.00
4	Jennersdorf	0.22	78.00	3.17	78.00	0.33	84.00	0.09	83.00
5	Mattersburg	0.22	73.00	3.07	72.00	0.08	29.00	0.02	31.00
6	Neusiedl am See	0.18	10.00	2.58	17.00	0.08	30.00	0.01	29.00
7	Oberpullendorf	0.21	56.00	2.96	62.00	0.14	45.00	0.03	46.00
8	Oberwart	0.21	57.00	3.00	65.00	0.32	80.00	0.08	80.00
9	Rust (Stadt)	0.19	24.00	inf	94.00	0.24	68.00	0.06	68.00
10	Amstetten	0.21	60.00	2.93	58.00	0.28	71.00	0.07	71.00
11	Baden	0.18	9.00	2.55	11.00	0.03	15.00	0.01	13.00
12	Bruck an der Leitha	0.18	11.00	2.56	13.00	0.02	12.00	0.00	12.00

Here we see each of the four indices (which are presented graphically earlier on the page) along with their rank (with rank 1 being the smallest value and rank 94 the largest). The first two indices are the Gini coefficient (Gini) and the income quintile share ratio (QSR) which are more general inequality measures that do not rely on a predefined poverty threshold but instead look at the shape of the income distribution. The Gini takes values from 0 to 1 where the larger the value the greater the income inequality and so a Gini value of 0 would mean all individuals earning equal income and a Gini value of 1 would mean 1 individual earning all the income in the area. For the 12 areas we see above the Gini estimates range from 0.18 to 0.29 with in fact the largest value in an area (*Eisenstadt (Stadt)*) with very little poverty but presumably a few very high earning individuals. The QSR compares the average earnings of the top 20% of earners with the bottom 20% of earners within a district as a ratio. Here we see this ranges for these 12 areas from 2.55 to 4.70 (with Rust (Stadt) not estimating) in other words in the area with the largest value (again *Eisenstadt (Stadt)*) the top 20% earn nearly 5 times as much on average as the bottom 20%.

The second pair of indices are the headcount ratio (HCR) and poverty gap index (PGI) which both relate the data to the threshold input by the user. The HCR is simply the proportion of the population in each small area that are is poverty so for example *Eisenstadt-Umgebung* is estimated to have only 3% in poverty whilst *Jennersdorf* has 33% of its population in poverty. The PGI looks in more detail at how far below the poverty line on average (as a percentage of the poverty line) people's incomes are and again larger values mean worse poverty problems.

We have here only scratched the surface of the outputs that the eBook produces so please feel free to look in more detail at the other plots and tables. You might also investigate what the impact of fitting simpler models with less predictors on the outputs here.

We can finally turn to page 5 and fit the same models using emdi by filling in the inputs as follows:



Small Area Estimation

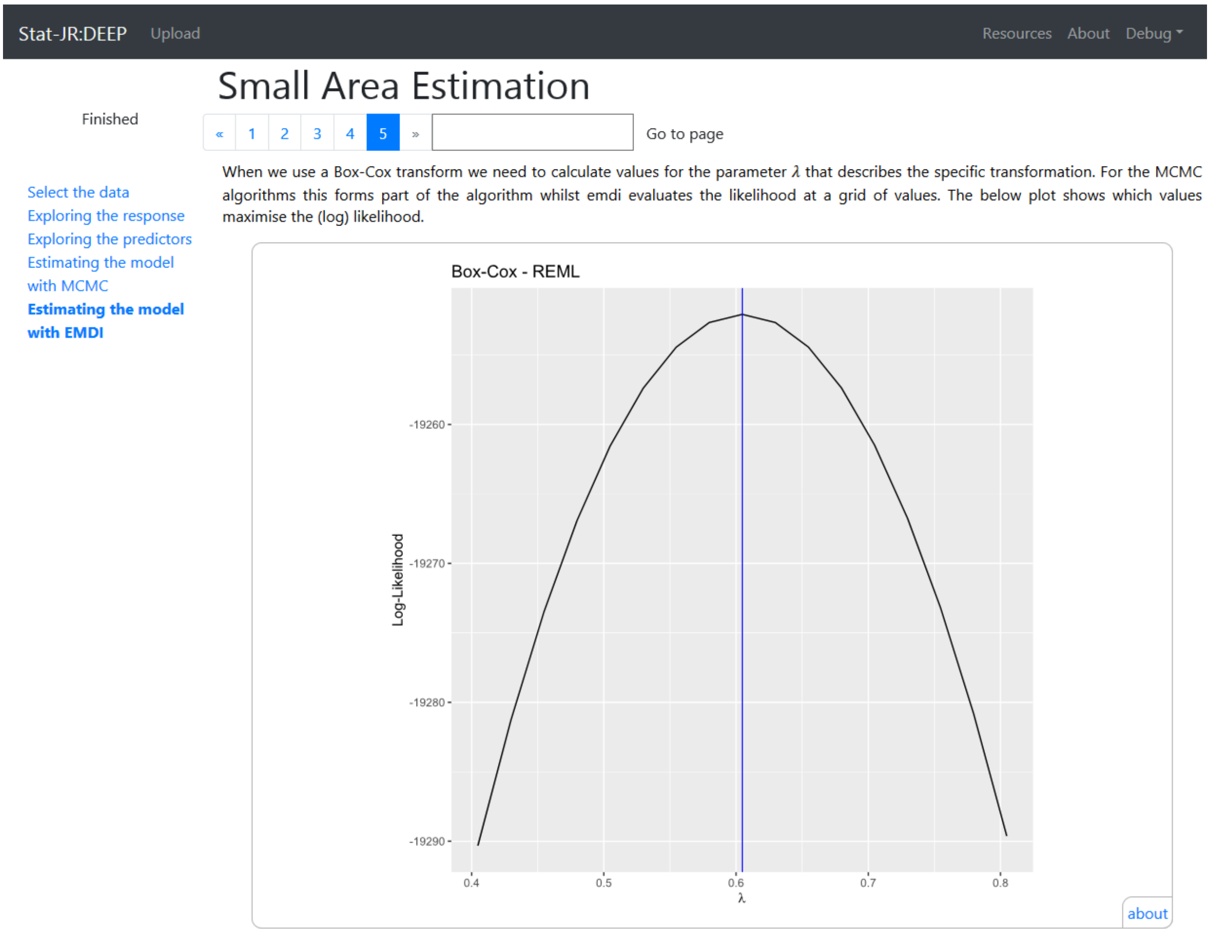
Estimating the model with EMDI

For comparison we can also fit models using interoperability with the R statistical software and the emdi package. The emdi package only fits Normal response models to continuous data but does allow a selection of transformations – identity, log and Box-Cox. Below you are asked again to input options for the model.

Response variable:	eqIncome change
Transformation:	box.cox change
Common 2 ID:	district change
Common predictor variables:	gender,eqsize,cash,self_empl,unempl_ben,age_ben,surv_ben,sick_ben,dis_ben,rent,fam_allow,house_allow,cap change
Do you want to calculate poverty related estimates, e.g. Head count ratio?:	Yes change
Threshold value:	Automatic change
Do you want to calculate inequality related estimates, e.g. GINI index?:	Yes change
Parallel cores for bootstrapping:	6 change

Here we see that the inputs are generally the same as we use for the MCMC method. Upon choosing all the inputs then we click **Submit** and as you see in the top left corner above the eBook runs the R script.

If we consider the outputs there are first once again some outputs that look at the normality of the residuals via comparative density plots and QQ plots and then this is followed by a plot regarding the estimation of the lambda parameter for the Box Cox transform as shown below:

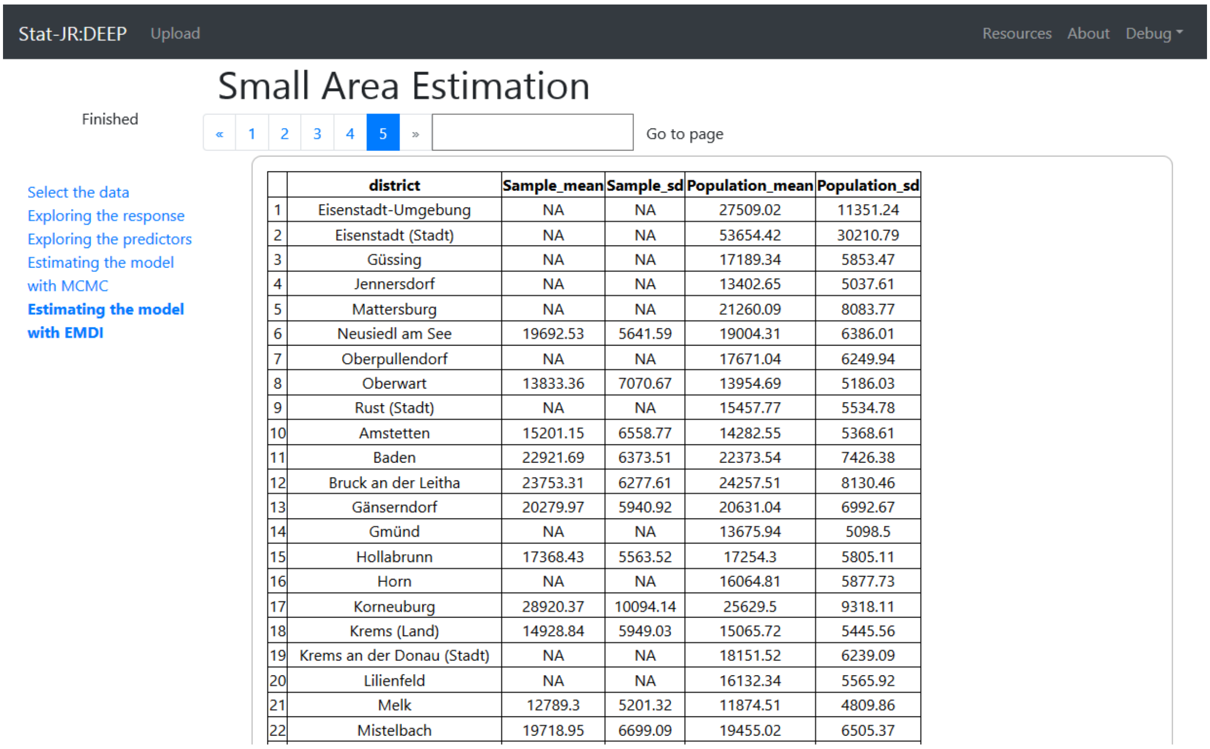


Small Area Estimation

When we use a Box-Cox transform we need to calculate values for the parameter λ that describes the specific transformation. For the MCMC algorithms this forms part of the algorithm whilst emdi evaluates the likelihood at a grid of values. The below plot shows which values maximise the (log) likelihood.

As explained emdi tries a selection (grid) of values for lambda and finds the one that maximises the likelihood. Here emdi estimates an optimal value of 0.6 which is slightly different to MCMC where it took value 0.58.

As with MCMC we get model-based (population) estimates from emdi as shown below:

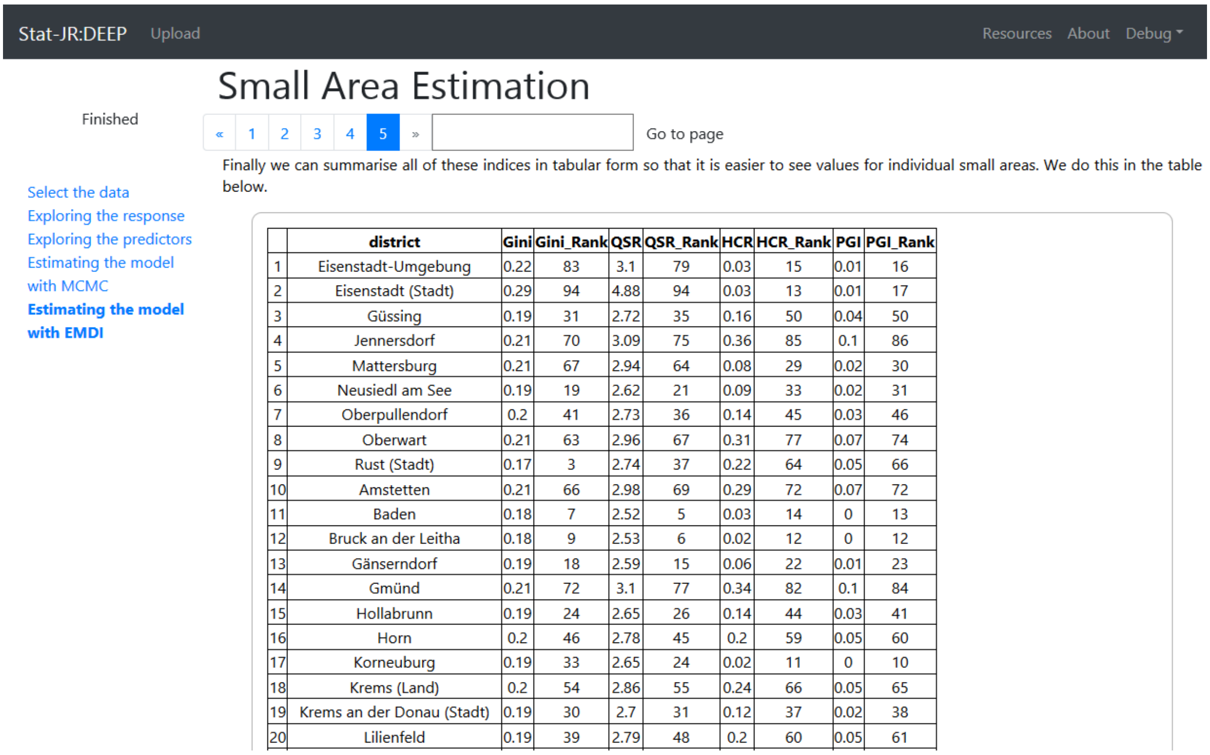


Small Area Estimation

	district	Sample_mean	Sample_sd	Population_mean	Population_sd
1	Eisenstadt-Umgebung	NA	NA	27509.02	11351.24
2	Eisenstadt (Stadt)	NA	NA	53654.42	30210.79
3	Güssing	NA	NA	17189.34	5853.47
4	Jennersdorf	NA	NA	13402.65	5037.61
5	Mattersburg	NA	NA	21260.09	8083.77
6	Neusiedl am See	19692.53	5641.59	19004.31	6386.01
7	Oberpullendorf	NA	NA	17671.04	6249.94
8	Oberwart	13833.36	7070.67	13954.69	5186.03
9	Rust (Stadt)	NA	NA	15457.77	5534.78
10	Amstetten	15201.15	6558.77	14282.55	5368.61
11	Baden	22921.69	6373.51	22373.54	7426.38
12	Bruck an der Leitha	23753.31	6277.61	24257.51	8130.46
13	Gänserndorf	20279.97	5940.92	20631.04	6992.67
14	Gmünd	NA	NA	13675.94	5098.5
15	Hollabrunn	17368.43	5563.52	17254.3	5805.11
16	Horn	NA	NA	16064.81	5877.73
17	Korneuburg	28920.37	10094.14	25629.5	9318.11
18	Krems (Land)	14928.84	5949.03	15065.72	5445.56
19	Krems an der Donau (Stadt)	NA	NA	18151.52	6239.09
20	Lilienfeld	NA	NA	16132.34	5565.92
21	Melk	12789.3	5201.32	11874.51	4809.86
22	Mistelbach	19718.95	6699.09	19455.02	6505.37

Here we see the emdi estimates compared to the sample (direct) estimates and observe that again the model-based and direct approaches do vary at times in order of 1000s. For example we see that the direct mean estimate for Korneuburg is 28920.37 whilst emdi estimates the mean income as 25,629.50. For further comparison the MCMC model-based estimate is 25,507.75 which is much closer to emdi than the direct estimate. There are subtle differences between the two approaches which may explain the estimate differences

As with the MCMC estimation there are many further equivalent outputs for emdi that can be interrogated so for comparison here is the equivalent table of inequality and poverty indices as we saw for MCMC.



Small Area Estimation

Finally we can summarise all of these indices in tabular form so that it is easier to see values for individual small areas. We do this in the table below.

	district	Gini	Gini_Rank	QSR	QSR_Rank	HCR	HCR_Rank	PGI	PGI_Rank
1	Eisenstadt-Umgebung	0.22	83	3.1	79	0.03	15	0.01	16
2	Eisenstadt (Stadt)	0.29	94	4.88	94	0.03	13	0.01	17
3	Güssing	0.19	31	2.72	35	0.16	50	0.04	50
4	Jennersdorf	0.21	70	3.09	75	0.36	85	0.1	86
5	Mattersburg	0.21	67	2.94	64	0.08	29	0.02	30
6	Neusiedl am See	0.19	19	2.62	21	0.09	33	0.02	31
7	Oberpullendorf	0.2	41	2.73	36	0.14	45	0.03	46
8	Oberwart	0.21	63	2.96	67	0.31	77	0.07	74
9	Rust (Stadt)	0.17	3	2.74	37	0.22	64	0.05	66
10	Amstetten	0.21	66	2.98	69	0.29	72	0.07	72
11	Baden	0.18	7	2.52	5	0.03	14	0	13
12	Bruck an der Leitha	0.18	9	2.53	6	0.02	12	0	12
13	Gänserndorf	0.19	18	2.59	15	0.06	22	0.01	23
14	Gmünd	0.21	72	3.1	77	0.34	82	0.1	84
15	Hollabrunn	0.19	24	2.65	26	0.14	44	0.03	41
16	Horn	0.2	46	2.78	45	0.2	59	0.05	60
17	Korneuburg	0.19	33	2.65	24	0.02	11	0	10
18	Krems (Land)	0.2	54	2.86	55	0.24	66	0.05	65
19	Krems an der Donau (Stadt)	0.19	30	2.7	31	0.12	37	0.02	38
20	Lilienfeld	0.19	39	2.79	48	0.2	60	0.05	61

Here the order has changed but we can see similar patterns for example the area Eisenstadt (Stadt) again has the highest Gini score and QSR of all districts with valid values though the values are slightly different with a Gini of 0.29 and a QSR of 4.88 compared with 0.29 and 4.70 respectively for MCMC.

We will finish this example here and move on to a third example that illustrates the use of Small area estimation in the situation where the variable of interest is binary.

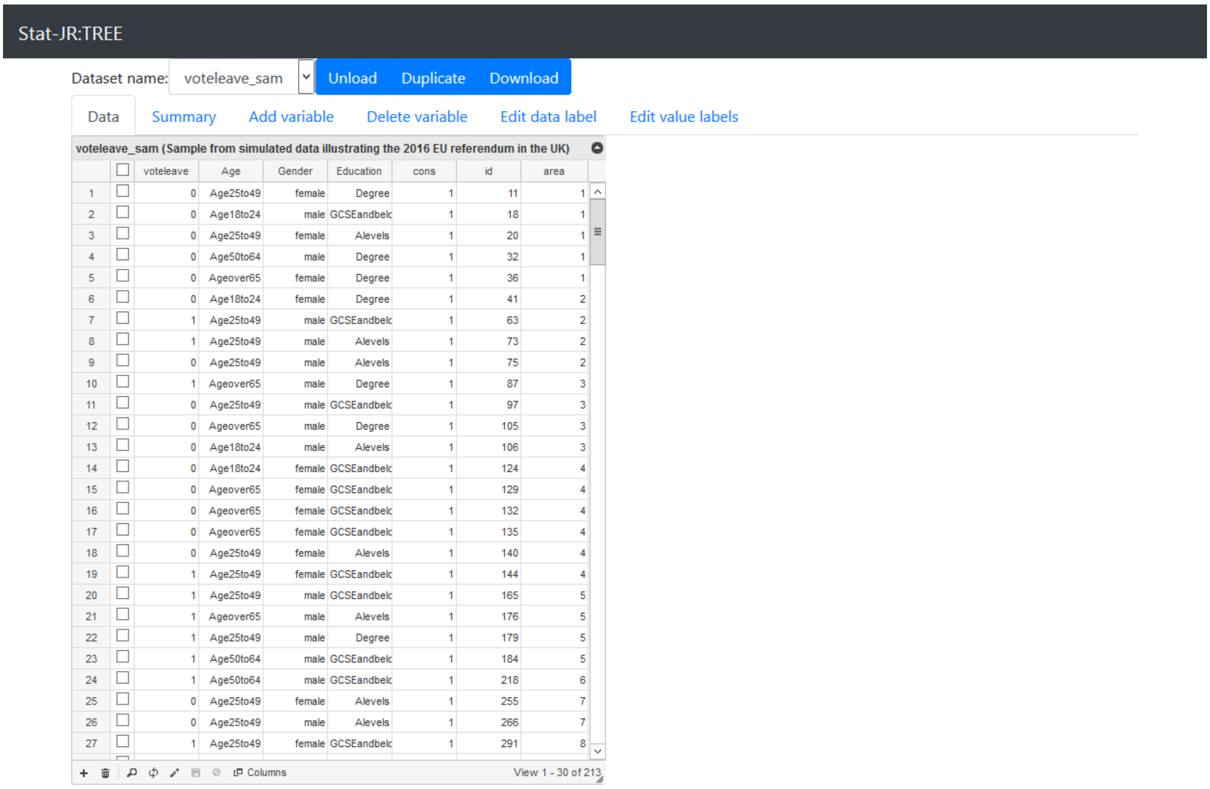
Example 3 – Voting Leave in the UK Brexit debate.

One of the most common binary or categorical variable uses of small area estimation is within politics when we consider voting intentions and try to predict the outcome of elections based on polls. Here the idea is that prior to an election polling companies canvass the voting intentions of a sample of the population and then use this information along with a model relating the voting intentions to demographic factors to predict how all of the voters not in the sample will vote.

We have for the purposes of this practical created a simulated (but realistic) dataset for voting intentions in the 2015 UK referendum on leaving the EU. Here the voting choice is a simple binary to leave or remain within the EU. For our simulated dataset we have a population dataset (*voteleave_pop.dta*) of 2,000 voters split into 50 equally populated (each with 40 voters) areas. From this we have taken a random sample (*voteleave_sam.dta*) of roughly 10% which in fact is 213 individuals with each area having between 1 and 7 voters in the sample. Note here it wouldn't matter if areas had no individuals in the sample (as we have seen for the tutorial example) as the small area estimation will still work in this scenario.

We will investigate this example using Stat-JR TREE rather than the eBook in DEEP as we can only use the MCMC algorithms for fitting these models as emdi only deals with Normal response models.

So if we first fire up Stat-JR TREE and **Choose** the dataset *voteleave_sam* and click **Use** then select to **View** the Dataset it should look as follows:



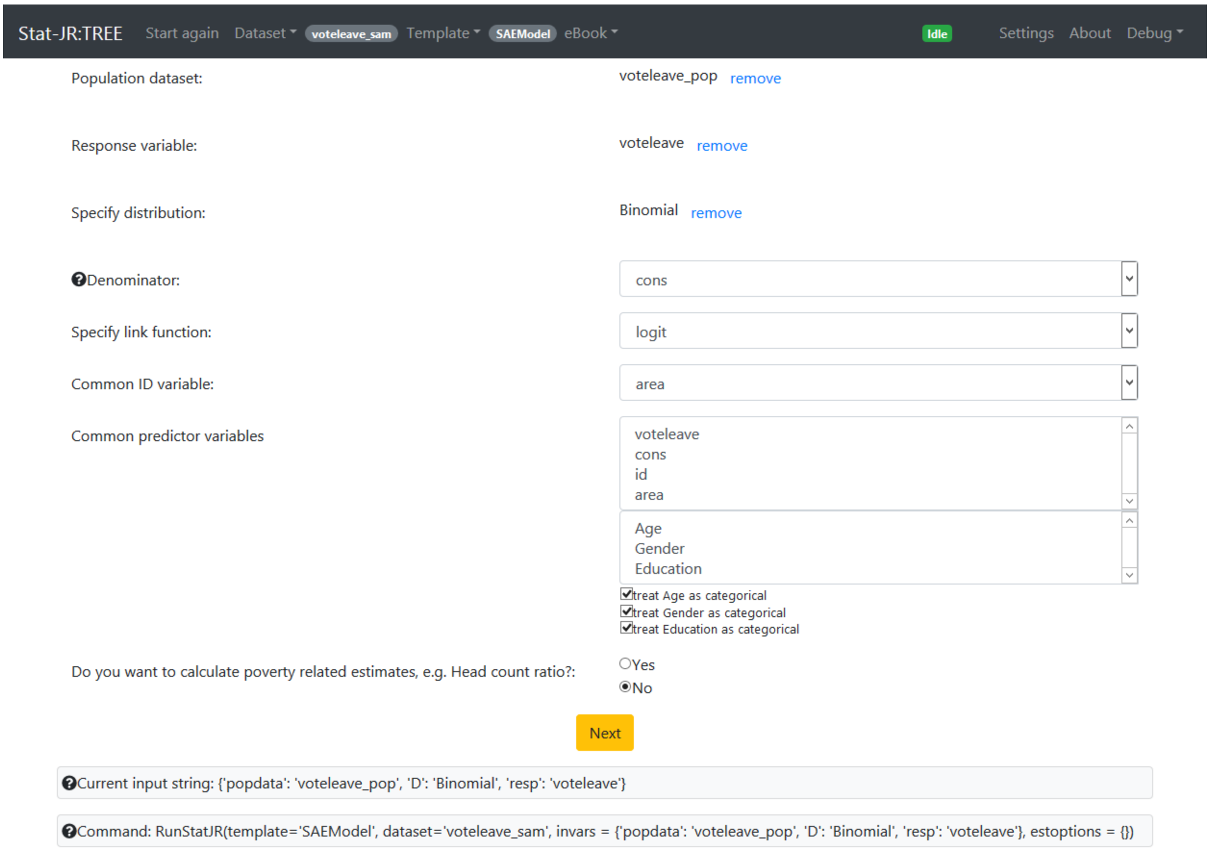
voteleave_sam (Sample from simulated data illustrating the 2016 EU referendum in the UK)

	□	voteleave	Age	Gender	Education	cons	id	area
1	□	0	Age25to49	female	Degree	1	11	1
2	□	0	Age18to24	male	GCSEandbelc	1	18	1
3	□	0	Age25to49	female	Alevels	1	20	1
4	□	0	Age50to64	male	Degree	1	32	1
5	□	0	Ageover65	female	Degree	1	36	1
6	□	0	Age18to24	female	Degree	1	41	2
7	□	1	Age25to49	male	GCSEandbelc	1	63	2
8	□	1	Age25to49	male	Alevels	1	73	2
9	□	0	Age25to49	male	Alevels	1	75	2
10	□	1	Ageover65	male	Degree	1	87	3
11	□	0	Age25to49	male	GCSEandbelc	1	97	3
12	□	0	Ageover65	male	Degree	1	105	3
13	□	0	Age18to24	male	Alevels	1	106	3
14	□	0	Age18to24	female	GCSEandbelc	1	124	4
15	□	0	Ageover65	female	GCSEandbelc	1	129	4
16	□	0	Ageover65	female	GCSEandbelc	1	132	4
17	□	0	Ageover65	female	GCSEandbelc	1	135	4
18	□	0	Age25to49	female	Alevels	1	140	4
19	□	1	Age25to49	female	GCSEandbelc	1	144	4
20	□	1	Age25to49	male	GCSEandbelc	1	165	5
21	□	1	Ageover65	male	Alevels	1	176	5
22	□	1	Age25to49	male	Degree	1	179	5
23	□	1	Age50to64	male	GCSEandbelc	1	184	5
24	□	1	Age50to64	male	GCSEandbelc	1	218	6
25	□	0	Age25to49	female	Alevels	1	255	7
26	□	0	Age25to49	male	Alevels	1	266	7
27	□	1	Age25to49	female	GCSEandbelc	1	291	8

View 1 - 30 of 213

Here we see the response variable *voteleave* which takes values 0 for voting remain and 1 for voting leave. There are then 3 categorical demographic variables that we will use to predict the probability of voting leave. Firstly we have *Age* which takes 4 categories, 0 for 25-49, 1 for 18-24, 2 for 50-64 and finally 3 for 65+ years old. We have used this unusual category order as we intend to use 25-49 as a base category capturing a large proportion of the population and expect to then see negative effects for the 18-24 category and positive effects for the older (50-64 and 65+) categories. Similarly we have *Education* with categories, 0 for *A levels* (which we will use as a baseline), 1 for *GCSE and lower qualifications* and 2 for *Degree and higher qualifications*. Again we expect the *GCSE and lower* group to have a positive effect and *Degree and higher qualifications* to have a negative effect. Finally gender is a binary with 0 for *male* and 1 for *female* with males slightly more likely to vote leave.

We will move straight to the **SAEModel** template and we set up the inputs as follows:



Population dataset: voteleave_pop remove

Response variable: voteleave remove

Specify distribution: Binomial remove

Denominator: cons

Specify link function: logit

Common ID variable: area

Common predictor variables
voteleave
cons
id
area

Age
Gender
Education

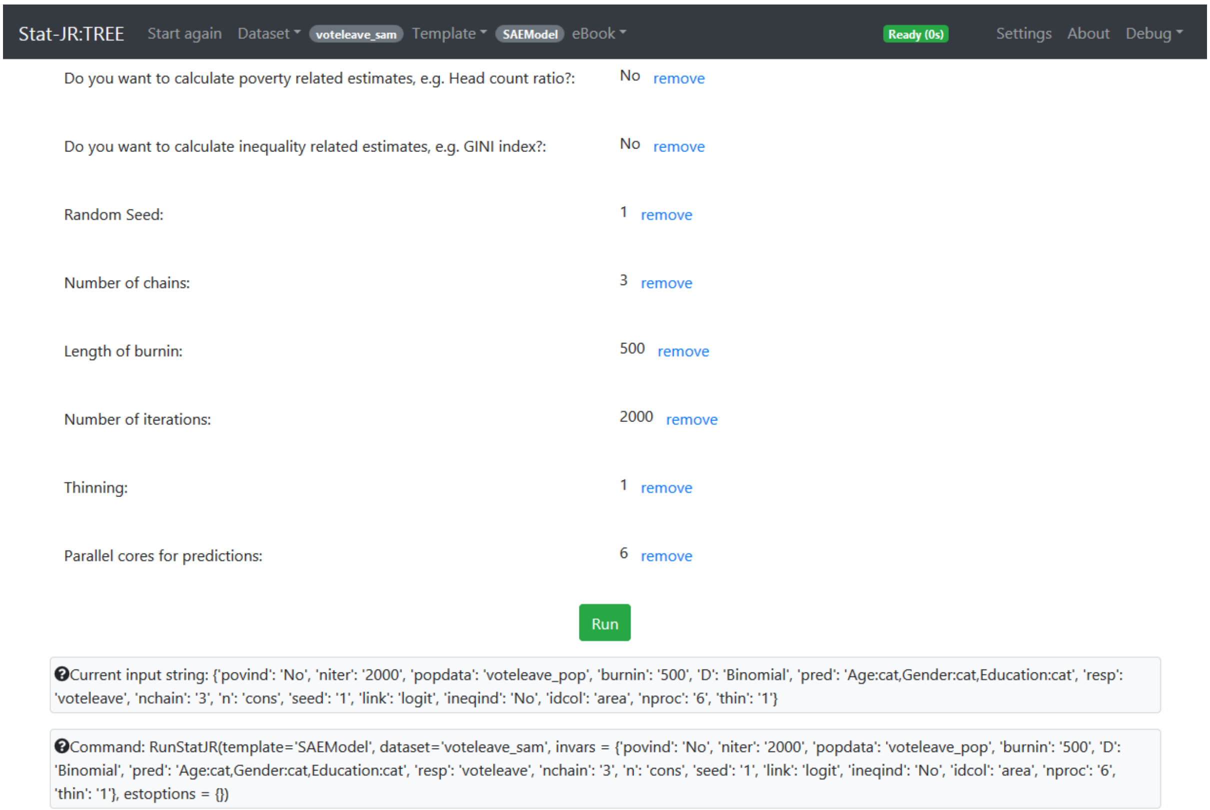
☑treat Age as categorical
☑treat Gender as categorical
☑treat Education as categorical

Do you want to calculate poverty related estimates, e.g. Head count ratio?: ○Yes ●No

Current input string: {'popdata': 'voteleave_pop', 'D': 'Binomial', 'resp': 'voteleave'}

Command: RunStatJR(template='SAEModel', dataset='voteleave_sam', invars = {'popdata': 'voteleave_pop', 'D': 'Binomial', 'resp': 'voteleave'}, estoptions = {})

Here you will see that when we choose Binomial for our model we have 2 additional inputs: *Denominator* for which we input *cons* which is a column of 1s and simply shows that our individual data points are individual people rather than groups and so the response is 0/1 rather than a proportion. We also specify a *logit link function* as we are planning to fit a (multilevel) logistic regression model. The final thing to note is that under the predictor variables we have ticked the *treat as categorical* options for all three predictors and here the first category will be treated as a base category (hence our unusual categorisation earlier) and the model to be fitted will have dummy variables for each other category. The poverty and inequality indexes don't make much sense for binary data so we will say *No* here. If we complete the inputs the screen will look as follows:



Do you want to calculate poverty related estimates, e.g. Head count ratio?: No remove

Do you want to calculate inequality related estimates, e.g. GINI index?: No remove

Random Seed: 1 remove

Number of chains: 3 remove

Length of burnin: 500 remove

Number of iterations: 2000 remove

Thinning: 1 remove

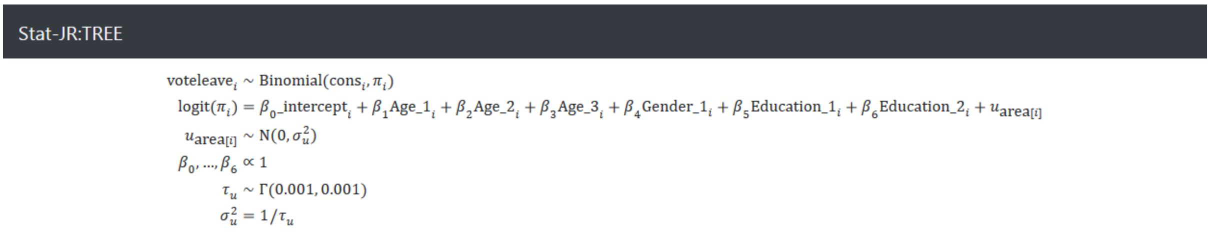
Parallel cores for predictions: 6 remove

Run

❓Current input string: {'povind': 'No', 'niter': '2000', 'popdata': 'voteleave_pop', 'burnin': '500', 'D': 'Binomial', 'pred': 'Age:cat,Gender:cat,Education:cat', 'resp': 'voteleave', 'nchain': '3', 'n': 'cons', 'seed': '1', 'link': 'logit', 'ineqind': 'No', 'idcol': 'area', 'nproc': '6', 'thin': '1'}

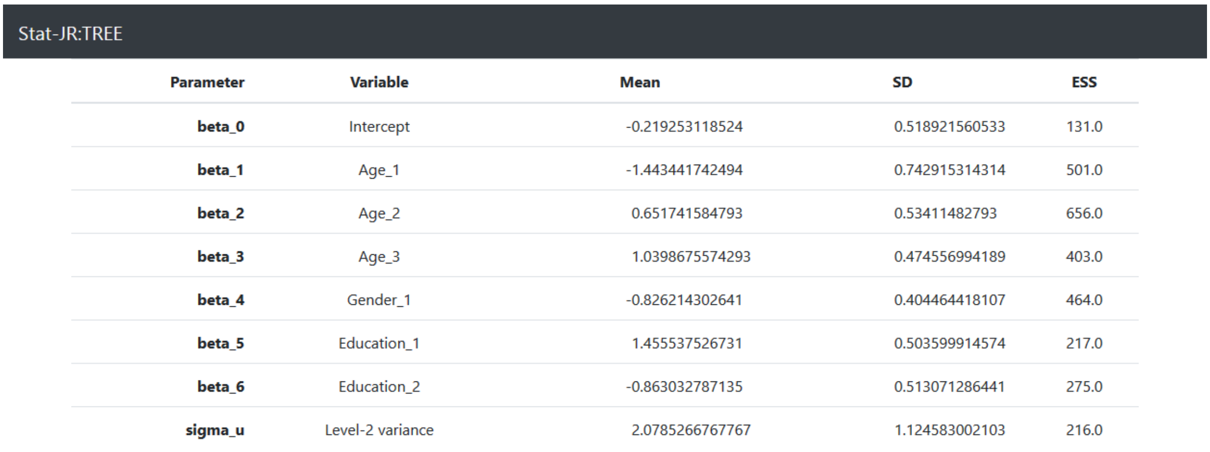
❓Command: RunStatJR(template='SAEModel', dataset='voteleave_sam', invars = {'povind': 'No', 'niter': '2000', 'popdata': 'voteleave_pop', 'burnin': '500', 'D': 'Binomial', 'pred': 'Age:cat,Gender:cat,Education:cat', 'resp': 'voteleave', 'nchain': '3', 'n': 'cons', 'seed': '1', 'link': 'logit', 'ineqind': 'No', 'idcol': 'area', 'nproc': '6', 'thin': '1'}, estoptions = {})

If we click on **Run** we can now wait for the model to fit and the rest of the SAE outputs to be generated. When this happens the timer will say **Ready** in the top right and if we look at *equation.tex* from the object pull down list and perhaps pop it out we will see the following:



$$\begin{aligned}
\mathrm{voteleave}_i &\sim \mathrm{Binomial}(\mathrm{cons}_i, \pi_i)\\
\mathrm{logit}(\pi_i) &= \beta_0\mathrm{intercept}_i + \beta_1\mathrm{Age\_1}_i + \beta_2\mathrm{Age\_2}_i + \beta_3\mathrm{Age\_3}_i + \beta_4\mathrm{Gender\_1}_i + \beta_5\mathrm{Education\_1}_i + \beta_6\mathrm{Education\_2}_i + u_{\mathrm{area}[i]}\\
u_{\mathrm{area}[i]} &\sim \mathrm{N}(0, \sigma^2_u)\\
\beta_0, \ldots, \beta_6 &\propto 1\\
\tau_u &\sim \Gamma(0.001, 0.001)\\
\sigma^2_u &= 1/\tau_u
\end{aligned}$$

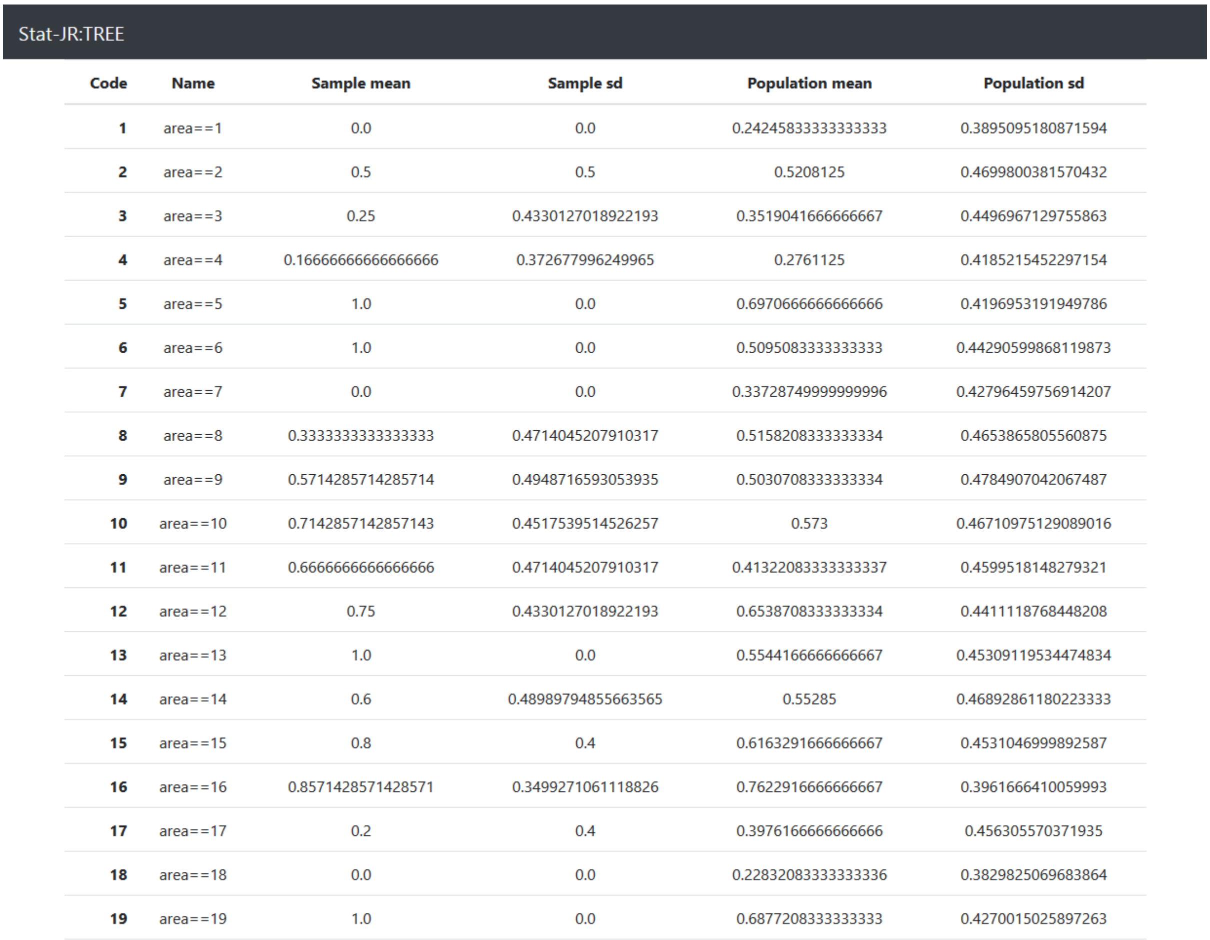
Here we see a logistic regression with an intercept and 6 predictors made up of the categories of the 3 categorical predictors along with random effects for the different areas. Looking further down the list of objects we can pop out the *sae_modelresults* object which looks as follows:



Parameter	Variable	Mean	SD	ESS
beta_0	Intercept	-0.219253118524	0.518921560533	131.0
beta_1	Age_1	-1.443441742494	0.742915314314	501.0
beta_2	Age_2	0.651741584793	0.53411482793	656.0
beta_3	Age_3	1.0398675574293	0.474556994189	403.0
beta_4	Gender_1	-0.826214302641	0.404464418107	464.0
beta_5	Education_1	1.455537526731	0.503599914574	217.0
beta_6	Education_2	-0.863032787135	0.513071286441	275.0
sigma_u	Level-2 variance	2.0785266767767	1.124583002103	216.0

Here we see as predicted the younger age group has a negative coefficient (-1.443) whilst the older age gropus have positive coefficients (0.652 and 1.040 respectively) although in terms of statistical significance only the *beta_3* term looks significant perhaps due to the rather small sample size of 213. We observe the predicted direction of effects also for education and gender and we see a very large (2.079) between area variance suggesting there are large differences in how areas voted not explained simply by the demographics of the indiivudals. Ideally we would like area level predictors to explain these differences but here we do not have such information.

Looking at the SAE objects we can first look at *prediction_summary* which looks as follows:

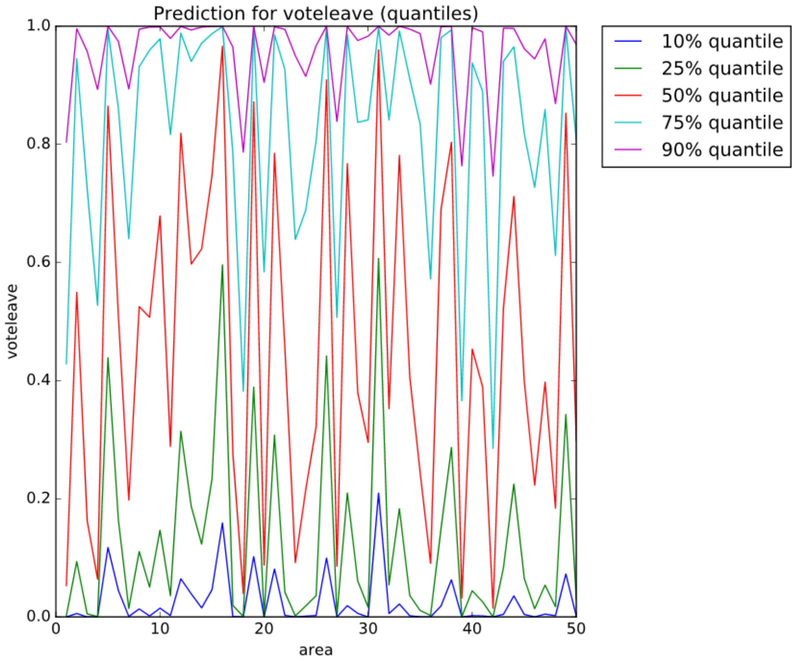


Stat-JR:TREE

Code	Name	Sample mean	Sample sd	Population mean	Population sd
1	area==1	0.0	0.0	0.24245833333333333	0.3895095180871594
2	area==2	0.5	0.5	0.5208125	0.4699800381570432
3	area==3	0.25	0.4330127018922193	0.3519041666666667	0.4496967129755863
4	area==4	0.16666666666666666	0.372677996249965	0.2761125	0.4185215452297154
5	area==5	1.0	0.0	0.6970666666666666	0.4196953191949786
6	area==6	1.0	0.0	0.5095083333333333	0.44290599868119873
7	area==7	0.0	0.0	0.33728749999999996	0.42796459756914207
8	area==8	0.3333333333333333	0.4714045207910317	0.5158208333333334	0.4653865805560875
9	area==9	0.5714285714285714	0.4948716593053935	0.5030708333333334	0.4784907042067487
10	area==10	0.7142857142857143	0.4517539514526257	0.573	0.46710975129089016
11	area==11	0.6666666666666666	0.4714045207910317	0.41322083333333337	0.4599518148279321
12	area==12	0.75	0.4330127018922193	0.6538708333333334	0.4411118768448208
13	area==13	1.0	0.0	0.5544166666666667	0.45309119534474834
14	area==14	0.6	0.48989794855663565	0.55285	0.46892861180223333
15	area==15	0.8	0.4	0.6163291666666667	0.4531046999892587
16	area==16	0.8571428571428571	0.3499271061118826	0.7622916666666667	0.3961666410059993
17	area==17	0.2	0.4	0.3976166666666666	0.456305570371935
18	area==18	0.0	0.0	0.22832083333333336	0.3829825069683864
19	area==19	1.0	0.0	0.6877208333333333	0.4270015025897263

Here we get to the left the estimates simply from the sample dataset – so for example for the first area we get an estimate of 0 i.e. meaning everyone votes remain which is because the 5 people in the sample voted remain. We see that the population estimate which takes into account demographics suggests that in fact we might expect 24.2% of the population voting leave. In fact we are in the fortunate position of having the actual voting decisions of all people in the population in the population dataset and we can use this to see that in fact 22.5% voted leave so in this case the estimate from the modelling is much better than the sample estimate. Similarly for area 2 we get probability of voting leave of 0.5 from the sample (2 out of 4) with a higher model estimate of 0.521 (and in fact the truth was 0.675).

We can look at other outputs like the quantiles which looks as follows:



What we see here is in fact we have quite wide quantiles which are a little hard to interpret as the underlying data is 0 and 1! It is probably easiest to think of these in terms of underlying probabilities so for example for the median the first area appears to have an estimate below 0.1 and so we would expect that the median person in the area here would most likely vote remain with only a small probability of them voting leave.

We will leave this example now and in fact this ends this introductory practical on using Stat-JR for small area estimation.