Modelling Longitudinal Data using the Stat-JR package

7 July 2014

Workshop Practical Exercises

Modelling Longitudinal Data using the Stat-JR package

7 July 2014

Practicals: Contents

MODELLING LONGITUDINAL DATA USING THE STAT-JR PACKAGE

Practical 1: Introduction to Growth Curve Modelling and the Stat-JR package

Aims of Practical

In this practical we will learn how to fit simple multilevel models in Stat-JR, firstly through interoperability with MLwiN and then using the e-STAT MCMC estimation engine. We will introduce the main dataset that we will use for all practicals that follow and show how to restructure longitudinal datasets from wide to long format. We will cover both random intercept and random slopes models for longitudinal datasets.

Getting Started with Stat-JR and the TREE interface

The Stat-JR package is written in Python and has several application interfaces that run in Python with a command prompt window in the background. The TREE (Template Reading and Execution Environment) interface is a flexible environment allowing the user to interact with Stat-JR via a web browser and try out any Stat-JR templates in combination with user datasets. We will use this interface throughout the workshop but it is worth noting that Stat-JR also has a DEEP (Documents with Embedded Execution and Provenance) interface which embeds Stat-JR's functionality within electronic books, and also a separate command line Python interface.

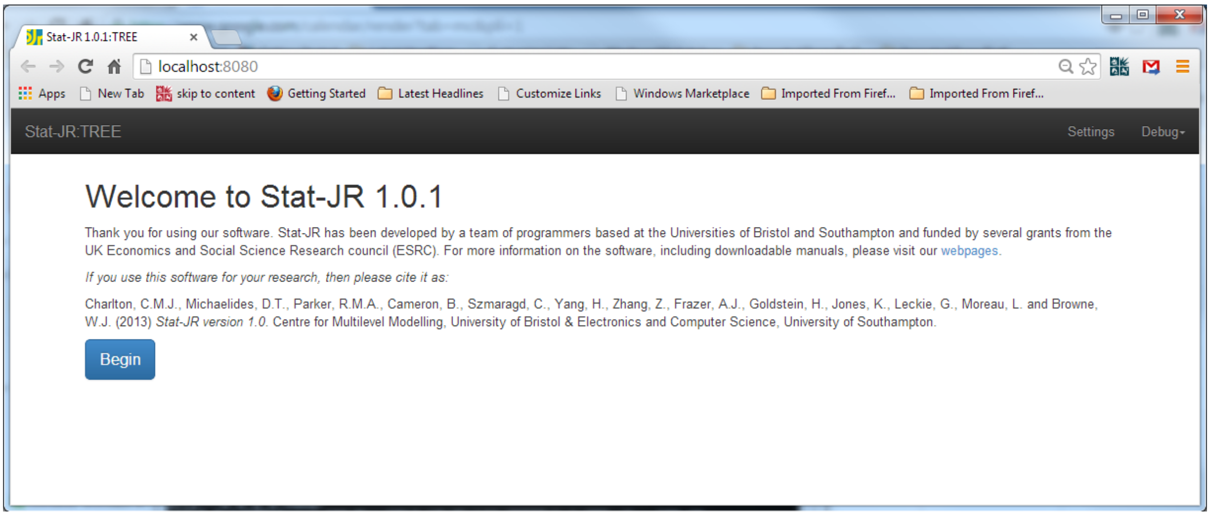
To start up the TREE interface, double-click *tree.cmd* in the base directory of the Stat-JR install (on your memory stick); this will bring up a command window in which a list of commands will appear similar to the following:

```
E:\newstruct\Software\StatJRrep\estat\trunk>SET Path=E:\newstruct\Software\StatJ
Rrep\estat\trunk\MinGW\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb
em;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files (x86)\QuickTime\
QTSystem\;C:\Program Files\TortoiseSVN\bin;C:\Program Files\MiKTeX 2.9\miktex\bi
n\x64\
E:\newstruct\Software\StatJRrep\estat\trunk>SET LTDL_LIBRARY_PATH=E:\newstruct\S
oftware\StatJRrep\estat\trunk\JAGS-3.3.0\i386\modules

E:\newstruct\Software\StatJRrep\estat\trunk>cd src\apps\webtest

E:\newstruct\Software\StatJRrep\estat\trunk\src\apps\webtest>..\..\..\App\Python
.exe webtest.py 8080
WARNING:root:Failed to load package GenStat_model (GenStat not found)
WARNING:root:Failed to load package gretl_model (Gretl not found)
WARNING:root:Failed to load package MATLAB_script (Matlab not found)
WARNING:root:Failed to load package Minitab_model (Minitab not found)
WARNING:root:Failed to load package Minitab_script (Minitab not found)
WARNING:root:Failed to load package MIXREGLS (MIXREGLS not found)
WARNING:root:Failed to load package Octave_script (Octave not found)
WARNING:root:Failed to load package SABRE (Sabre not found)
WARNING:root:Failed to load package SAS_model (SAS not found)
WARNING:root:Failed to load package SAS_script (SAS not found)
WARNING:root:Failed to load package SPSS_model (SPSS not found)
WARNING:root:Failed to load package SPSS_script (SPSS not found)
WARNING:root:Failed to load package Stata_MLwiN (Stata not found)
WARNING:root:Failed to load package Stata_model (Stata not found)
WARNING:root:Failed to load package Stata_script (Stata not found)
WARNING:root:Failed to load package SuperMix (SuperMIX not found)
INFO:root:Trying to locate and open default web browser
http://0.0.0.0:8080/
```

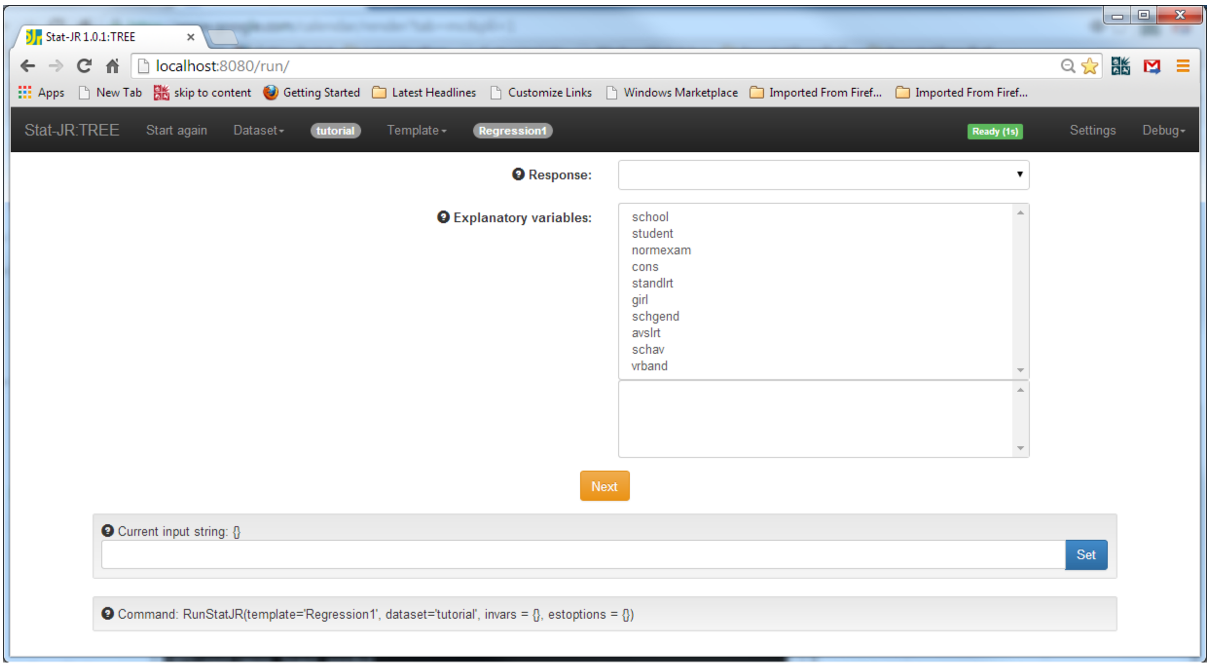
The last line indicates that a web process is starting; Stat-JR uses a web browser as an input/output device however the computation will be done on your own computer. If you haven't got a web browser already open, the default web browser will open and look as follows:



Two important things to note:

- The number 8080 (in this example) will vary each time you run the software to allow several versions of Stat-JR to run at once.
- Stat-JR works best with either Chrome or Firefox, so if the default browser on your machine is Internet Explorer it is best to open a different browser and copy the html path to it.

Clicking on the **Begin** button will then bring up the main screen for Stat-JR



- This window shows the **Current template** and **Current dataset** at the top of the screen along with pull down menus from which one can select different templates and datasets.

- Underneath you will see the first inputs for the currently selected template.

- The **Current input string** and **Command** boxes will contain information about the inputs being used and will be populated as the user chooses their inputs. One can paste in a string of inputs into the **Current input string** box and click **Set** as an alternative to filling in the inputs manually and the **Command** box can be used to store the command that contain the current inputs that can then be used in a Command line version of Stat-JR.

Introduction to the reading development dataset

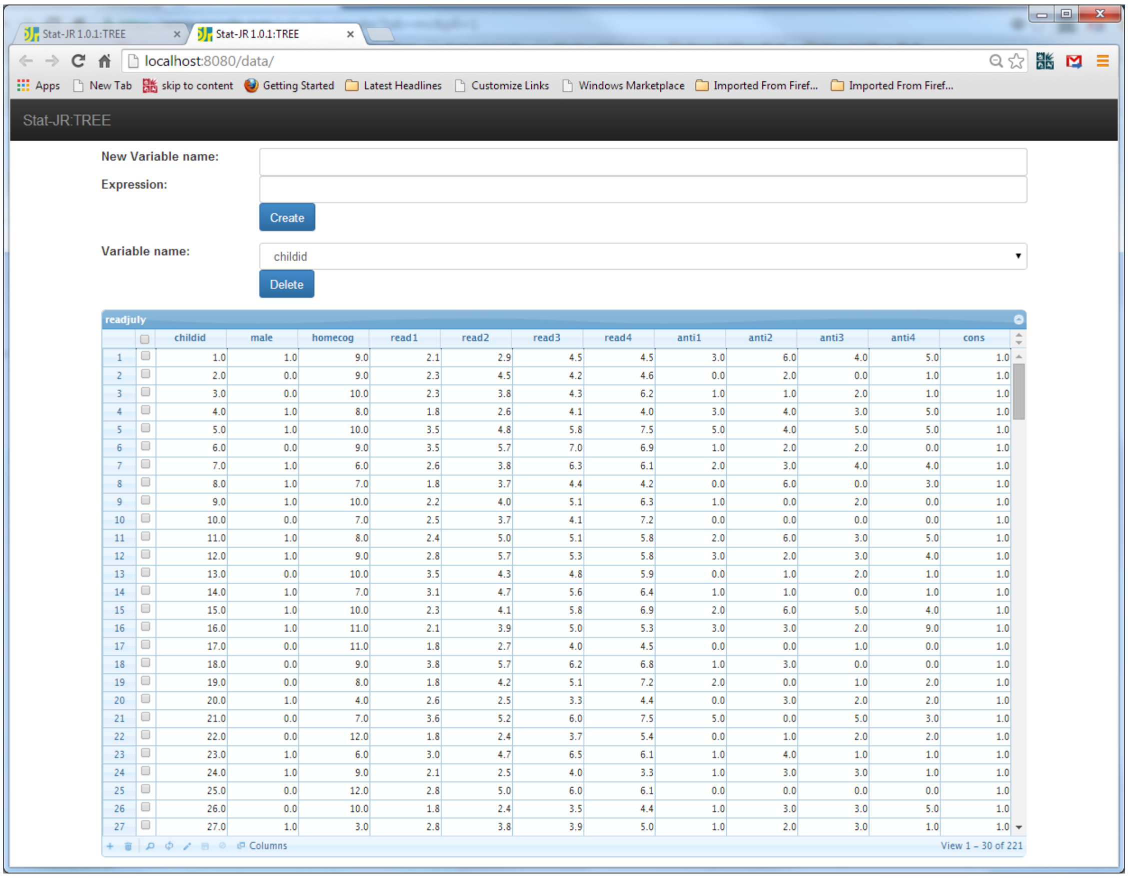
In this exercise, we will analyse a subsample of data from the National Longitudinal Survey of Youth (NLSY) of Labor Market Experience in Youth. Starting in 1986, children of the female respondents of the original NLSY Youth sample were assessed in 1986, and again in 1988, 1990 and 1992. For inclusion in the subsample considered here, children had to be between 6 and 8 years old at the first measurement. (For more details of the dataset and measures see http://www.unc.edu/~curran/srcd-docs/srcdmeth.pdf). We also restrict the analysis to 221 children who were assessed on all four occasions. The data file is called *readjuly* (saved as *readjuly.dta* in the datasets subdirectory of the Stat-JR install).

The file contains the following variables:

CHILDID	Child identifier (coded 1 to 221)
MALE	Child's gender (1=male, 0=female)
HOMECOG	Amount of cognitive support at home, computed as the sum of 14 binary items, e.g. *Does you family get a daily newspaper? How often do you read stories to your child?* (score from 3 to 14)
READ1	Reading score at time 1 (1986, when child aged 6-8 years), measuring word recognition and pronunciation ability
READ2	Reading score at time 2 (1988)
READ3	Reading score at time 3 (1990)
READ4	Reading score at time 4 (1992)
ANTI1	Antisocial behaviour score at time 1 (1986), based on mother's report on six items e.g. cheats or tells lies, bullies, disobedient at school (score from 0-10)
ANTI2	Antisocial behaviour score at time 2 (1988)
ANTI3	Antisocial behaviour score at time 3 (1990)
ANTI4	Antisocial behaviour score at time 4 (1992)
CONS	A column of 1s to represent the intercept

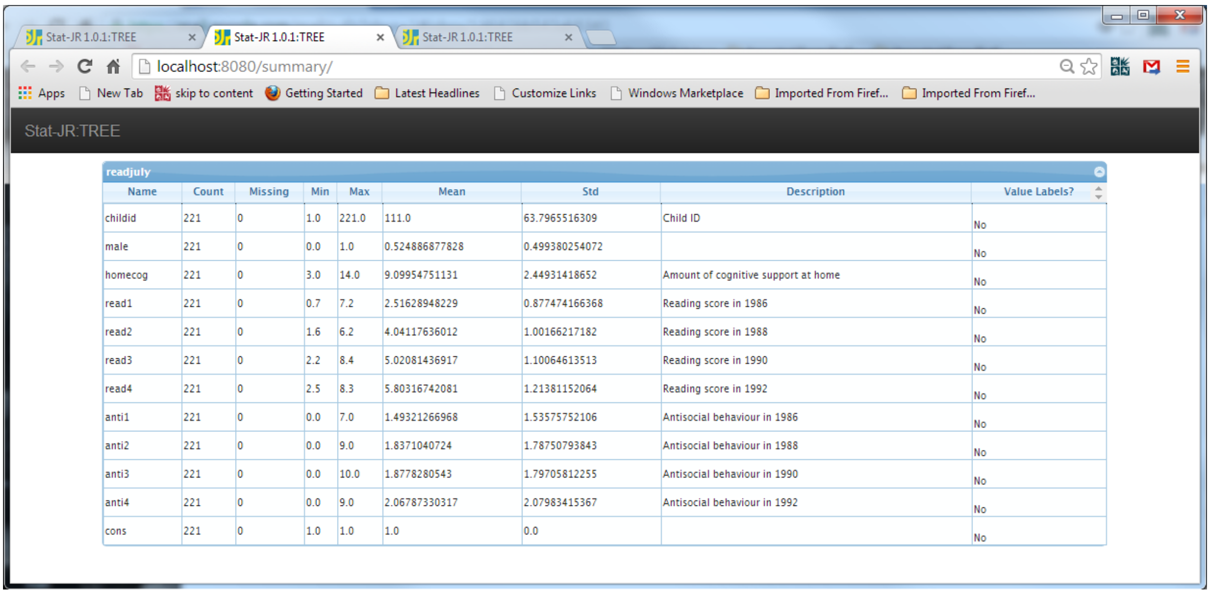
To select the dataset *readjuly*, click on the **Dataset** pull down list and select **Choose.** From the dataset list scroll down until you find *readjuly*, highlight it, and click on the **Use** button to the bottom right of the window.

After pressing **Use,** the **Current dataset** will change at the top of the window to confirm your selection, and we can select **View** from the **Dataset** pull down list which will bring up a separate tab at the top of the screen with the first data records in the dataset as shown overleaf:



	childid	male	homecog	read1	read2	read3	read4	anti1	anti2	anti3	anti4	cons
1	1.0	1.0	9.0	2.1	2.9	4.5	4.5	3.0	6.0	4.0	5.0	1.0
2	2.0	0.0	9.0	2.3	4.5	4.2	4.6	0.0	2.0	0.0	1.0	1.0
3	3.0	0.0	10.0	2.3	3.8	4.3	6.2	1.0	1.0	2.0	1.0	1.0
4	4.0	1.0	8.0	1.8	2.6	4.1	4.0	3.0	4.0	3.0	5.0	1.0
5	5.0	1.0	10.0	3.5	4.8	5.8	7.5	5.0	4.0	5.0	5.0	1.0
6	6.0	0.0	9.0	3.5	5.7	7.0	6.9	1.0	2.0	2.0	0.0	1.0
7	7.0	1.0	6.0	2.6	3.8	6.3	6.1	2.0	3.0	4.0	4.0	1.0
8	8.0	1.0	7.0	1.8	3.7	4.4	4.2	0.0	6.0	0.0	3.0	1.0
9	9.0	1.0	10.0	2.2	4.0	5.1	6.3	1.0	0.0	2.0	0.0	1.0
10	10.0	0.0	7.0	2.5	3.7	4.1	7.2	0.0	0.0	0.0	0.0	1.0
11	11.0	1.0	8.0	2.4	5.0	5.1	5.8	2.0	6.0	3.0	5.0	1.0
12	12.0	1.0	9.0	2.8	5.7	5.3	5.8	3.0	2.0	3.0	4.0	1.0
13	13.0	0.0	10.0	3.5	4.3	4.8	5.9	0.0	1.0	2.0	1.0	1.0
14	14.0	1.0	7.0	3.1	4.7	5.6	6.4	1.0	1.0	0.0	1.0	1.0
15	15.0	1.0	10.0	2.3	4.1	5.8	6.9	2.0	6.0	5.0	4.0	1.0
16	16.0	1.0	11.0	2.1	3.9	5.0	5.3	3.0	3.0	2.0	9.0	1.0
17	17.0	0.0	11.0	1.8	2.7	4.0	4.5	0.0	0.0	1.0	0.0	1.0
18	18.0	0.0	9.0	3.8	5.7	6.2	6.8	1.0	3.0	0.0	0.0	1.0
19	19.0	0.0	8.0	1.8	4.2	5.1	7.2	2.0	0.0	1.0	2.0	1.0
20	20.0	1.0	4.0	2.6	2.5	3.3	4.4	0.0	3.0	2.0	2.0	1.0
21	21.0	0.0	7.0	3.6	5.2	6.0	7.5	5.0	0.0	5.0	3.0	1.0
22	22.0	0.0	12.0	1.8	2.4	3.7	5.4	0.0	1.0	2.0	2.0	1.0
23	23.0	1.0	6.0	3.0	4.7	6.5	6.1	1.0	4.0	1.0	1.0	1.0
24	24.0	1.0	9.0	2.1	2.5	4.0	3.3	1.0	3.0	3.0	1.0	1.0
25	25.0	0.0	12.0	2.8	5.0	6.0	6.1	0.0	0.0	0.0	0.0	1.0
26	26.0	0.0	10.0	1.8	2.4	3.5	4.4	1.0	3.0	3.0	5.0	1.0
27	27.0	1.0	3.0	2.8	3.8	3.9	5.0	1.0	2.0	3.0	1.0	1.0

The above data structure, with one record per child and measurements at each time point stored as separate variables, is commonly referred to as **wide form**. We can get some basic summary information for each column selecting **Summary** from the Dataset pull down**;** this produces the following in a new tab:

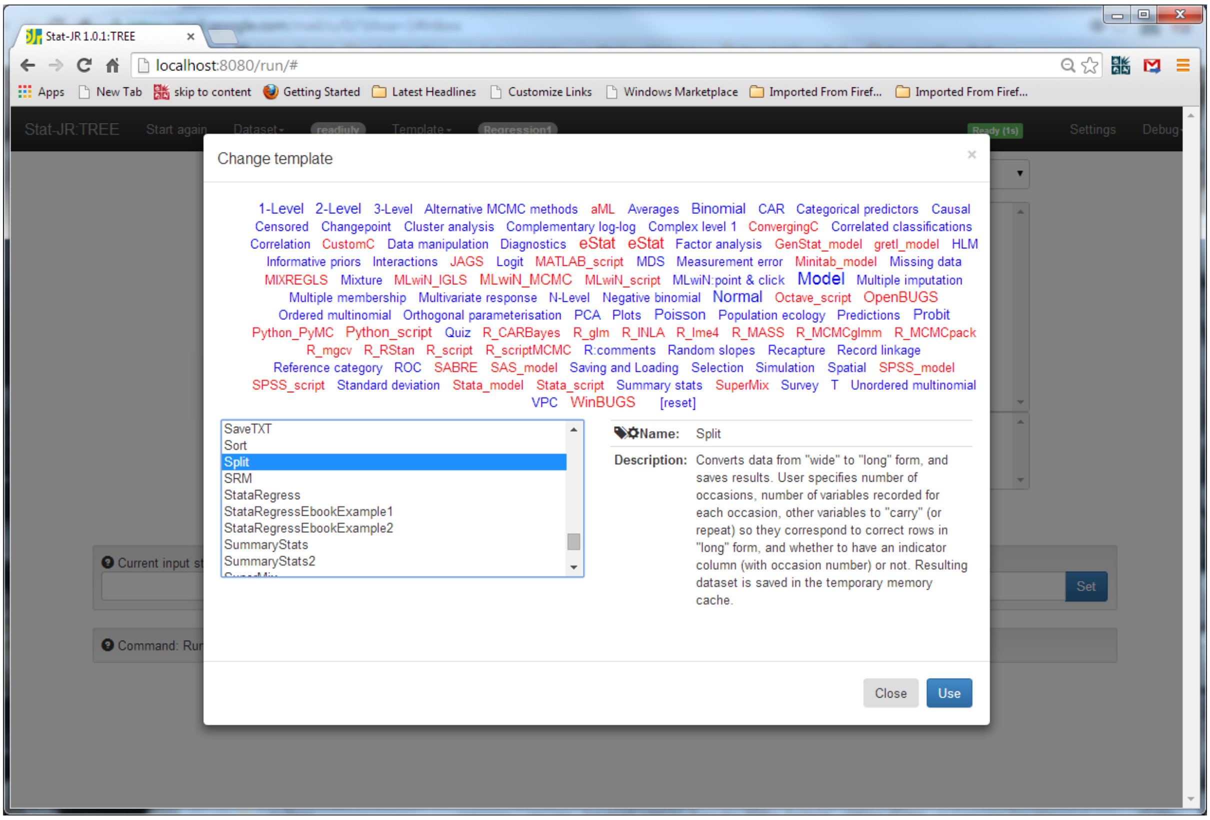


Name	Count	Missing	Min	Max	Mean	Std	Description	Value Labels?
childid	221	0	1.0	221.0	111.0	63.7965516309	Child ID	No
male	221	0	0.0	1.0	0.524886877828	0.499380254072		No
homecog	221	0	3.0	14.0	9.09954751131	2.44931418652	Amount of cognitive support at home	No
read1	221	0	0.7	7.2	2.51628948229	0.877474166368	Reading score in 1986	No
read2	221	0	1.6	6.2	4.04117636012	1.00166217182	Reading score in 1988	No
read3	221	0	2.2	8.4	5.02081436917	1.10064613513	Reading score in 1990	No
read4	221	0	2.5	8.3	5.80316742081	1.21381152064	Reading score in 1992	No
anti1	221	0	0.0	7.0	1.49321266968	1.53575752106	Antisocial behaviour in 1986	No
anti2	221	0	0.0	9.0	1.8371040724	1.78750793843	Antisocial behaviour in 1988	No
anti3	221	0	0.0	10.0	1.8778280543	1.79705812255	Antisocial behaviour in 1990	No
anti4	221	0	0.0	9.0	2.06787330317	2.07983415367	Antisocial behaviour in 1992	No
cons	221	0	1.0	1.0	1.0	0.0		No

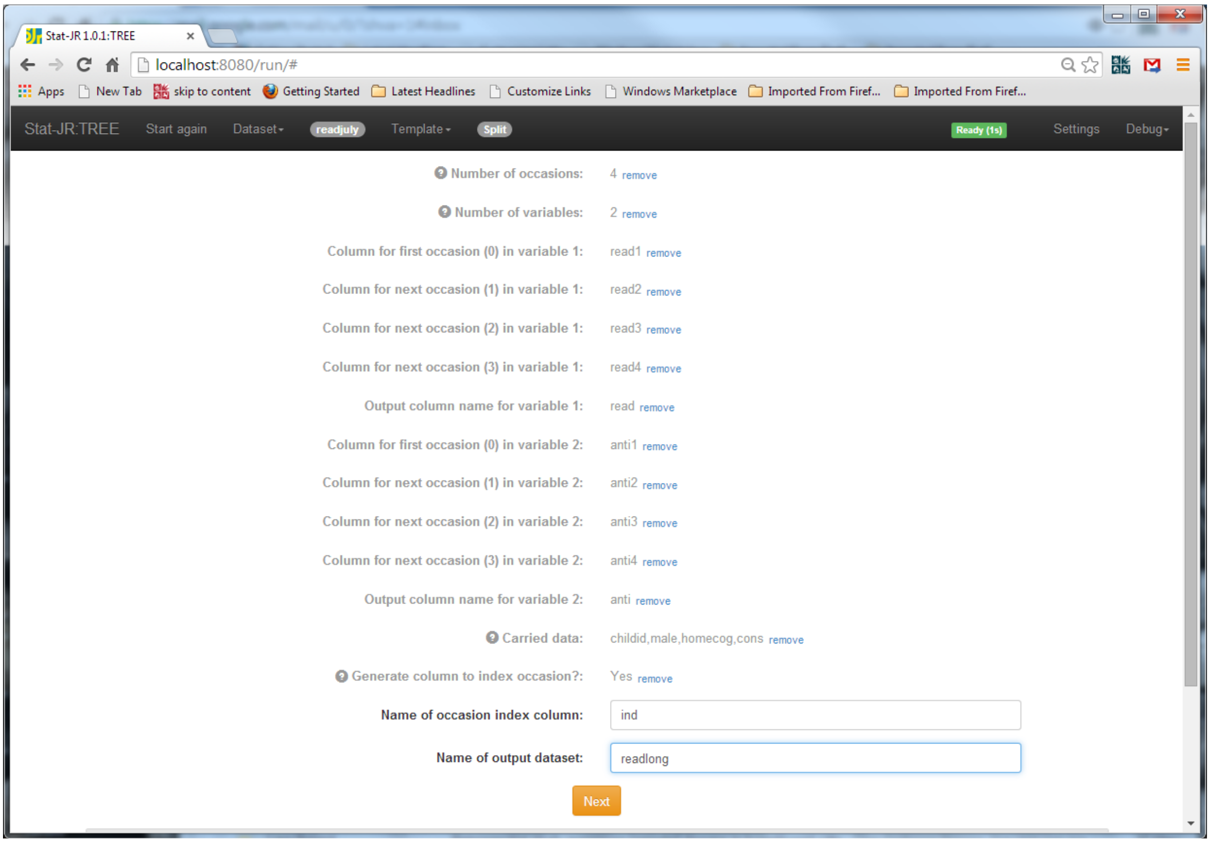
We see that, as expected, the mean reading score increases with time (or age).

Restructuring the data from wide to long form, and plotting observed trajectories

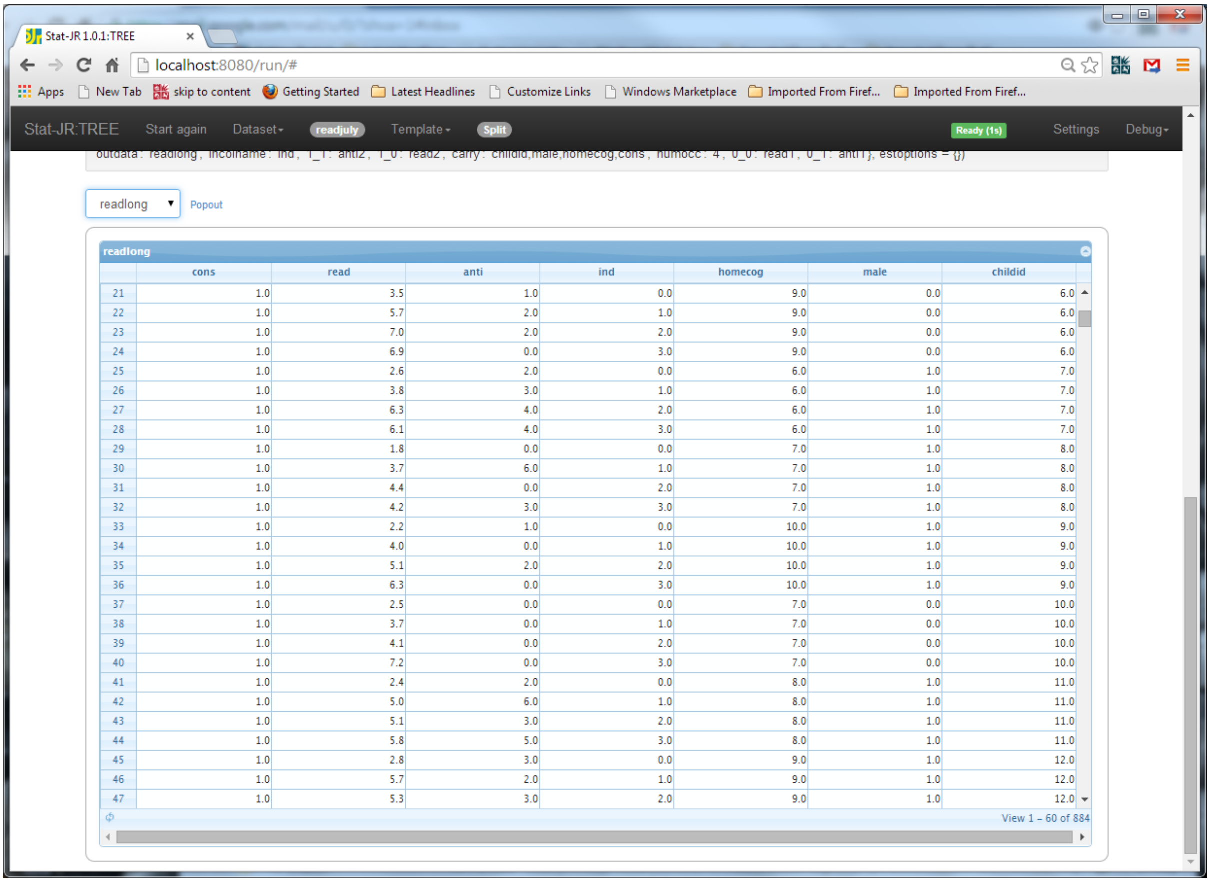
Most methods for longitudinal data analysis require data to be in long form, with repeated measures stacked into a single variable to give 1 record per year for each individual. Restructuring from wide to long form is possible in Stat-JR via the *Split* template (alternatively the *UnSplit* template is used to restructure datasets from long to wide form). Therefore, from the main Stat-JR screen, we need to use the **Template** pull down list at the top of the screen and select **Choose**. From the screen that appears scroll down through the template list and select *Split* (alternatively we could have clicked on *Data manipulation* in the tag cloud to reduce the list of templates to just those concerning data manipulation, as shown in the screenshot below). Note that when *Split* is highlighted we get a description of it as well.



Clicking on **Use** will change the **Current template**, listed at the top, to *Split* and then we can execute the template. We then need to fill in the inputs as follows:



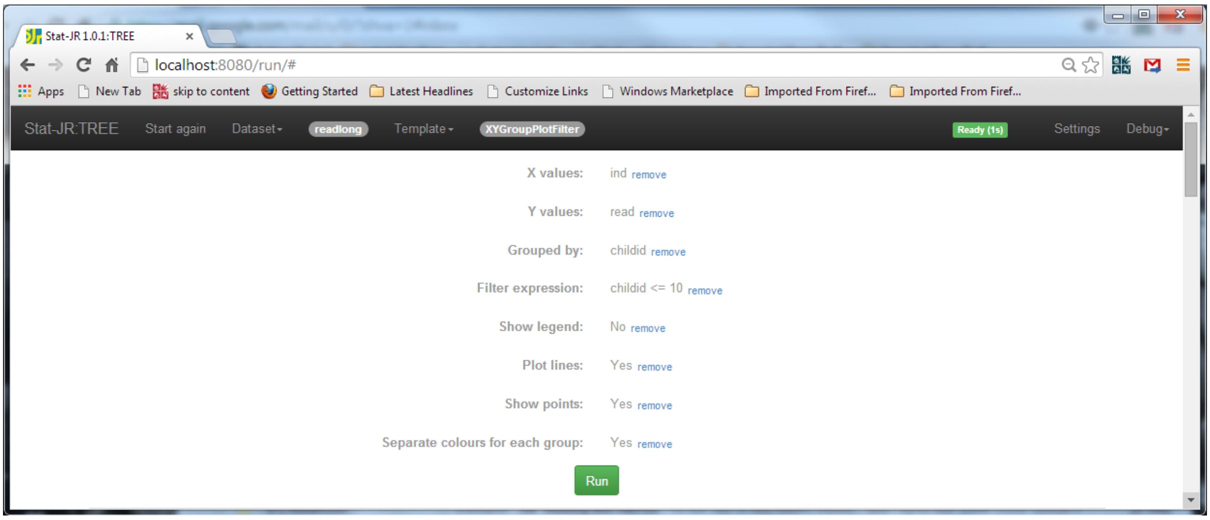
Then clicking on **Next** and **Run** we can display the generated dataset (*readlong*) in the output objects pane at the bottom of the screen by selecting it from the pull down list as shown below:



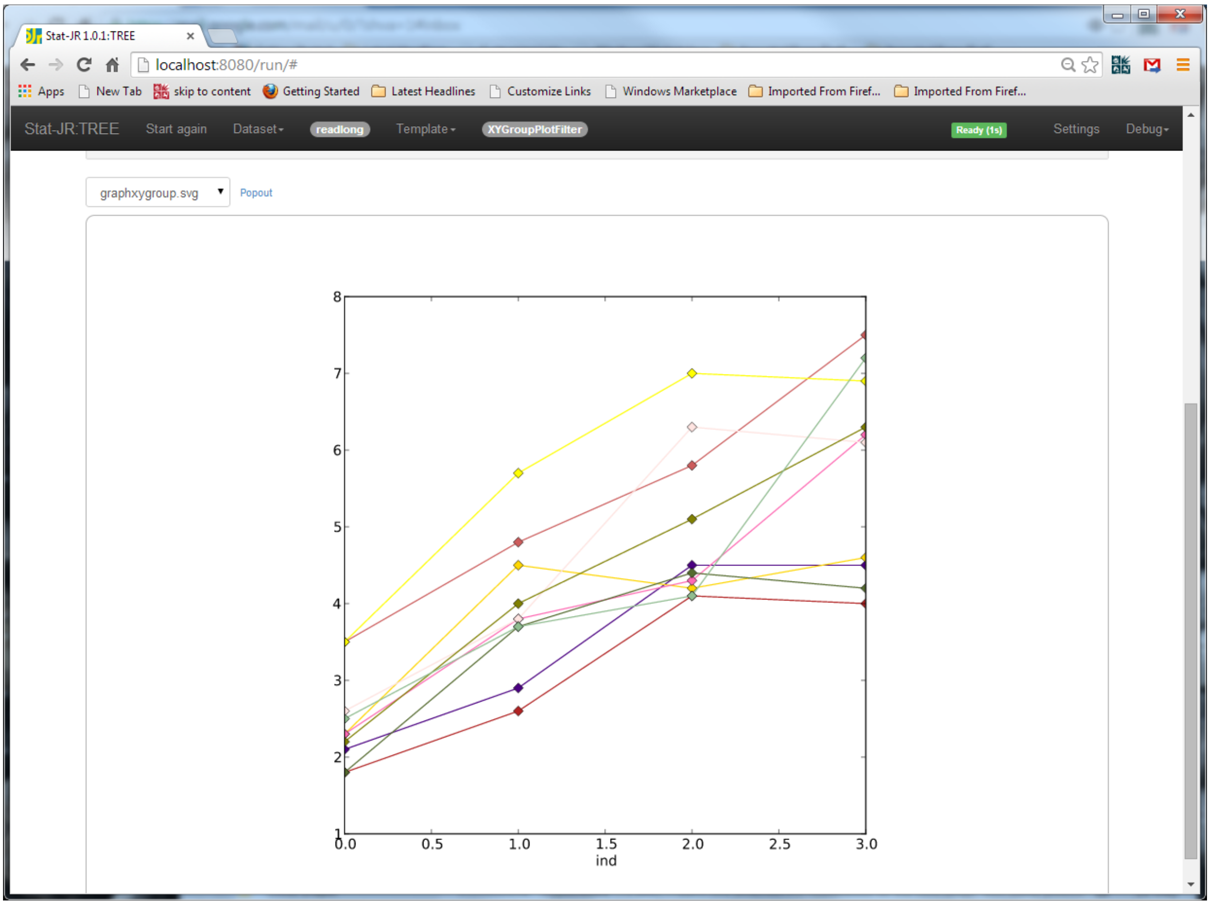
	cons	read	anti	ind	homecog	male	childid
21	1.0	3.5	1.0	0.0	9.0	0.0	6.0
22	1.0	5.7	2.0	1.0	9.0	0.0	6.0
23	1.0	7.0	2.0	2.0	9.0	0.0	6.0
24	1.0	6.9	0.0	3.0	9.0	0.0	6.0
25	1.0	2.6	2.0	0.0	6.0	1.0	7.0
26	1.0	3.8	3.0	1.0	6.0	1.0	7.0
27	1.0	6.3	4.0	2.0	6.0	1.0	7.0
28	1.0	6.1	4.0	3.0	6.0	1.0	7.0
29	1.0	1.8	0.0	0.0	7.0	1.0	8.0
30	1.0	3.7	6.0	1.0	7.0	1.0	8.0
31	1.0	4.4	0.0	2.0	7.0	1.0	8.0
32	1.0	4.2	3.0	3.0	7.0	1.0	8.0
33	1.0	2.2	1.0	0.0	10.0	1.0	9.0
34	1.0	4.0	0.0	1.0	10.0	1.0	9.0
35	1.0	5.1	2.0	2.0	10.0	1.0	9.0
36	1.0	6.3	0.0	3.0	10.0	1.0	9.0
37	1.0	2.5	0.0	0.0	7.0	0.0	10.0
38	1.0	3.7	0.0	1.0	7.0	0.0	10.0
39	1.0	4.1	0.0	2.0	7.0	0.0	10.0
40	1.0	7.2	0.0	3.0	7.0	0.0	10.0
41	1.0	2.4	2.0	0.0	8.0	1.0	11.0
42	1.0	5.0	6.0	1.0	8.0	1.0	11.0
43	1.0	5.1	3.0	2.0	8.0	1.0	11.0
44	1.0	5.8	5.0	3.0	8.0	1.0	11.0
45	1.0	2.8	3.0	0.0	9.0	1.0	12.0
46	1.0	5.7	2.0	1.0	9.0	1.0	12.0
47	1.0	5.3	3.0	2.0	9.0	1.0	12.0

You will see that we now have 4 rows for each child and the *ind* variable identifies these as records 0, 1, 2 and 3 which correspond to the years 1986, 1988, 1990 and 1992, respectively. We will use this new dataset in the rest of the practical. So at the top of the window select **Choose** from the dataset pull down list and select *readlong* which has now appeared in the dataset list, and then click on the **Use** button.

It is useful to first take a look at the data graphically, which we can do by using one of Stat-JR's plotting templates; Firstly select Choose from the Template pull down list and here we'll use *XYGroupPlotFilter* (clicking on *Plots* in the cloud of terms above the list of templates will list all plotting templates available). Select this template from the list and click on **Use** and then **Run**. We will use this template to produce a plot of the reading test scores for the first 10 children; to do this we select the inputs as follows:



Clicking on **Run** and selecting *graphxygroup.svg* from the objects list produces the following plot, towards the bottom of the page:



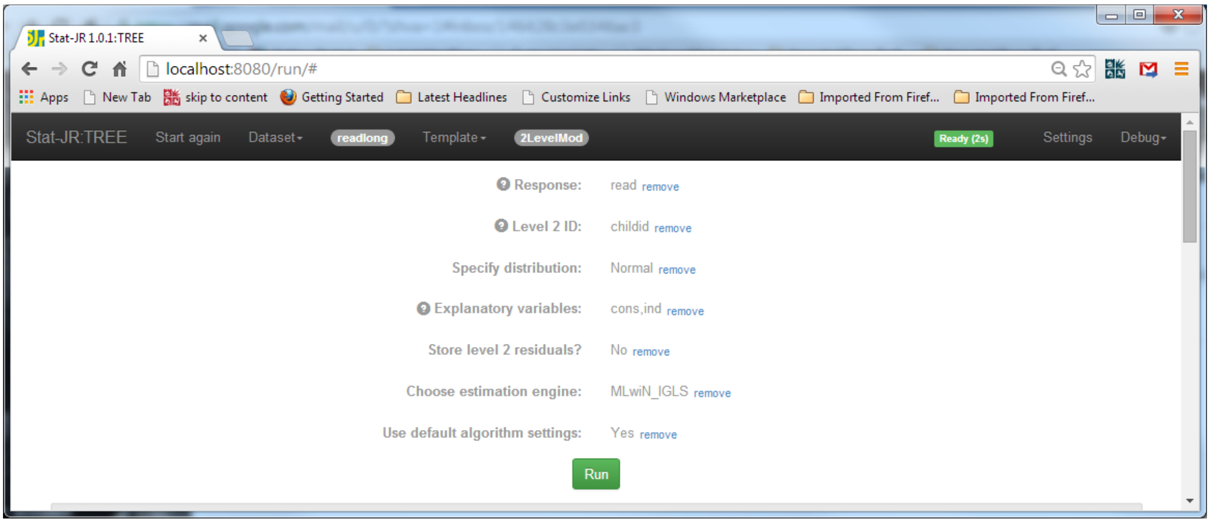
For these first ten children we see generally upward trends but lots of variability.

Fitting a basic linear growth model (random intercepts)

We will now consider fitting the random intercepts model that we covered in the lecture. We will do this first using maximum likelihood and the IGLS method in MLwiN before moving on to fitting the model using MCMC and the e-STAT engine.

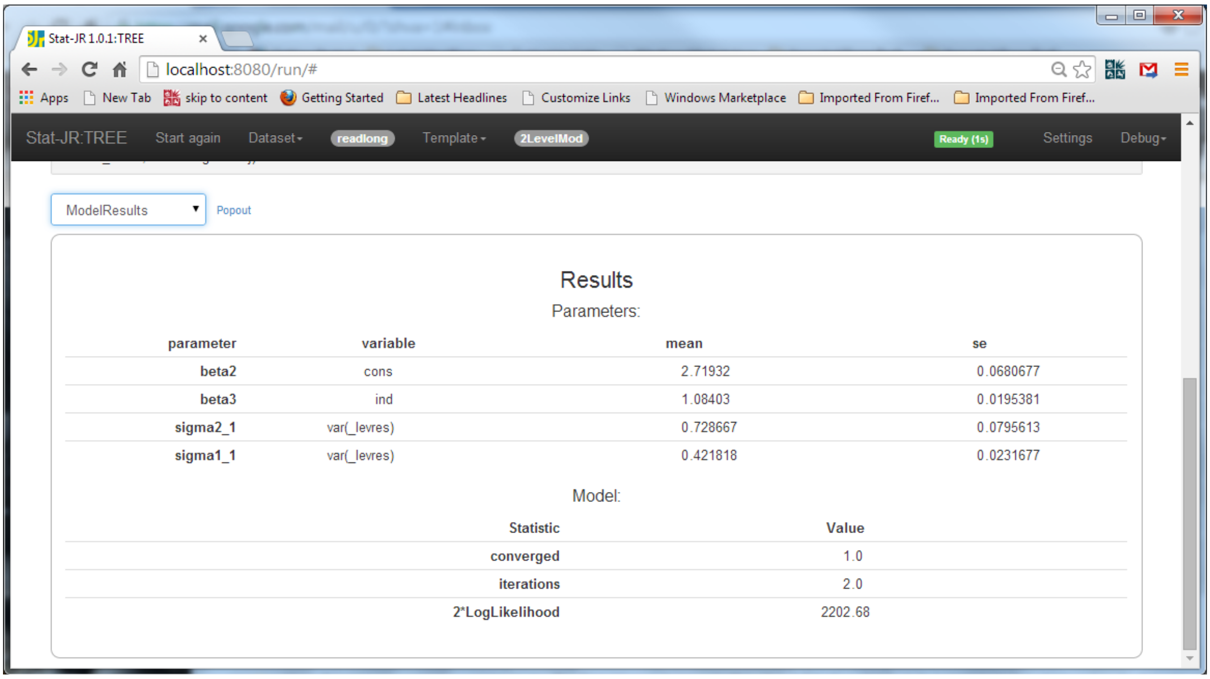
We will use variable *ind* to represent time; as this variable is defined from 0 to 3, the intercepts will represent the values of reading at time point 0 (1986).

Stat-JR has many templates for model fitting and here we will use the *2LevelMod* template that specifically fits random intercept models. So, return to the main Stat-JR tab, select *2LevelMod* from the templates list, and then click on **Use**. Fill in the inputs as follows:



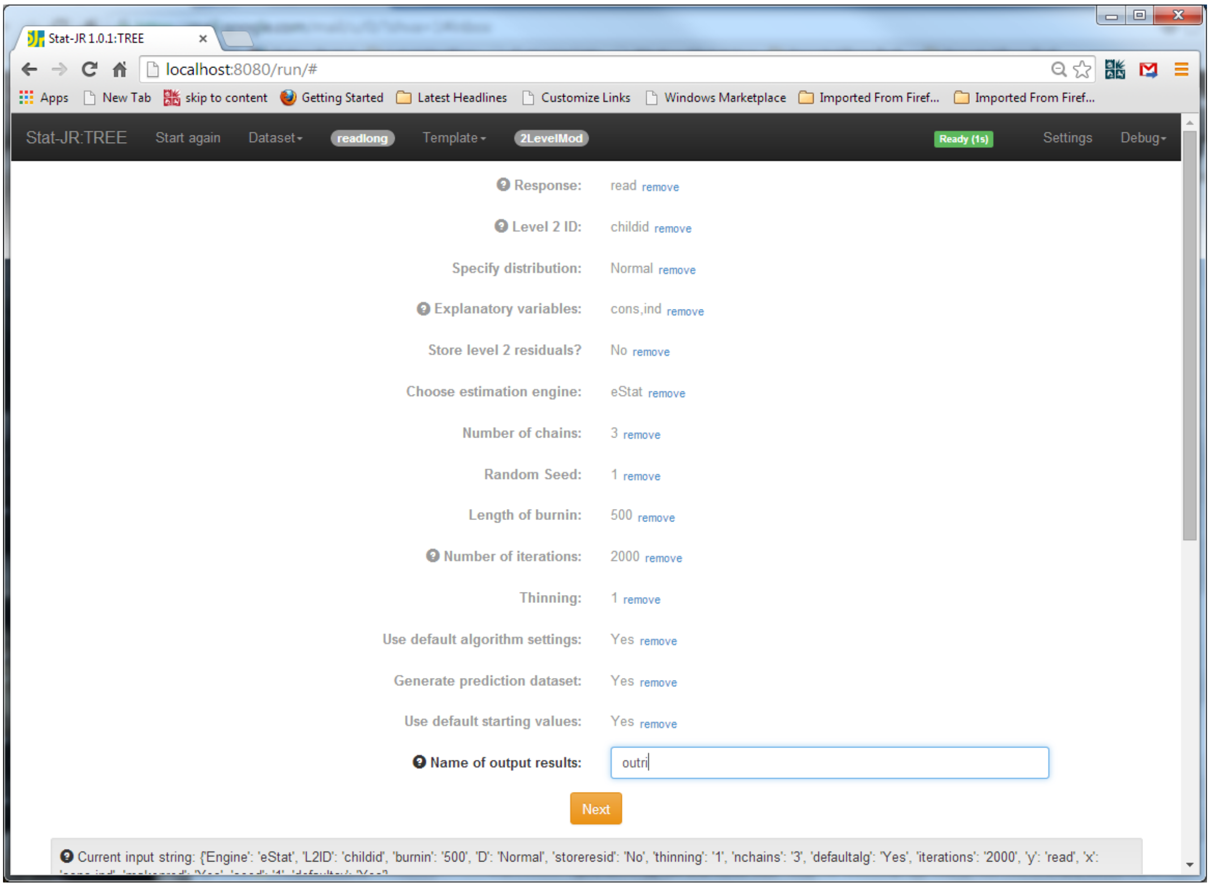
Here we define *read* as our response variable and have two predictor variables, *cons* for the intercept and *ind* for the time effects. We also need to tell Stat-JR that *childid* identifies the level 2 identifiers.

Clicking on **Run** gives many outputs in the objects list towards the bottom of the page, and if we select *ModelResults* from the pull-down list we see the following:

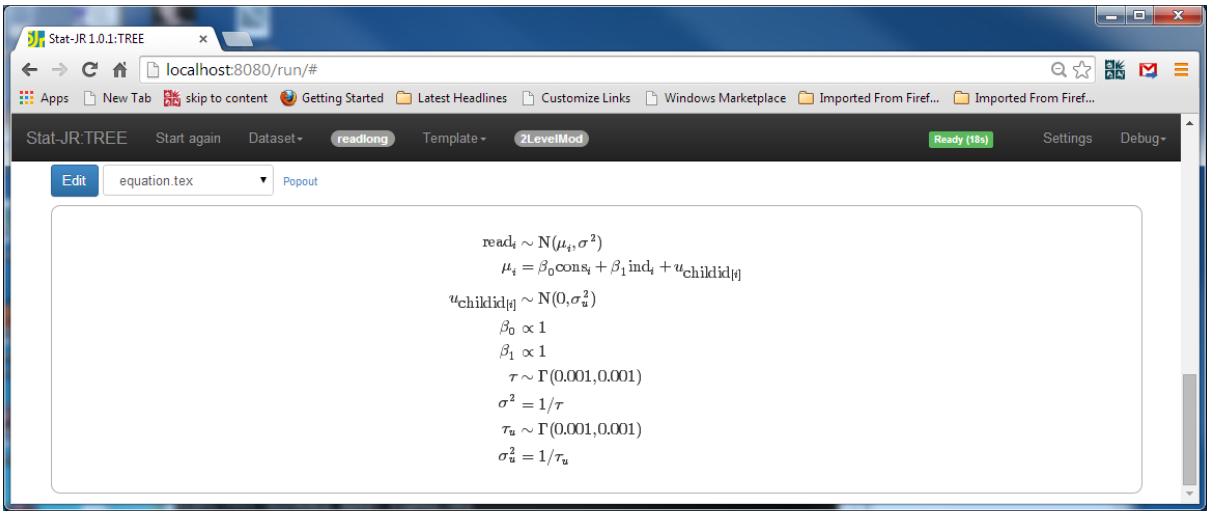


Here we see estimates for the intercept (*beta2*) and slope (*beta3*), and the variances at the child (*sigma2_1*) and residual (*sigma1_1*) levels. So, at time point zero, we expect a reading test score of 2.72, increasing by 1.08 for each test (2 year period). Much of the variability is at level 2 (between children); in fact the variance partition coefficient (VPC) = 0.729/(0.729+0.422) = 0.633: i.e. 63% of the variability can be attributed to between-children differences.

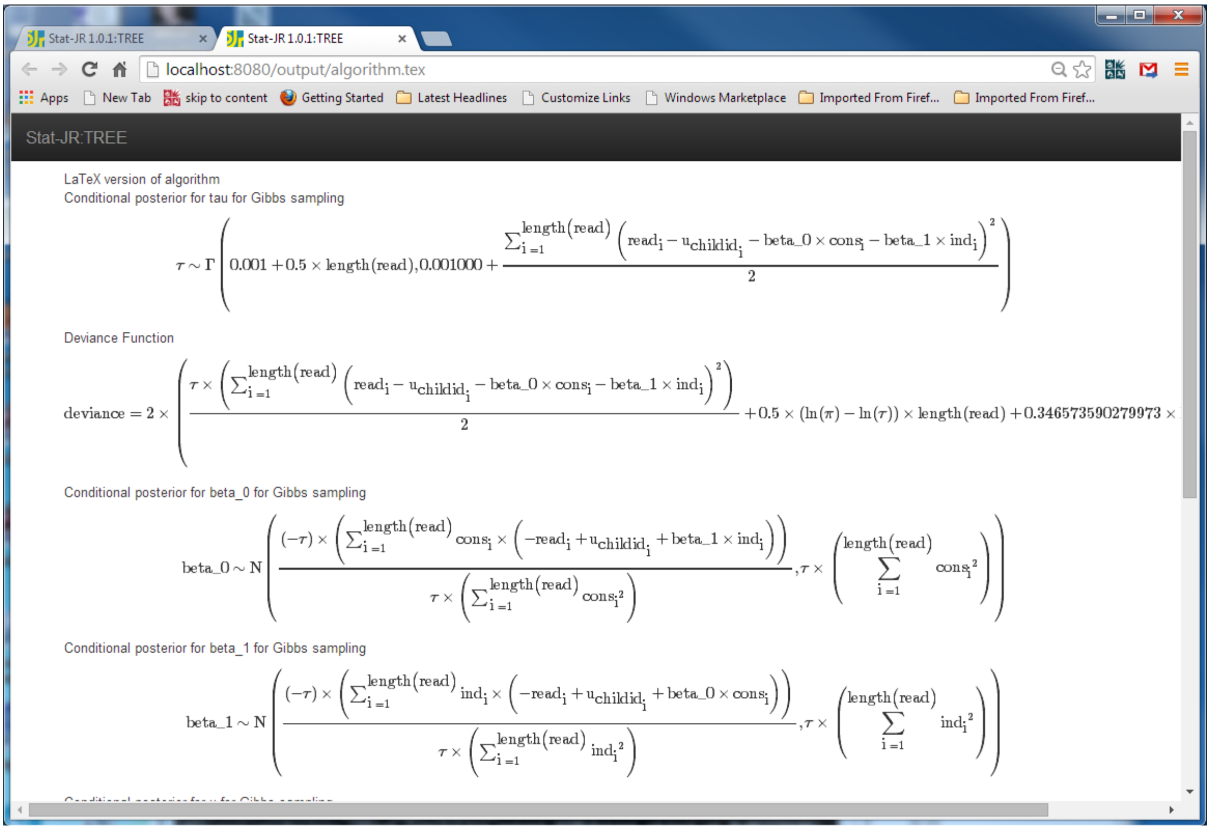
We can also fit this using MCMC via Stat-JR's e-STAT engine. To do this, click on the **remove** text next to the Choose estimation engine input and then make the following alternative choices:



Here we will run 3 MCMC chains each for 2000 iterations after a burnin of 500 iterations. Clicking on the **Next** button will prompt Stat-JR to create the algorithm, and then the program code, for fitting the model. We can observe both the model code (*model.txt*) and the model in mathematical form (*equation.tex*) in the objects pane:

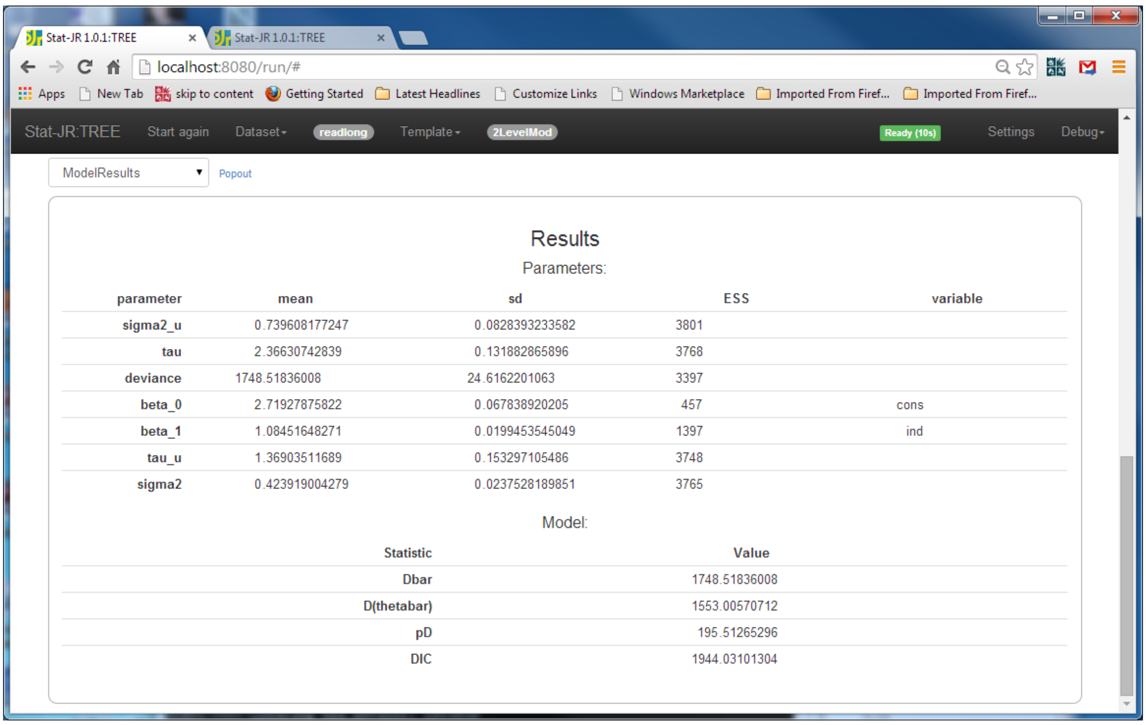


When the code has been compiled we can look at some of the other outputs created, for example the output *algorithm.tex* displays the MCMC algorithm that is to be used. This can be put in its own tab in the browser by clicking on the **Popout** text next to the object pull down list.



For this model we have a Gibbs sampling algorithm for all parameters.

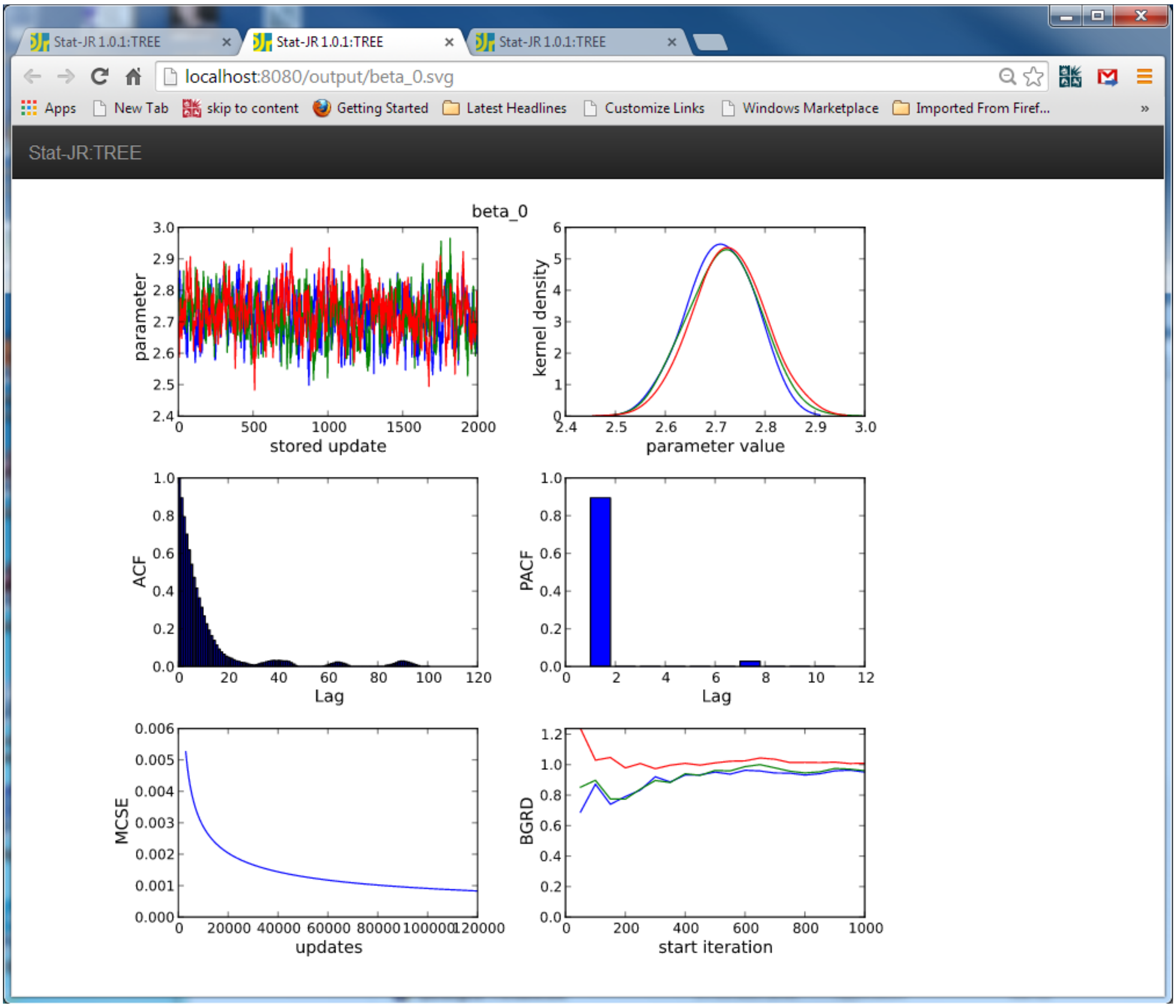
Returning to the main tab, clicking on the **Run** button will run the code produced. When this has finished, we can select *ModelResults* from the list of objects:



Here we see similar estimates to those we saw from IGLS. Below we compare these estimates in a table:

Parameter	IGLS Estimate (SE)	MCMC Estimate (SD)
β_0 (intercept)	2.719 (0.068)	2.719 (0.069)
β_1 (slope)	1.084 (0.020)	1.085 (0.020)
σ^2_u (level 2 variance)	0.729 (0.080)	0.740 (0.083)
σ^2_e (level 1 variance)	0.422 (0.023)	0.424 (0.024)

The level 2 variance is slightly larger in MCMC but this is a posterior mean estimate whilst IGLS gives the mode. The effective sample sizes (ESS) for all parameters are reasonable, although the ESS for β_0 is slightly lower. We can select the MCMC diagnostics for this parameter by selecting *beta_0.svg* from the object list and popping it out:



Running left to right down the rows for each parameter, the charts include a trace plot of the 2,000 estimates for each chain, a kernel density plot of the posterior distribution for each chain, plots of the autocorrelation (ACF) and partial auto-correlation (PACF) functions for assessing chain mixing, a Monte Carlo standard error (MCSE) plot, and finally a plot of the Brooks-Gelman-Rubin diagnostic (BGRD).

- The **trace plot** provides some indication of how well a chain is mixing: as a crude test, the absence of large white patches in the plot would indicate that the whole of the posterior distribution is being visited in a short space of time, suggesting the chain is mixing well; here you can see all 3 chains plotted in different colours.

- The **kernel density plot** is like a smoothed histogram of the posterior distribution; again, you can see the 3 chains plotted in different colours.

- The **ACF** measures how correlated the values in the chain are with their close neighbours: an independent chain will have approximately zero autocorrelation at each lag.

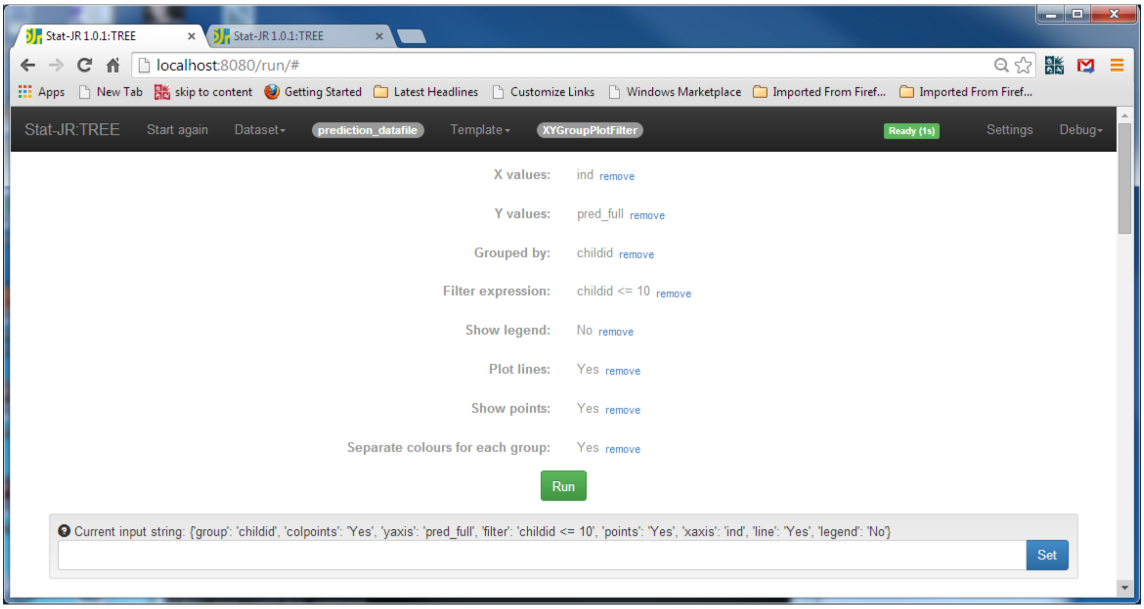
- The **PACF** measures discrepancies in the AR(1) process (referring to the fact that the Markov chain should have a power relationship in the lags (i.e. if ACF(1) = rho, then ACF(2) = rho^2, etc.); normally, the **PACF** should have values 0 after lag 1).

- The **MCSE chart** plots the Monte Carlo standard error of the mean against the number of iterations. The MCSE is an indication of the accuracy of the mean estimate (MCSE = SD/√n, where SD is the standard deviation from the chain of values, and n is the number of iterations), and allows the user to calculate how long to run a chain to achieve a mean estimate with a particular desired MCSE.

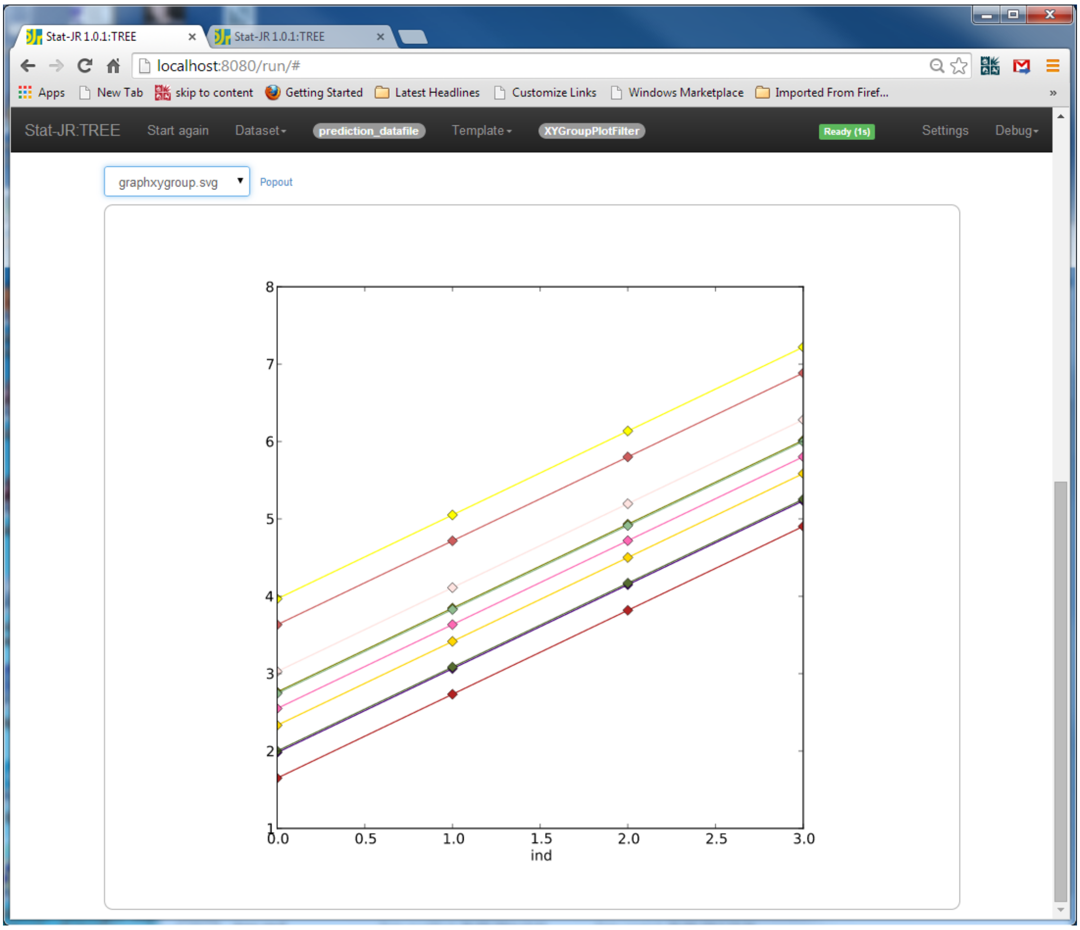
- The final plot charts the **Brooks-Gelman-Rubin diagnostic (BGRD)** statistic, with the width of the central 80% interval of the pooled runs in green, the average width of the 80% intervals within the individual runs in blue, and their ratio *BGRD* (= pooled / within) in red. *BGRD* would generally be expected to be greater than 1 if the starting values are suitably over-dispersed. Brooks and Gelman (1998) emphasise that one would wish to see convergence of R to 1, and with convergence of both the pooled and within interval widths to stability (but not necessarily to 1).

Here we see that the chains are mixing reasonably well, though after only 2,000 iterations there is still some variability in the projected kernel density plots. The ACF shows that there is some correlation between draws that are up to about 25 iterations apart. The other diagnostics all look fine.

Amongst the outputs produced when running the model using the e-STAT engine is a predictions dataset that contains predicted values for the model and can be used to graphically view the model fit. We will do this, as before, for the first 10 children. Return to the main Stat-JR tab and select *XYGroupPlotFilter* as the template, and *prediction_datafile* (which is the name given to the latest predictions produced) as the dataset, remembering to press the corresponding **Use** button after each selection. Next we fill in the inputs as shown:



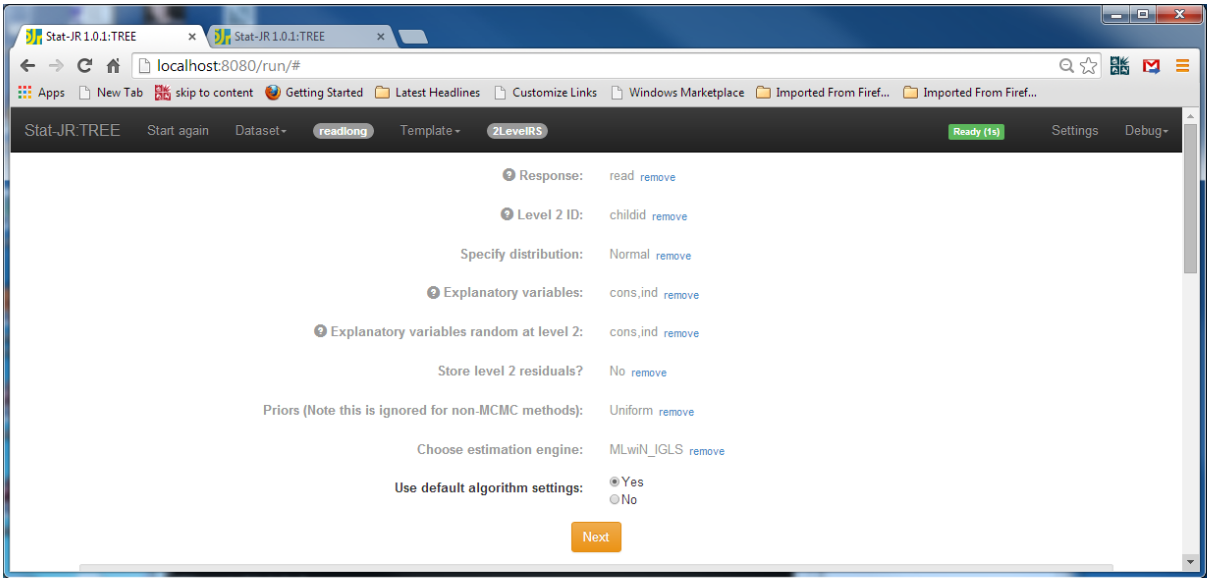
If we click on **Run** and select *graphxygroup.svg* we will see the predictions as follows:



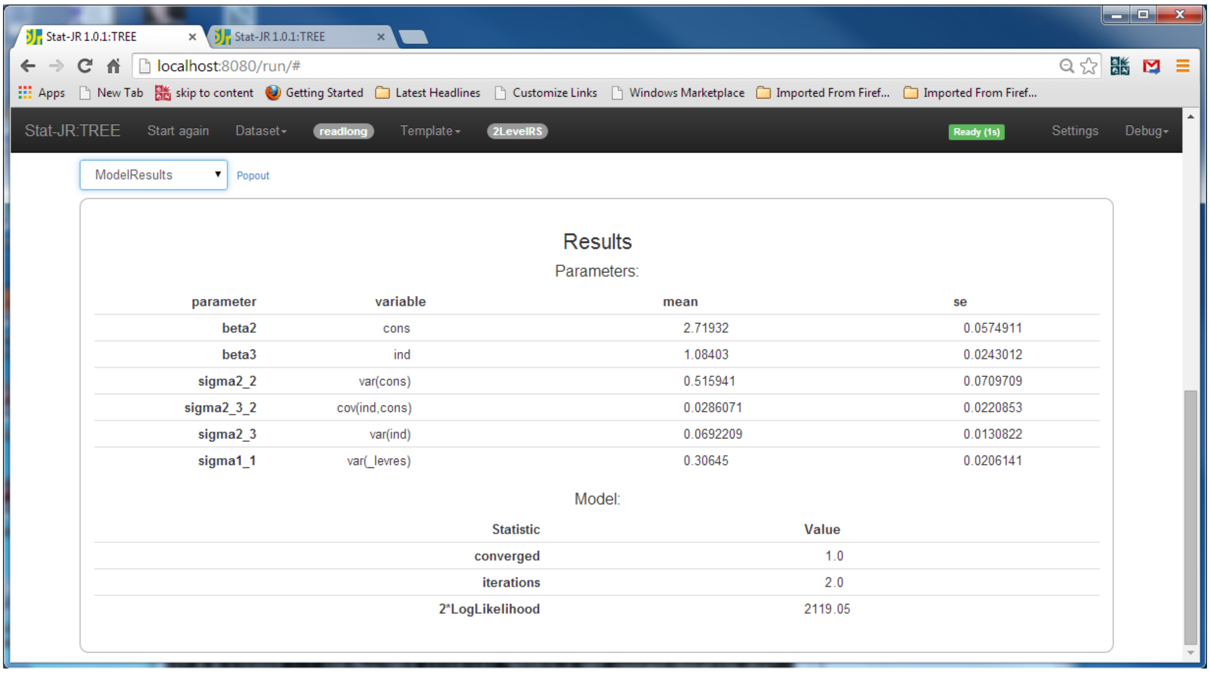
So, as we expect, we see parallel lines for each child, with large variability between them. The random intercepts model only allows the intercept to vary between children whilst the slope is fixed; as you'll remember from the lectures, we also considered the random slopes model which removes this constraint. We'll investigate this in the section below; once again we'll consider both the frequentist approach, via interoperability with MLwiN, and MCMC using Stat-JR's e-STAT engine.

Fitting a random slopes model

Firstly, return to the top of the main Stat-JR screen and this time select *2LevelRS* from the template list and *readlong* for the dataset. Then click on **Run** and fill in the inputs as shown below:

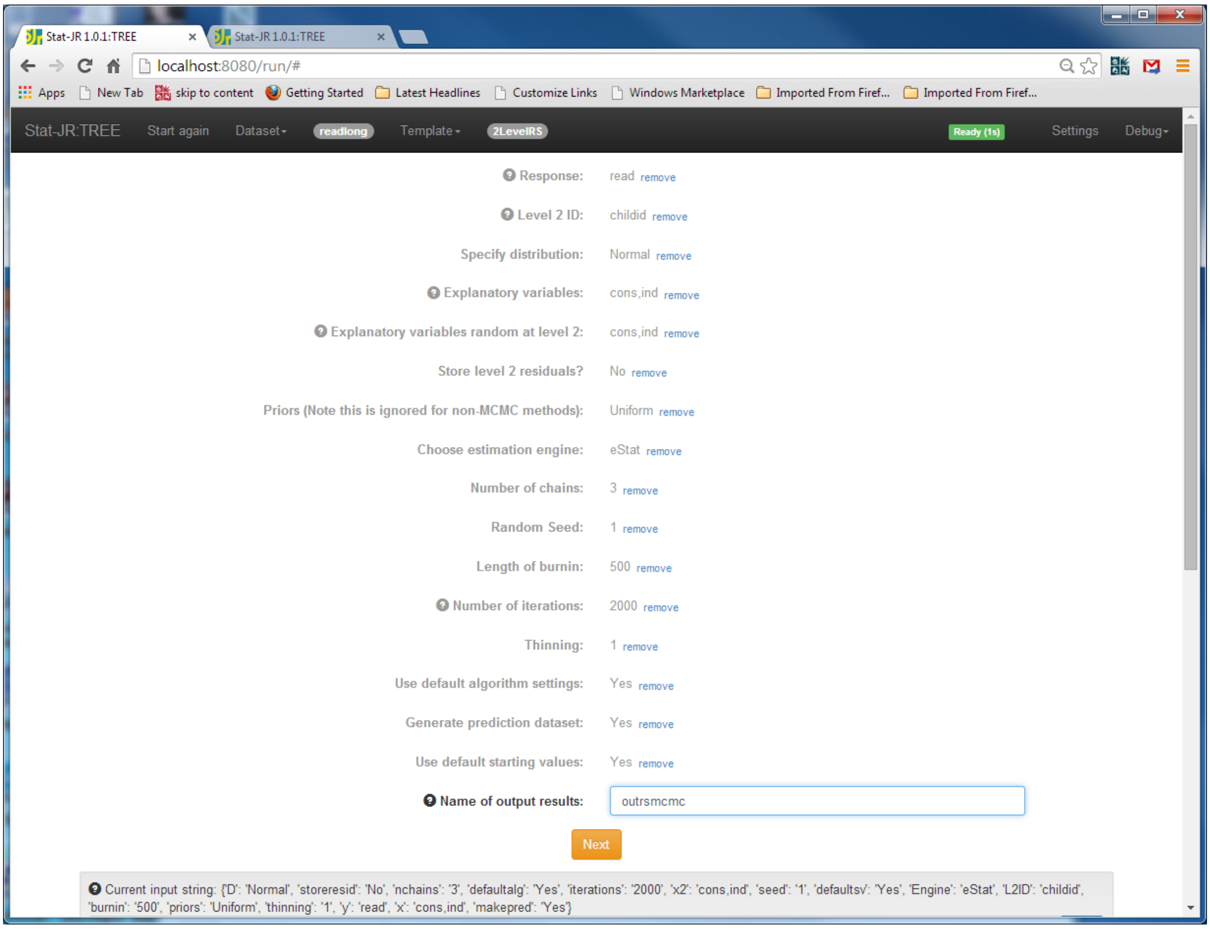


Note that here we are looking at *frequentist* methods in MLwiN, and so, as it states, the "**Priors**" input will be ignored. Clicking on **Next** and **Run** gives the results shown below if we choose *ModelResults* from the objects list:

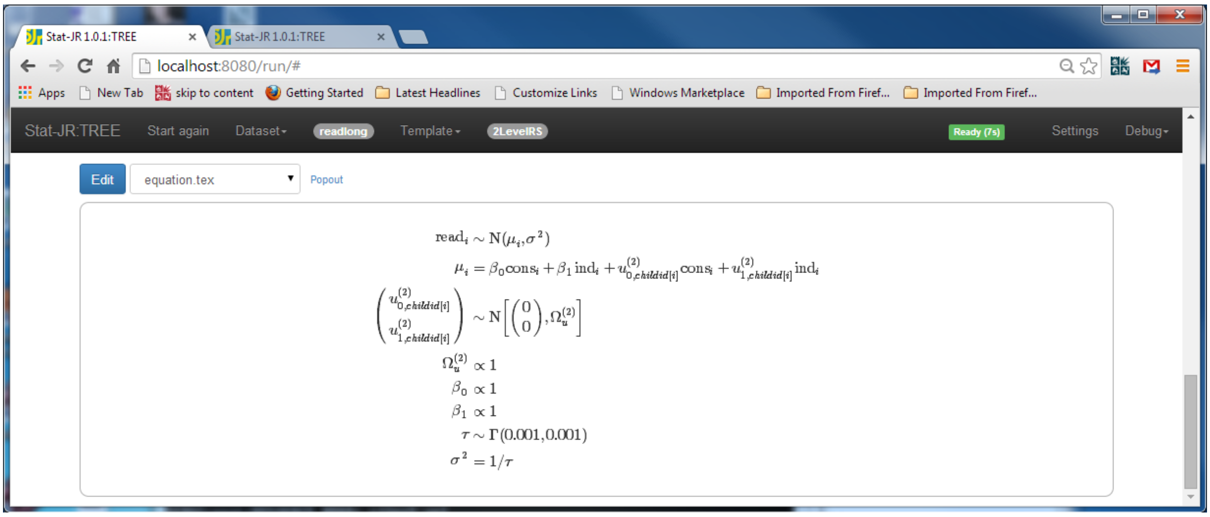


Here we see two additional parameters for the slopes (*ind*) variance at level 2 and the covariance between intercepts and slopes. The Deviance (*-2*LogLikehood*) value for this model is 2119.1 compared with 2202.7 earlier for the random intercepts model. We therefore have a change in deviance of 83.6 which follows a chi-squared distribution with 2 degrees of freedom (for the 2 additional parameters). This corresponds to a *p*-value of <0.001, and so the random slopes model is a significantly better model for this dataset.

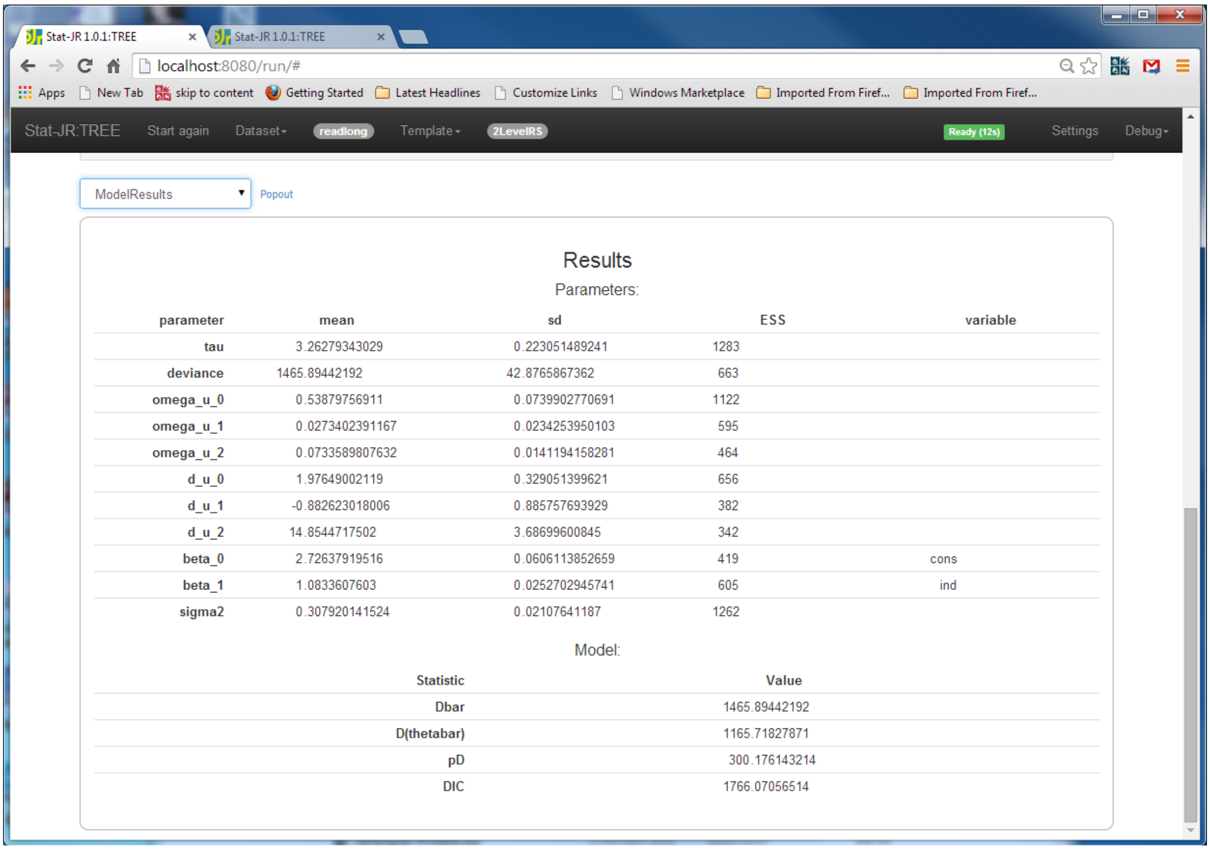
We will also fit the model using MCMC. If we click on **remove** next to **Choose estimation engine** we can redo the inputs as shown below:



As you can see, we have chosen to use *Uniform* priors for now. When we press **Next** we can view the the mathematical description (*equation.tex*) as shown below:



Clicking on **Run** will run the model, and after a short while we can select *ModelResults* from the outputs in the right-hand pane thus:



parameter	mean	sd	ESS	variable
tau	3.26279343029	0.223051489241	1283	
deviance	1465.89442192	42.8765867362	663	
omega_u_0	0.53879756911	0.0739902770691	1122	
omega_u_1	0.0273402391167	0.0234253950103	595	
omega_u_2	0.0733589807632	0.0141194158281	464	
d_u_0	1.97649002119	0.329051399621	656	
d_u_1	-0.882623018006	0.885757693929	382	
d_u_2	14.8544717502	3.68699600845	342	
beta_0	2.72637919516	0.0606113852659	419	cons
beta_1	1.0833607603	0.0252702945741	605	ind
sigma2	0.307920141524	0.02107641187	1262	

Statistic	Value
Dbar	1465.89442192
D(thetabar)	1165.71827871
pD	300.176143214
DIC	1766.07056514

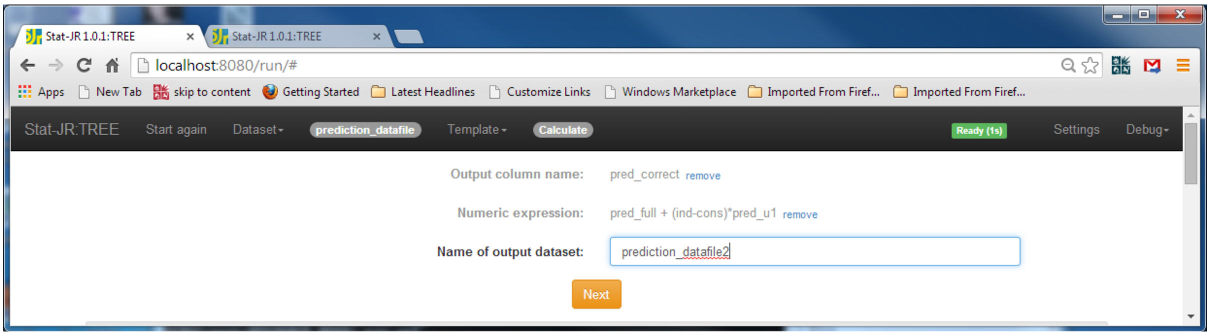
Again, we can compare the different parameter estimates from MCMC and IGLS, as we did previously for the random intercepts model:

Parameter	**IGLS Estimate (SE)**	**MCMC Estimate (SD)**
β_0 (intercept)	2.719 (0.057)	2.726 (0.061)
β_1 (slope)	1.084 (0.024)	1.083 (0.025)
Ω_{u00} (level 2 intercept variance)	0.516 (0.071)	0.539 (0.074)
Ω_{u01} (level 2 intercept/slope covariance)	0.029 (0.022)	0.027 (0.023)
Ω_{u11} (level 2 slope variance)	0.069 (0.013)	0.073 (0.014)
σ^2_e (level 1 variance)	0.306 (0.021)	0.308 (0.021)

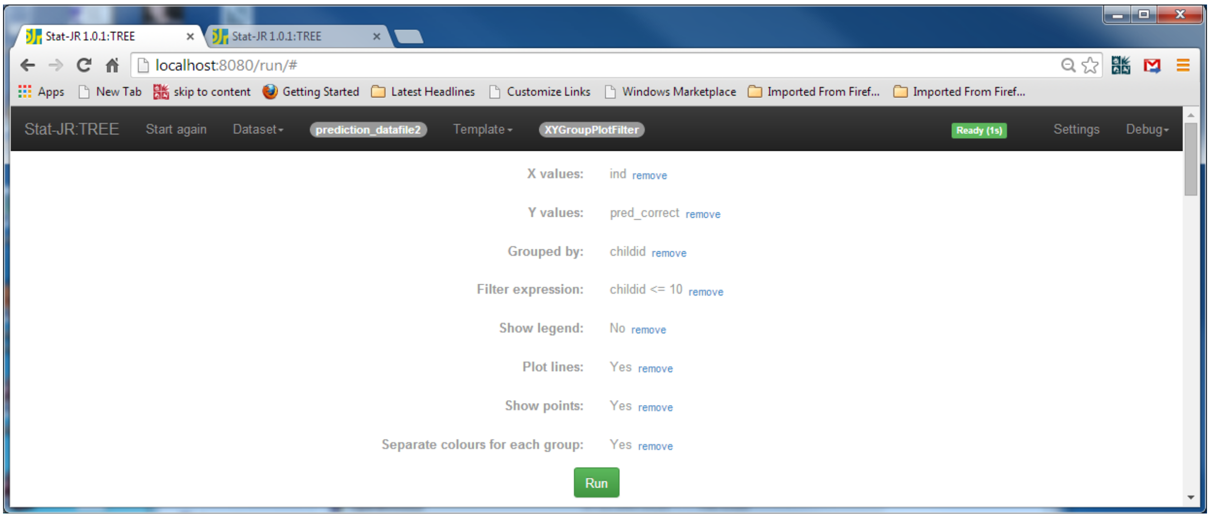
As before, we obtain similar estimates between the two methods, with the level 2 variances being slightly larger using MCMC but being posterior means. Note also that the Uniform prior is known to slightly overestimate the variances (Browne and Draper, 2000).

For both the random intercepts and random slopes models, estimated via MCMC, we have a DIC diagnostic; this comprises a combination of fit and complexity, and can be used for model selection. For the random intercepts model we had a DIC of 1944.0 whilst here the DIC has reduced to 1766.1 indicating the random slopes model is much better, as we also found when using IGLS.

We can plot the fitted values for the random slopes model to see that the lines are not parallel. To do this we need to return to the main screen and change dataset to *prediction_datafile.* We now first need to calculate the predictions manually for this model[1] using the *Calculate* template - see the screenshot below (**Output column name:** *pred_correct*; **Numeric expression:** *pred_full + (ind-cons)*pred_u1*; **Name of output dataset:** *prediction_datafile2*):

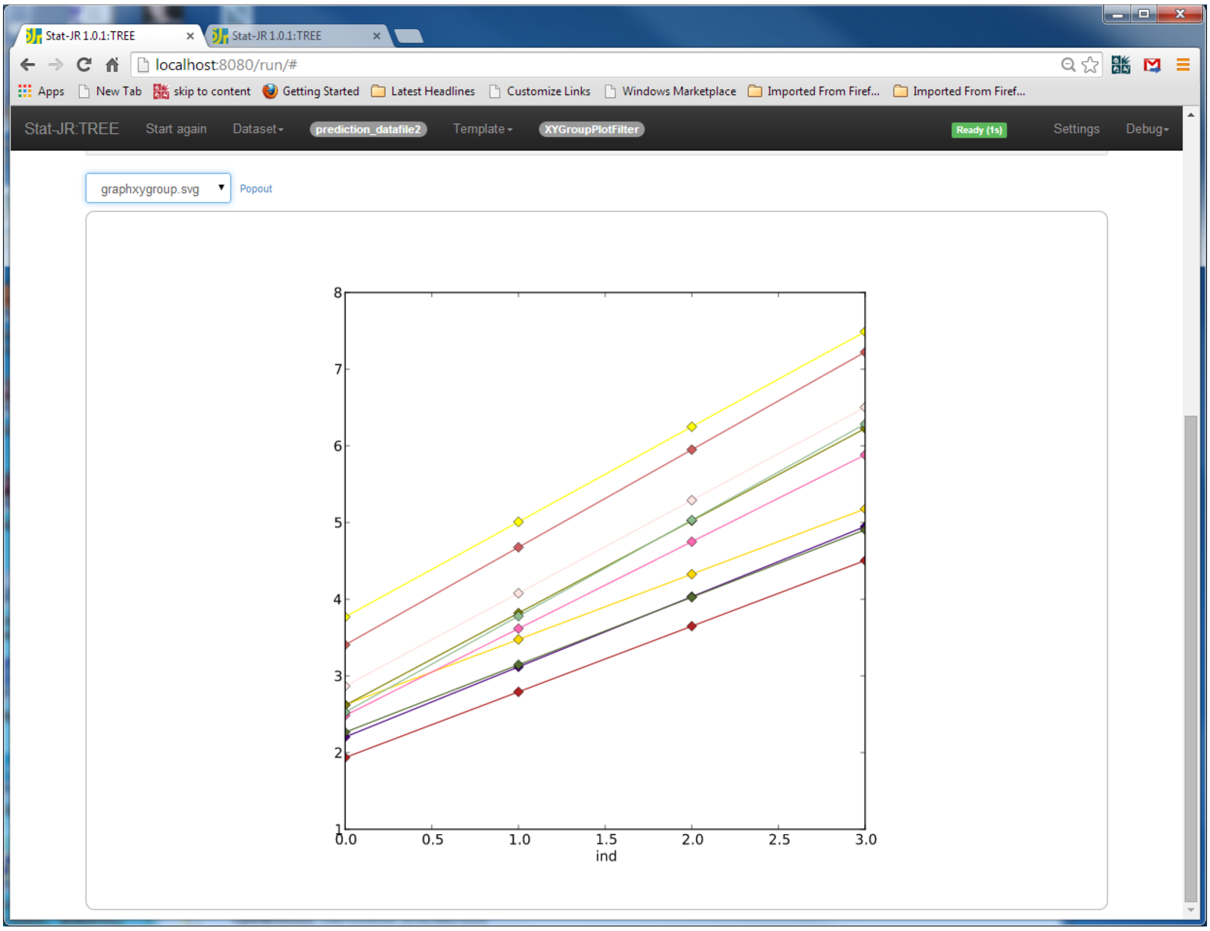


After pressing **Next**, then **Run,** this will add the corrected predictions to a new dataset called *prediction_datafile2* and we can proceed to plot the lines by selecting this dataset and the *XYGroupPlotFilter* template and choosing inputs thus:

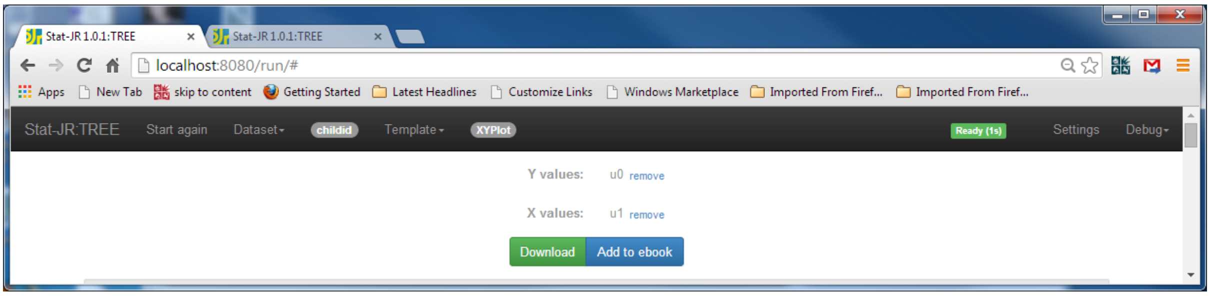


Pressing **Run** and selecting *graphxygroup.svg* gives the lines:

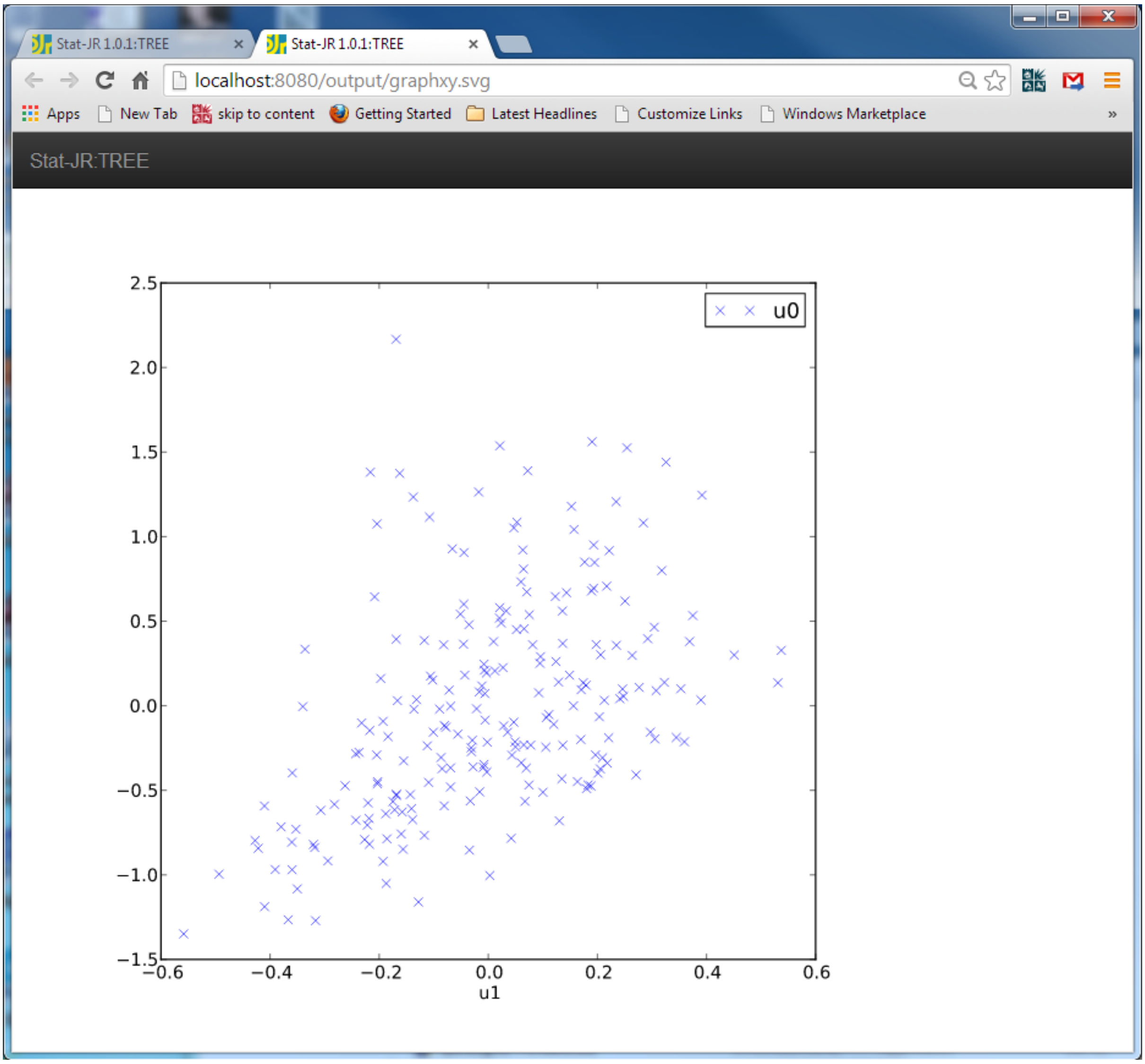
[1] Basically the column *pred_full* is constructed by simply adding all the predictions together, but for the random slopes models the predicted u variables for slopes need *multiplying* by their predictors and not just adding together. We therefore do this above, storing the result in *pred_correct*.



Here we see that the lines exhibit different slopes with a 'fanning out' pattern indicating a positive covariance between intercepts and slopes. To back this up we can also look at the residuals at level 2 as these are stored in a dataset called *childid.* If we wish to plot pairwise residuals we need to return to the top of the screen and choose *XYPlot* as the template and *childid* as the dataset. We can then set-up the inputs as shown (**Y values:** *u0*; **X values:** *u1*):

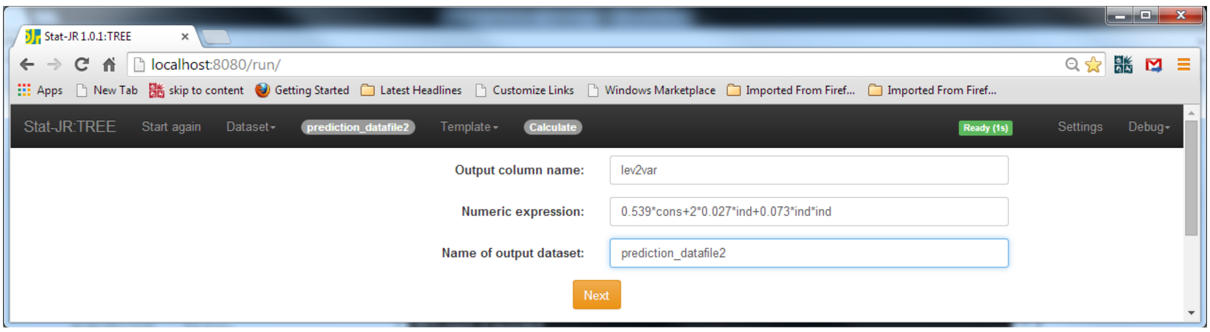


We can look at the plot in its own tab by popping it out as shown below:



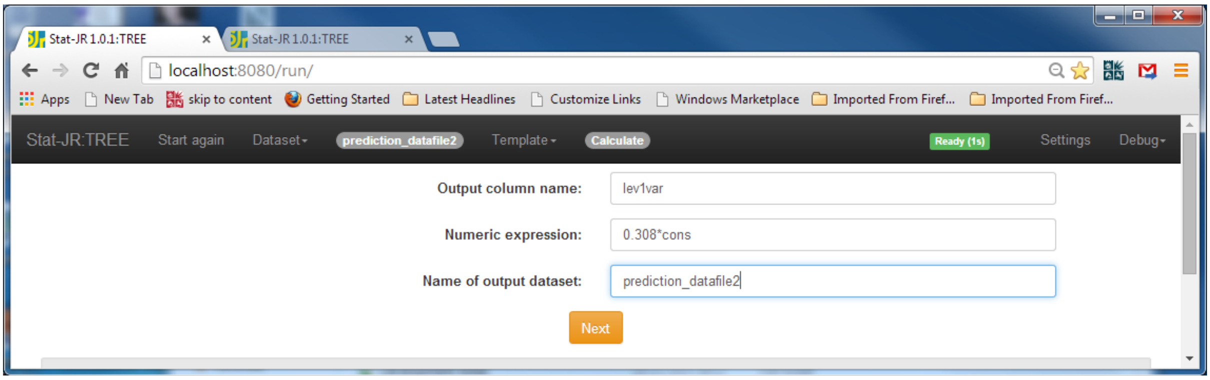
Here we see a slight positive correlation as indicated by the positive covariance at level 2.

Finally we can also calculate the variance function at levels 1 and 2 by firstly returning to the main screen and choosing *prediction_datafile2* as the dataset. Then via the *Calculate* template, inputs as follows (**Output column name**: *lev2var;* **Numeric expression:** *0.539*cons + 2*0.027*ind + 0.073*ind*ind;* **Name of output dataset:** *prediction_datafile2*):



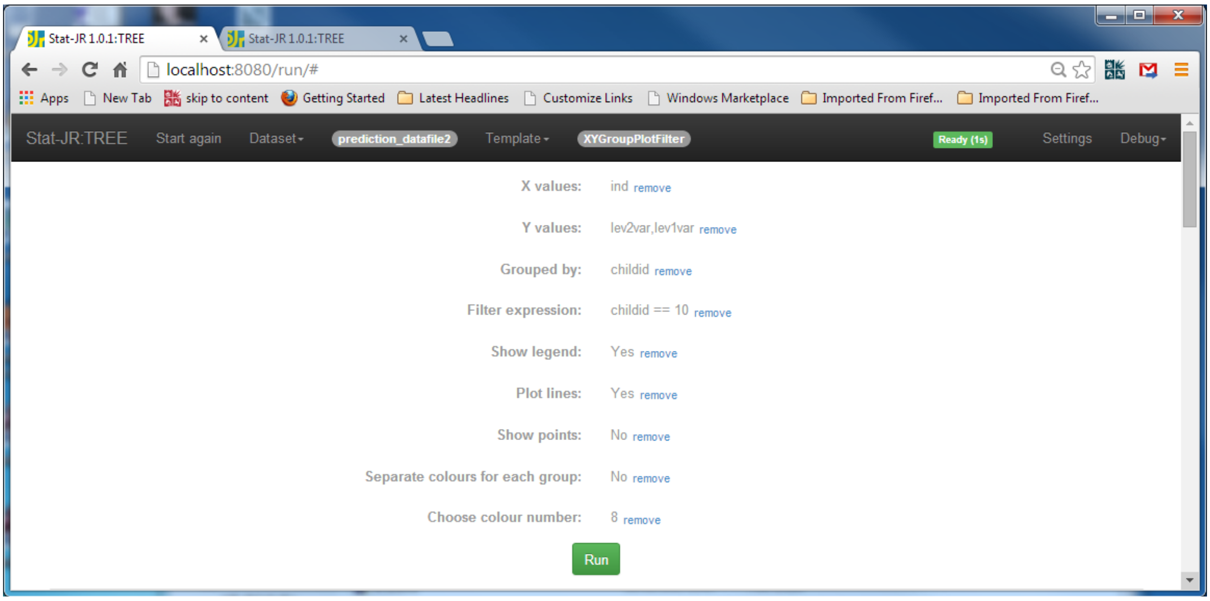
If we press **Next** and **Run**, this adds the column *lev2var* to the dataset *prediction_datafile2*. If we then press **Start again** (which you can find in the black bar towards the top of the browser window), we enter the following inputs, again using the *Calculate* template (**Output**

column name: *lev1var;* **Numeric expression:** *0.308*cons;* **Name of output dataset:** *prediction_datafile2*):

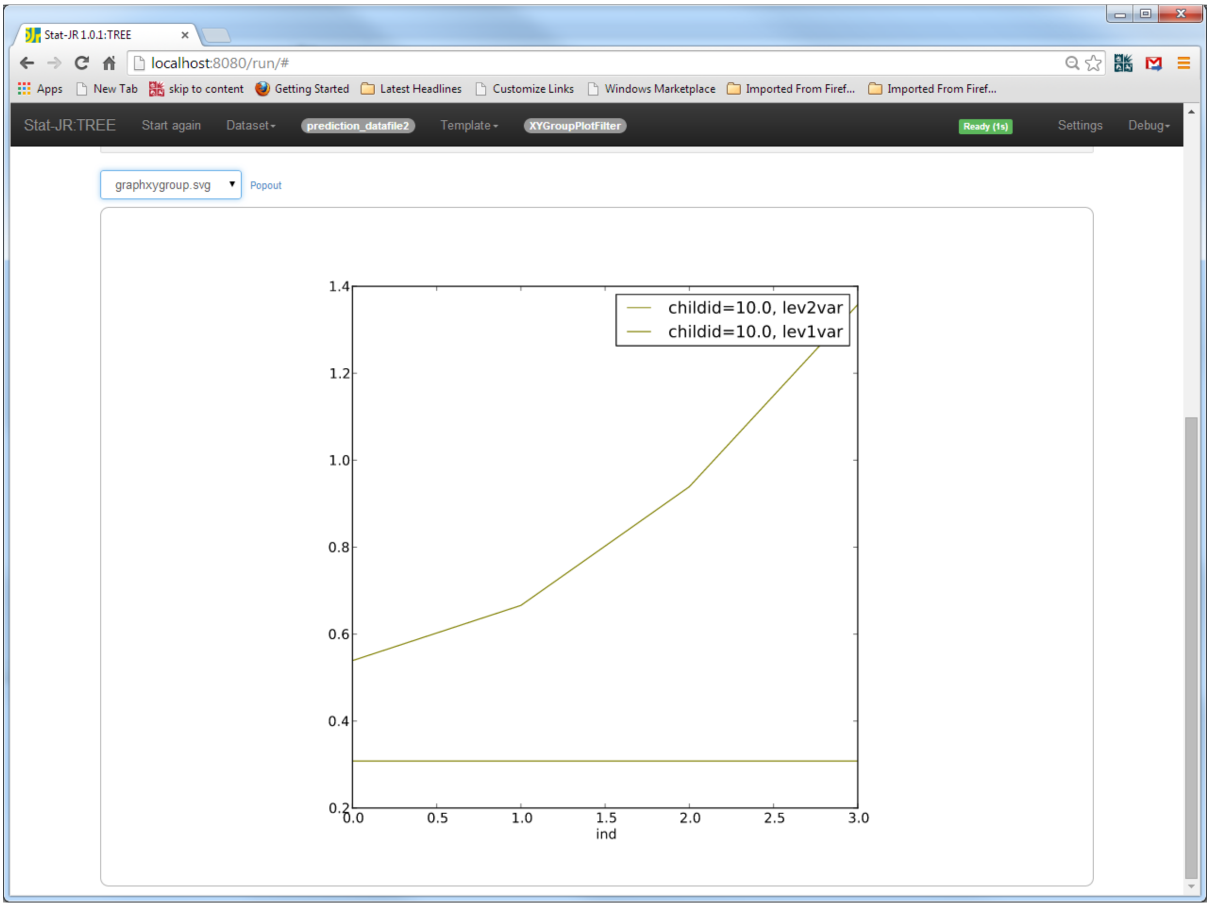


Pressing **Next** and **Run** adds the column *lev1var.*

We can then use the *XYGroupPlotFilter* template to plot the curves as follows:



This returns the following plot:



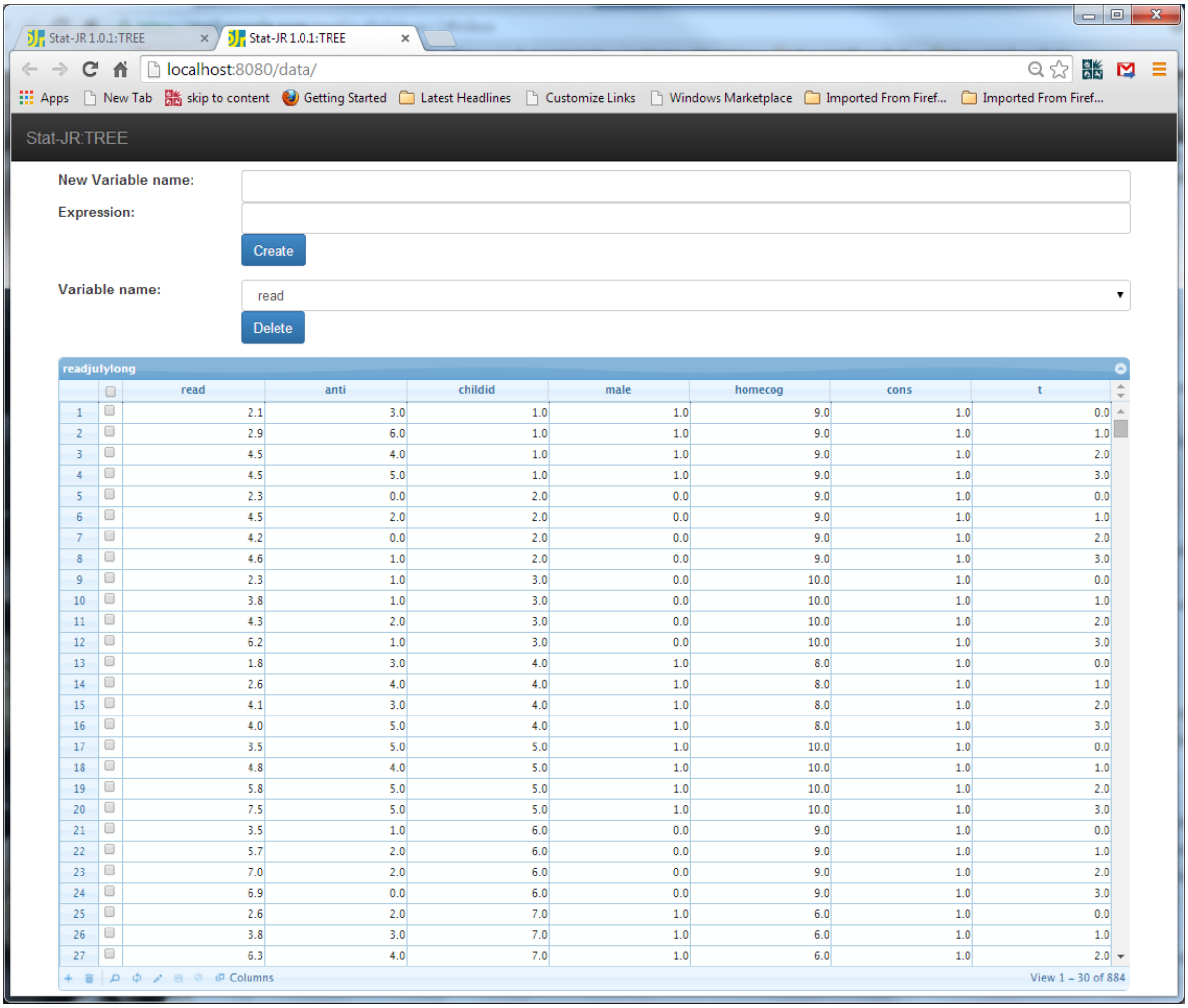
Here we see that the level 2 variance (top line) increases as time increases – this is to be expected, given the fanning out pattern.

MODELLING LONGITUDINAL DATA USING THE STAT-JR PACKAGE

Practical 2: Extensions to Growth Curve Models

Introduction

In this exercise we will continue our analysis of reading development using measurements taken on four occasions for 221 U.S. children. The dataset was described in the first practical where we fitted models using both interoperability to MLwiN, and MCMC via Stat-JR's e-Stat engine. In this practical we will simply use MCMC but you are welcome to investigate fitting the models using the other estimation methods that are on offer after finishing the practical. We will use the long form of the data (with 1 record per year) which we have stored in the dataset *readjulylong*. To start, open Stat-JR by clicking on *tree.cmd* and then find this dataset in the list and click **Use** to select it. If we select **View** from the Dataset menu we will see the following:

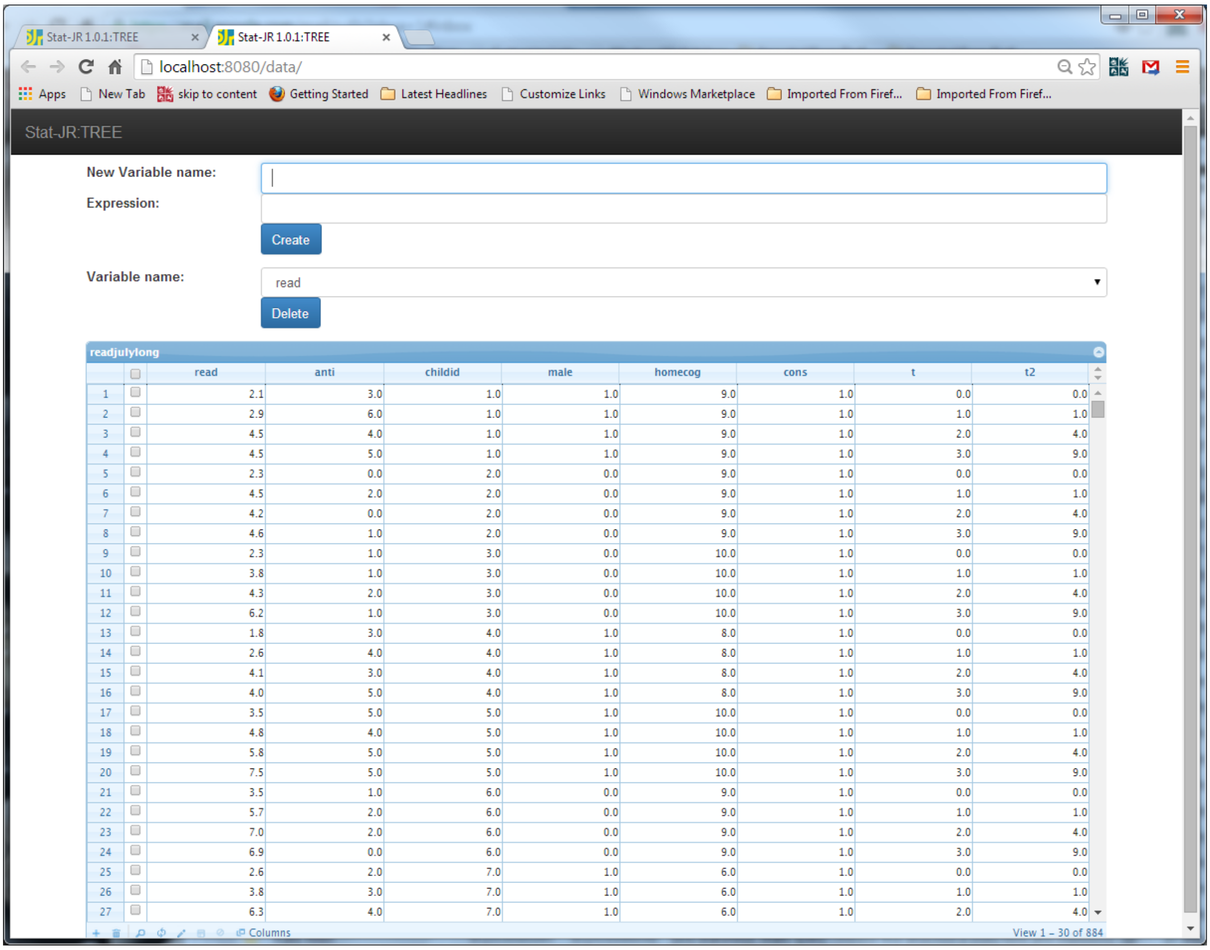


	read	anti	childid	male	homecog	cons	t
1	2.1	3.0	1.0	1.0	9.0	1.0	0.0
2	2.9	6.0	1.0	1.0	9.0	1.0	1.0
3	4.5	4.0	1.0	1.0	9.0	1.0	2.0
4	4.5	5.0	1.0	1.0	9.0	1.0	3.0
5	2.3	0.0	2.0	0.0	9.0	1.0	0.0
6	4.5	2.0	2.0	0.0	9.0	1.0	1.0
7	4.2	0.0	2.0	0.0	9.0	1.0	2.0
8	4.6	1.0	2.0	0.0	9.0	1.0	3.0
9	2.3	1.0	3.0	0.0	10.0	1.0	0.0
10	3.8	1.0	3.0	0.0	10.0	1.0	1.0
11	4.3	2.0	3.0	0.0	10.0	1.0	2.0
12	6.2	1.0	3.0	0.0	10.0	1.0	3.0
13	1.8	3.0	4.0	1.0	8.0	1.0	0.0
14	2.6	4.0	4.0	1.0	8.0	1.0	1.0
15	4.1	3.0	4.0	1.0	8.0	1.0	2.0
16	4.0	5.0	4.0	1.0	8.0	1.0	3.0
17	3.5	5.0	5.0	1.0	10.0	1.0	0.0
18	4.8	4.0	5.0	1.0	10.0	1.0	1.0
19	5.8	5.0	5.0	1.0	10.0	1.0	2.0
20	7.5	5.0	5.0	1.0	10.0	1.0	3.0
21	3.5	1.0	6.0	0.0	9.0	1.0	0.0
22	5.7	2.0	6.0	0.0	9.0	1.0	1.0
23	7.0	2.0	6.0	0.0	9.0	1.0	2.0
24	6.9	0.0	6.0	0.0	9.0	1.0	3.0
25	2.6	2.0	7.0	1.0	6.0	1.0	0.0
26	3.8	3.0	7.0	1.0	6.0	1.0	1.0
27	6.3	4.0	7.0	1.0	6.0	1.0	2.0

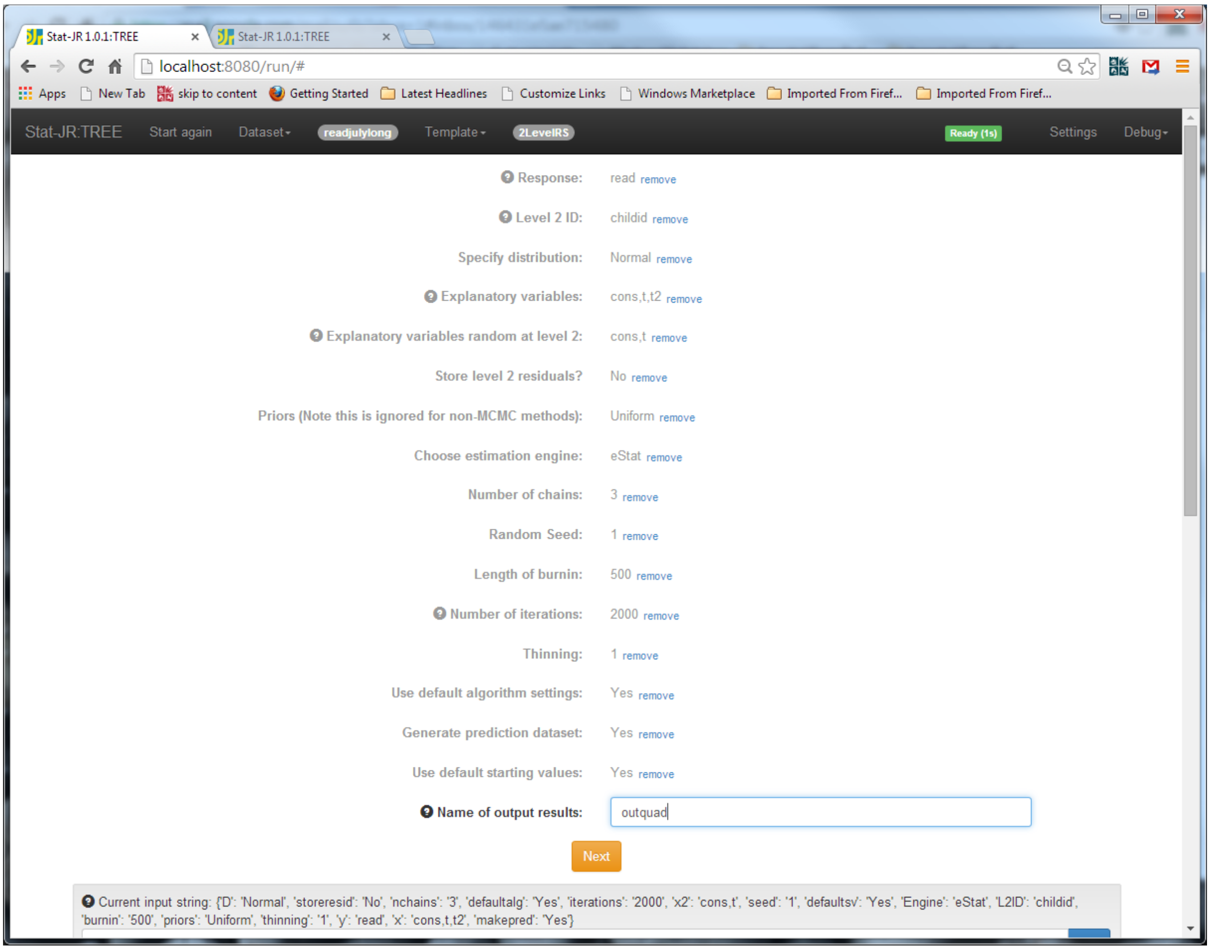
In this saved version we have named the time variable *t* (as opposed to *ind*). Again, *t* = 0 represents the first year of data (1986).

Quadratic Growth curve model

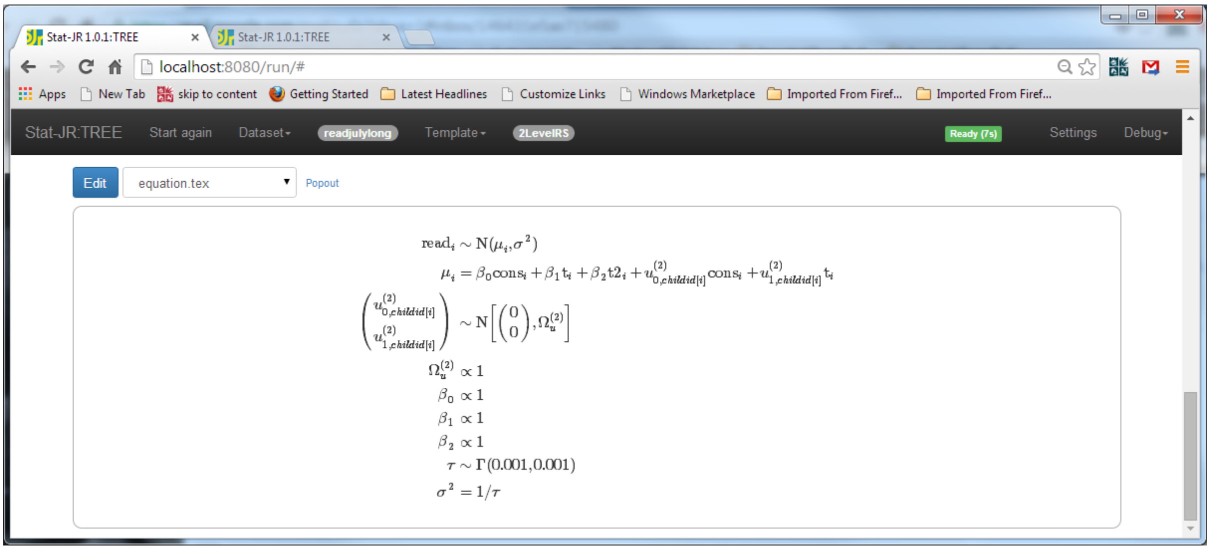
We will begin by extending the modelling to allow for a non-linear trend in the time variable. The quadratic term doesn't exist in the dataset therefore we will need to construct it. We could do this via the *Calculate* template or alternatively simply use the New variable options in this **View** data screen. We will create the variable *t2* by filling in the boxes as follows: **New Variable Name:** *t2*; **Expression:** *t*t*; click on **Create.** The new variable t2 should appear to the right of the list:



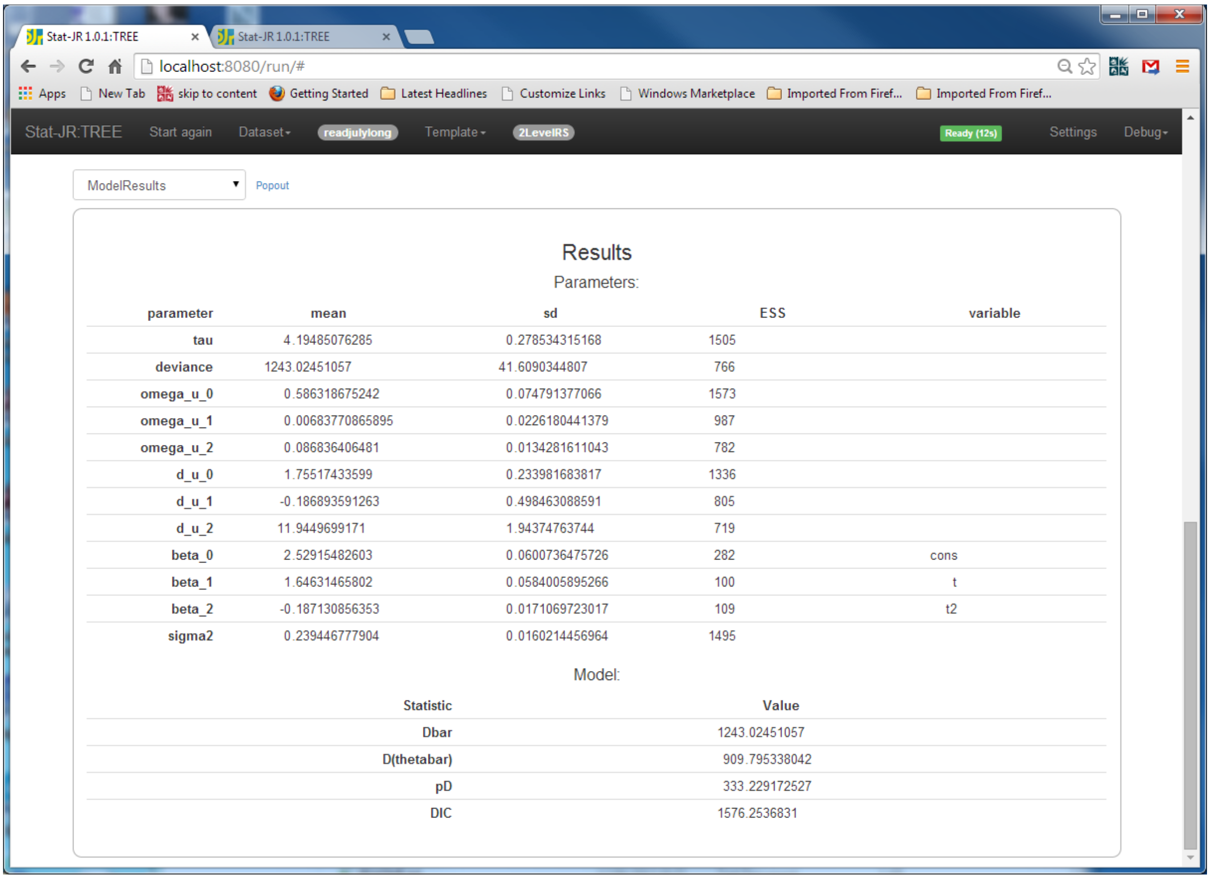
We now wish to add this quadratic term, *t2*, as a fixed effect to the last random slopes model from practical 1, so click on the **Choose** option from the Template list. Here select *2LevelRS* from the template list and click on **Use**. The inputs should then be as follows, noting that for now we are just adding *t2* to the **explanatory variables** list.



Clicking **Next** will start the algebra system that constructs the algorithm and then the code to fit the model will be written and compiled. While this is happening we can see that in the object pane we have a mathematical description of the model.



Clicking on the **Run** button will then fit the model and, when the timer stops, selecting *ModelResults* from the object list gives the following:



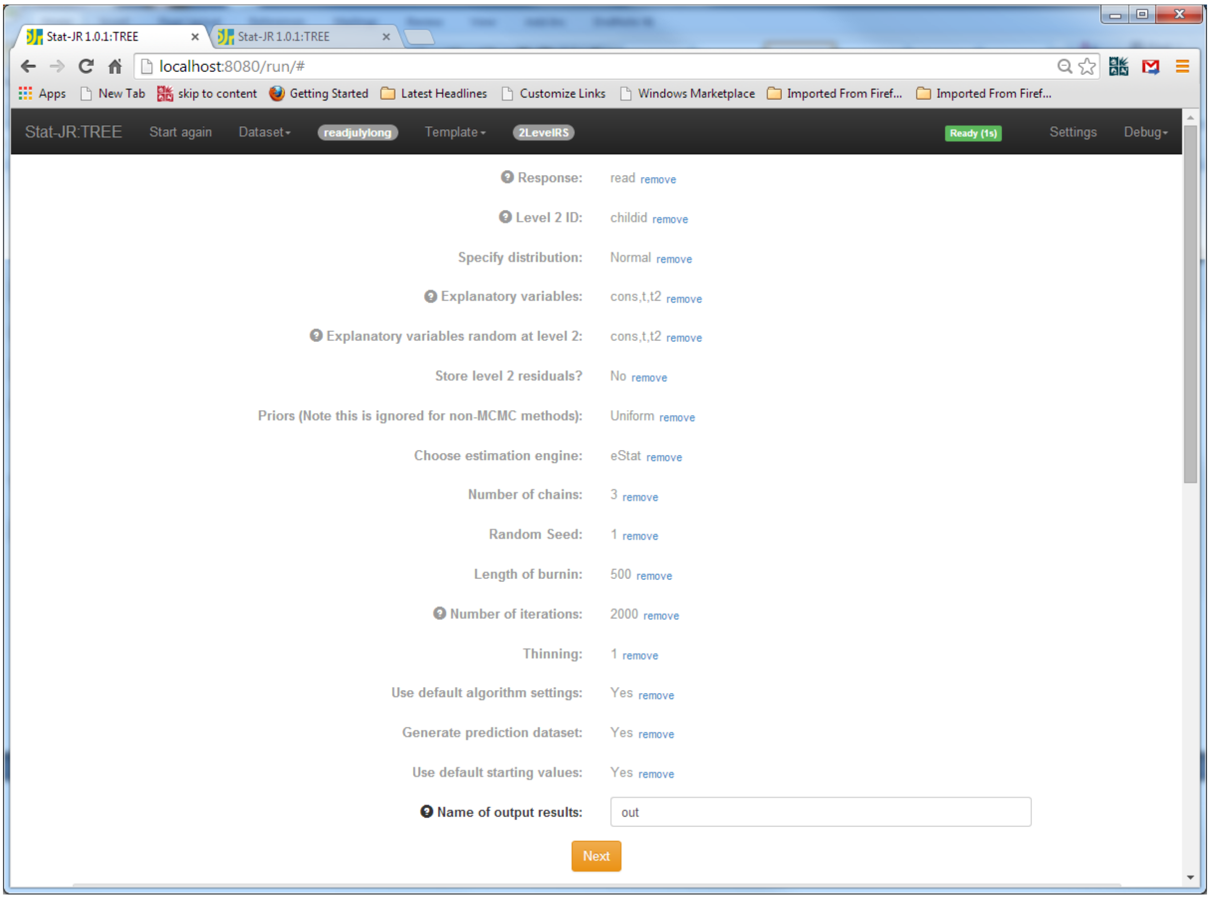
parameter	mean	sd	ESS	variable
tau	4.19485076285	0.278534315168	1505	
deviance	1243.02451057	41.6090344807	766	
omega_u_0	0.586318675242	0.074791377066	1573	
omega_u_1	0.00683770865895	0.0226180441379	987	
omega_u_2	0.086836406481	0.0134281611043	782	
d_u_0	1.75517433599	0.233981683817	1336	
d_u_1	-0.186893591263	0.498463088591	805	
d_u_2	11.9449699171	1.94374763744	719	
beta_0	2.52915482603	0.0600736475726	282	cons
beta_1	1.64631465802	0.0584005895266	100	t
beta_2	-0.187130856353	0.0171069723017	109	t2
sigma2	0.239446777904	0.0160214456964	1495	

Model:

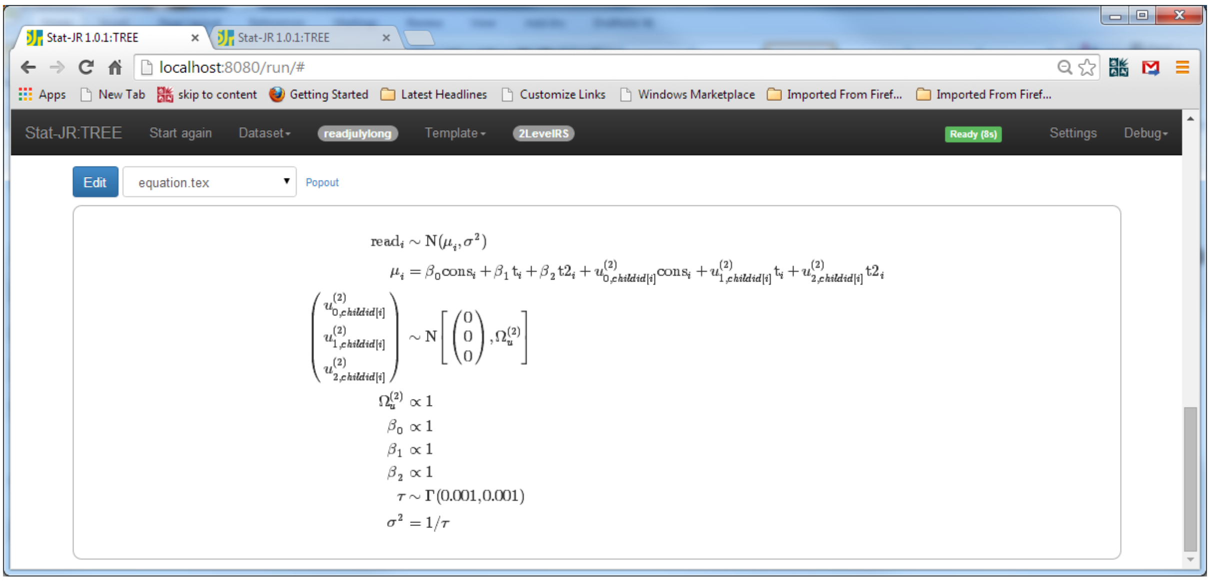
Statistic	Value
Dbar	1243.02451057
D(thetabar)	909.795338042
pD	333.229172527
DIC	1576.2536831

Here the DIC diagnostic takes the value 1576.3, whilst for the model without the quadratic term the DIC was 1766.1: nearly 200 greater; so the addition of the quadratic term yields a much better model. We can also see that *beta_2* takes value -0.187 (with standard error 0.017) which is highly-significant backing up the DIC difference. The linear term takes value 1.646 (with standard error 0.058) so, as one would expect, we have an increase in reading score with age but the speed of increase reduces (due to the negative squared term) as children get older. With this model we have, of course, forced the same quadratic coefficient for each child (by including *t2* as only a fixed effect) so the obvious next model to try is to consider making the quadratic term different for each child. To do this click on the **Start again** button and fill in the inputs as follows[2]:

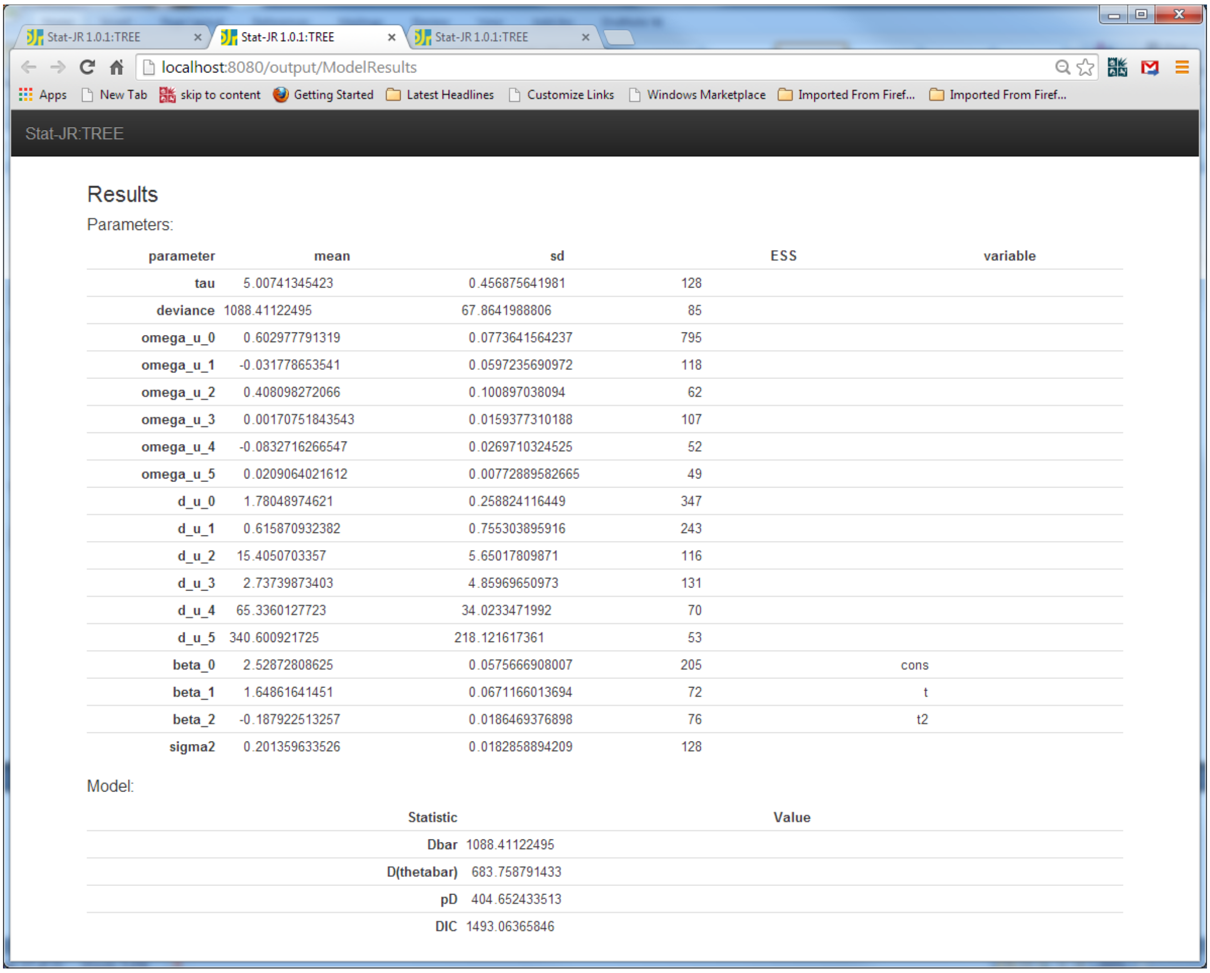
[2] Alternatively, you can copy the **Input string** – a line of text appearing midway down the browser window: you need the section of it between, and including, the curly brackets. If you then press **Start again** (in the black bar at the top), and paste it in the **Input String** box towards the bottom of the window, and then change *'x2': 'cons,t'* to *'x2': 'cons,t,t2'* before pressing the **Set** button, you'll set-up the same inputs as before, except that *t2* is now allowed to randomly-vary at level 2.



Clicking on **Next** should give the following model:



Again we need to click on **Run** to fit the model and the results will be available as *ModelResults* in the right-hand list. As this model has lots of parameters we have displayed *ModelResults* in its own tab by popping it out.



Results

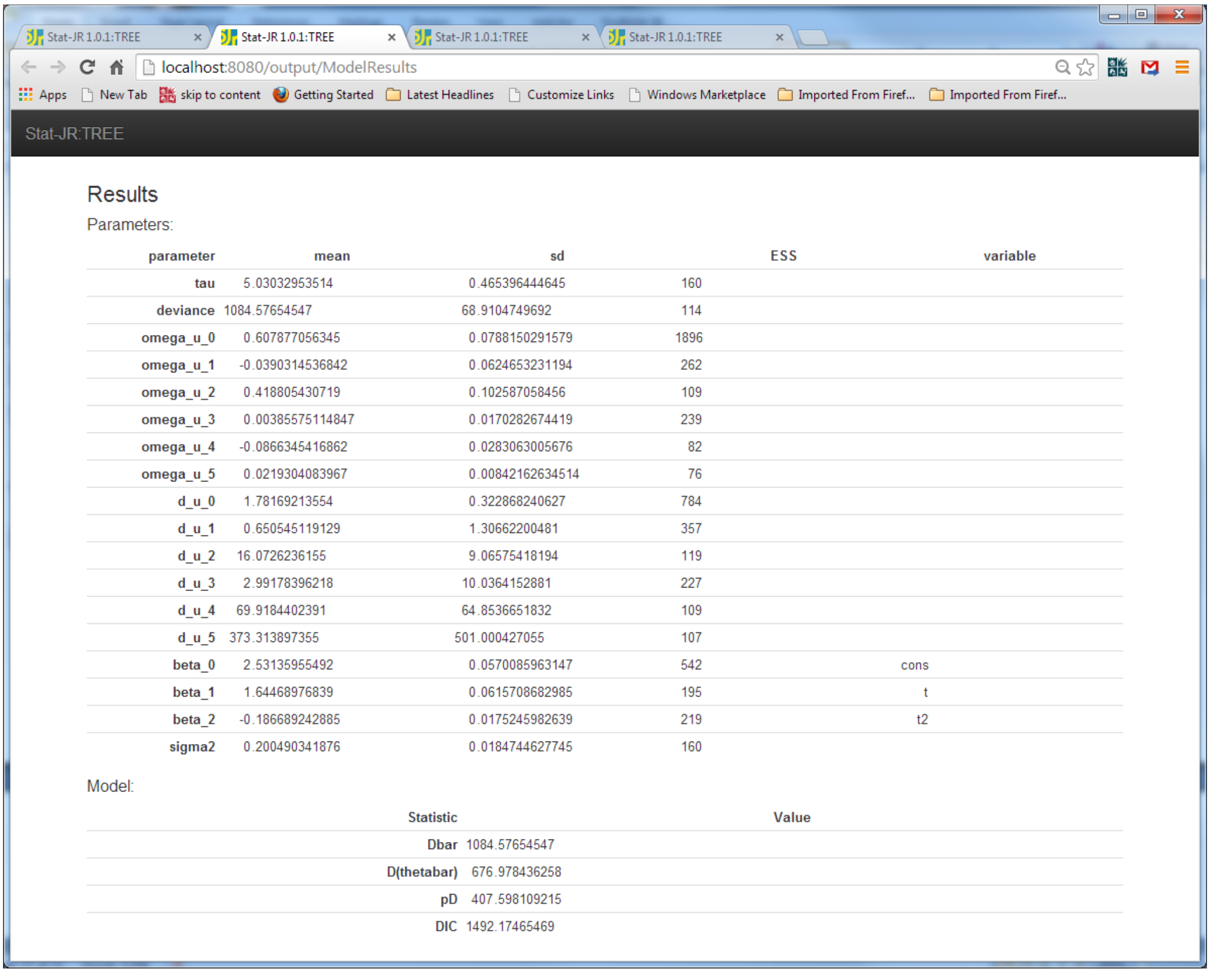
Parameters:

parameter	mean	sd	ESS	variable
tau	5.00741345423	0.456875641981	128	
deviance	1088.41122495	67.8641988806	85	
omega_u_0	0.602977791319	0.0773641564237	795	
omega_u_1	-0.031778653541	0.0597235690972	118	
omega_u_2	0.408098272066	0.100897038094	62	
omega_u_3	0.00170751843543	0.0159377310188	107	
omega_u_4	-0.0832716266547	0.0269710324525	52	
omega_u_5	0.0209064021612	0.00772889582665	49	
d_u_0	1.78048974621	0.258824116449	347	
d_u_1	0.615870932382	0.755303895916	243	
d_u_2	15.4050703357	5.65017809871	116	
d_u_3	2.73739873403	4.85969650973	131	
d_u_4	65.3360127723	34.0233471992	70	
d_u_5	340.600921725	218.121617361	53	
beta_0	2.52872808625	0.0575666908007	205	cons
beta_1	1.64861641451	0.0671166013694	72	t
beta_2	-0.187922513257	0.0186469376898	76	t2
sigma2	0.201359633526	0.0182858894209	128	

Model:

Statistic	Value
Dbar	1088.41122495
D(thetabar)	683.758791433
pD	404.652433513
DIC	1493.06365846

One thing to notice with this model is that the ESS values are quite low. To improve this we can run for an extra 3000 iterations (per chain) by typing 3000 into the **Extra Iterations** box on the main Stat-JR tab and clicking on the **More** button. On doing so we get the following results:



Results

Parameters:

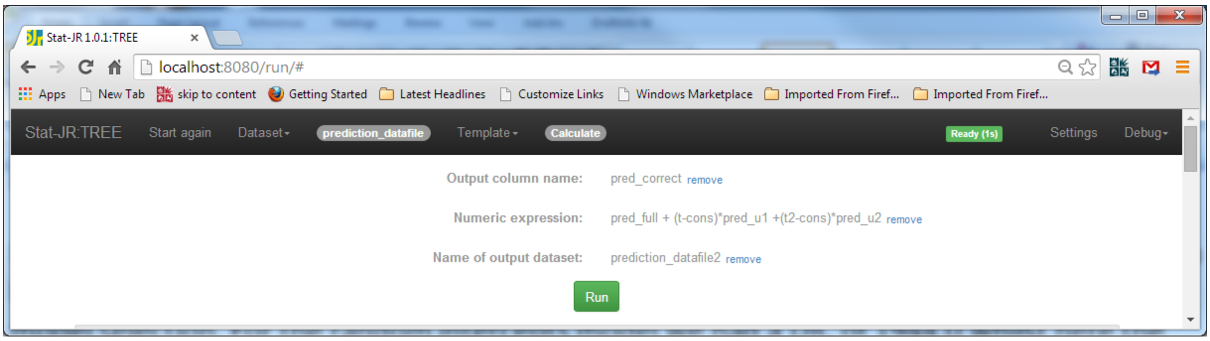
parameter	mean	sd	ESS	variable
tau	5.03032953514	0.465396444645	160	
deviance	1084.57654547	68.9104749692	114	
omega_u_0	0.607877056345	0.0788150291579	1896	
omega_u_1	-0.0390314536842	0.0624653231194	262	
omega_u_2	0.418805430719	0.102587058456	109	
omega_u_3	0.00385575114847	0.0170282674419	239	
omega_u_4	-0.0866345416862	0.0283063005676	82	
omega_u_5	0.0219304083967	0.00842162634514	76	
d_u_0	1.78169213554	0.322868240627	784	
d_u_1	0.650545119129	1.30662200481	357	
d_u_2	16.0726236155	9.06575418194	119	
d_u_3	2.99178396218	10.0364152881	227	
d_u_4	69.9184402391	64.8536651832	109	
d_u_5	373.313897355	501.000427055	107	
beta_0	2.53135955492	0.0570085963147	542	cons
beta_1	1.64468976839	0.0615708682985	195	t
beta_2	-0.186689242885	0.0175245982639	219	t2
sigma2	0.200490341876	0.0184744627745	160	

Model:

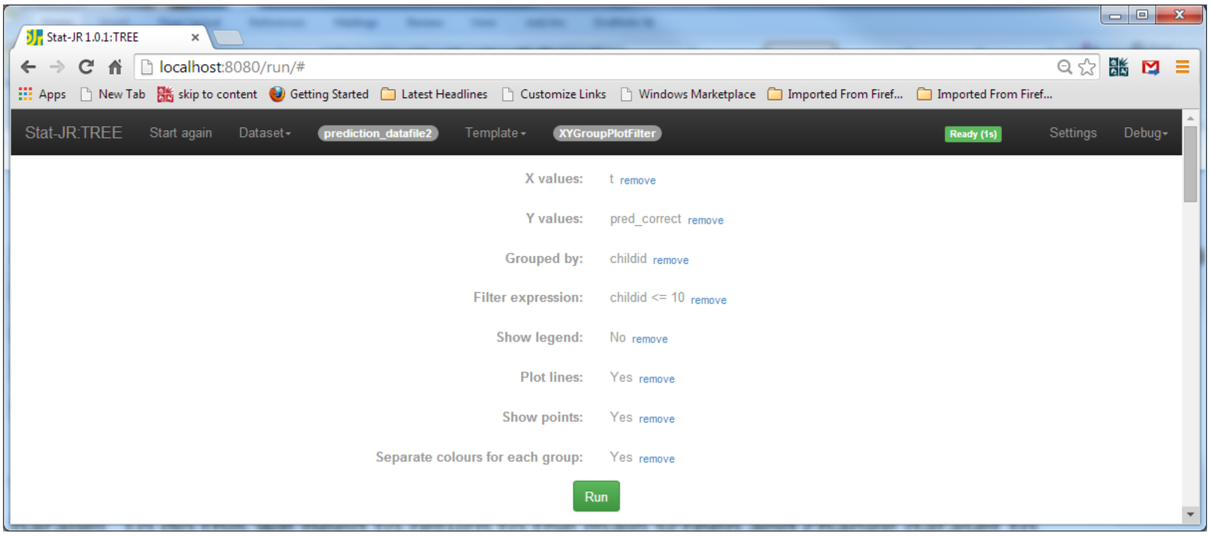
Statistic	Value
Dbar	1084.57654547
D(thetabar)	676.978436258
pD	407.598109215
DIC	1492.17465469

This improves the ESS a little, and the DIC for this model is 1492.2 (as compared with 1576.3 for just a fixed effect for *t2*).

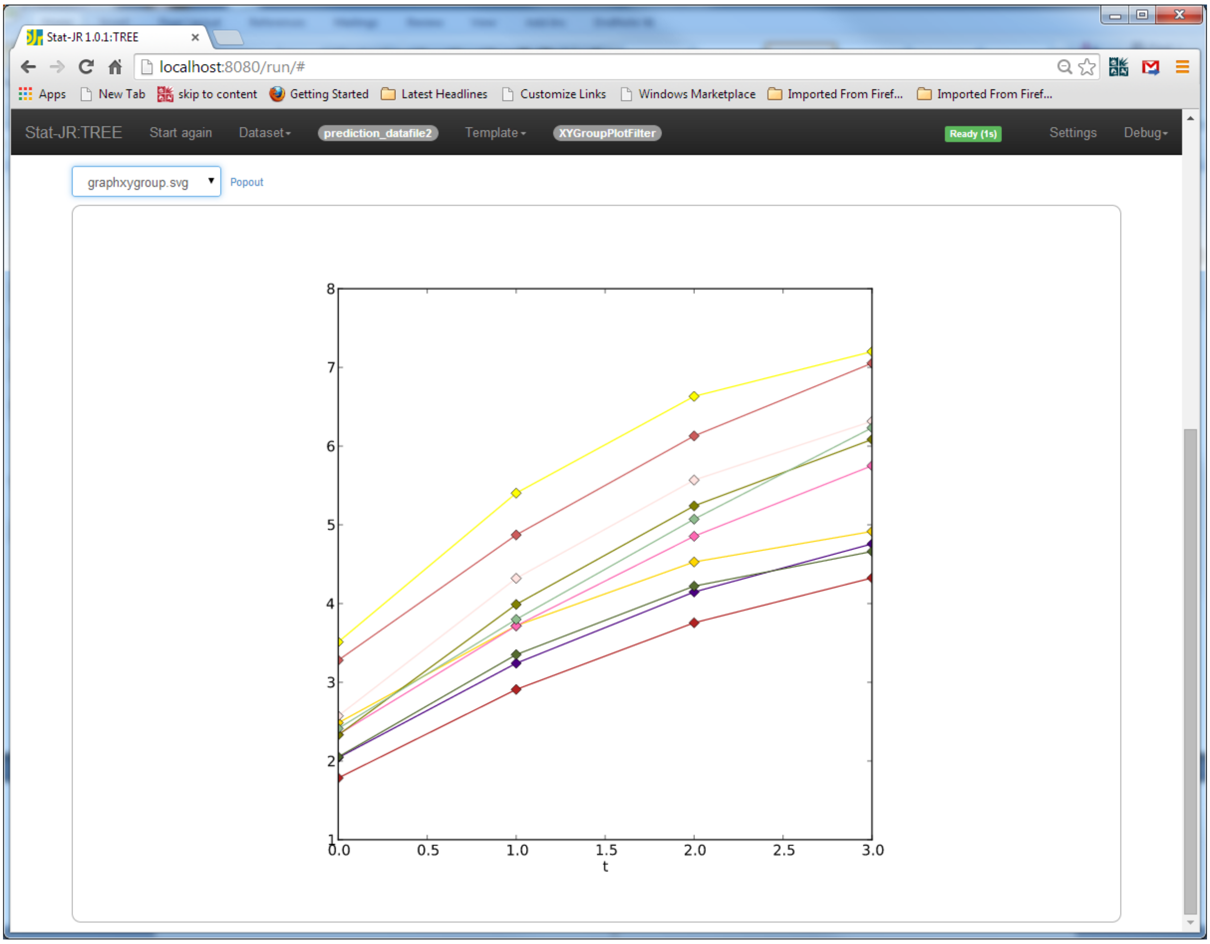
It is therefore sensible to allow children to have their own quadratic term. So the next question is: what do the predicted curves for the children's reading look like? As in Practical 1, we will need to modify the predictions obtained by default from Stat-JR to correctly include random slopes. To do this we select *Calculate* from the template list, and *prediction_datafile* from the dataset list, clicking **Use** after selecting each. After pressing **Run** we enter the following inputs (**Output column name:** *pred_correct*; **Numeric expression:** *pred_full + (t-cons)*pred_u1 + (t2-cons)*pred_u2*; **Name of output dataset:** *prediction_datafile2*):



Clicking **Run** will add the variable *pred_correct* to form the dataset *prediction_datafile2*. We can then create the graphs for the first 10 children by returning to the main window and selecting *XYGroupPlotFilter* as the template and *prediction_datafile2* as the dataset. Then, click on **Run,** and choose the inputs as shown below:



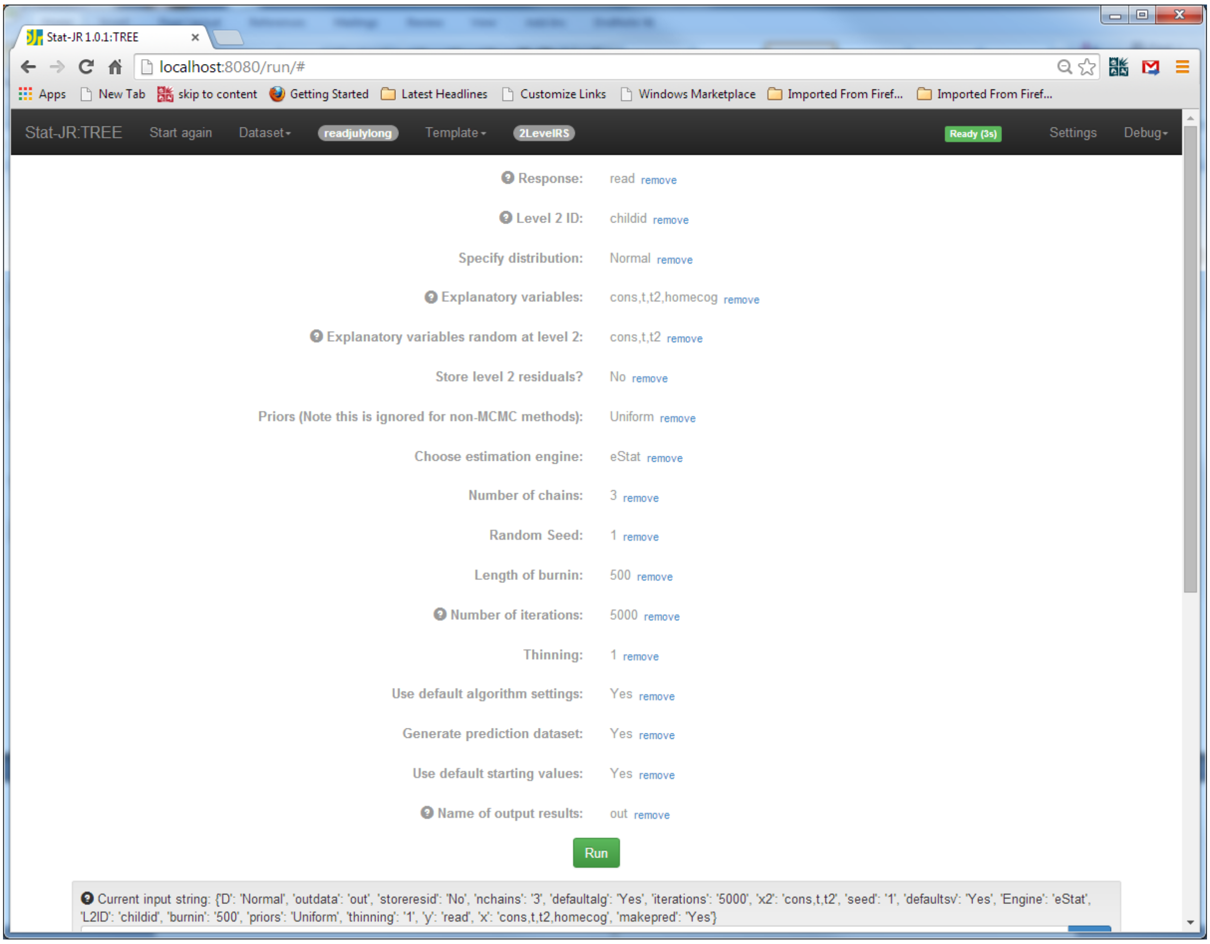
Clicking on **Run** and selecting *graphxygroup.svg* gives the graph below:



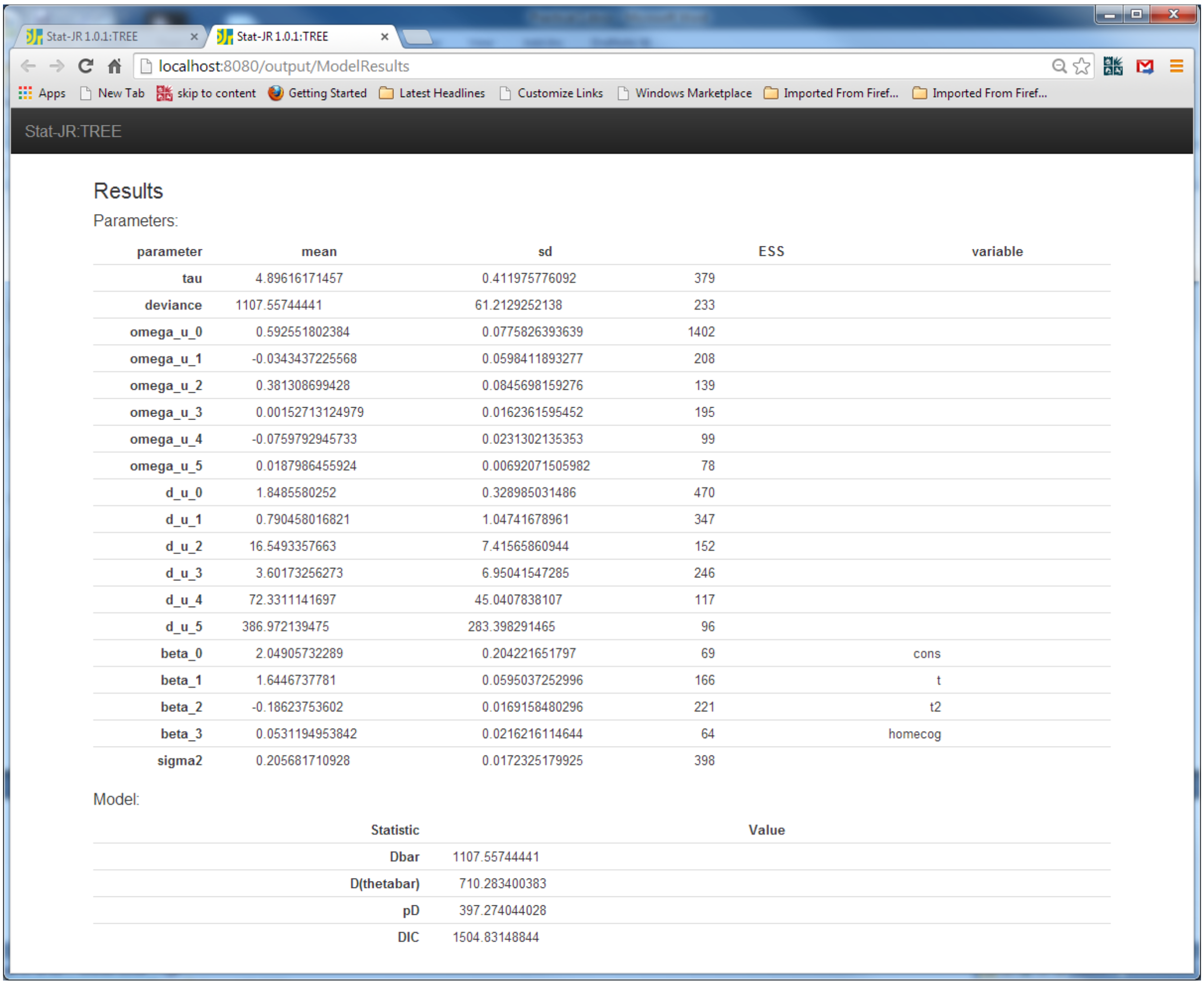
Here we see that the graphs curve and indeed the rate of increase reduces with age as reading scores begin to plateau off. We will next consider the covariate *homecog* which represents the amount of support given to the children at home.

Introducing covariates

It is straightforward to add covariates to a growth curve model. We begin by exploring whether a child's reading score is associated with the amount of cognitive support received at home (*homecog*). First we test whether *homecog* predicts the level of reading (i.e. the intercept) by simply including it as a fixed effect. To do this we return to the template *2LevelRS* and the dataset *readjulylong.* We set up the inputs as before but with *homecog* added to the explanatory variables list:



Note here that we have increased the number of iterations to 5,000 per chain after seeing the poor mixing in the previous model. If we click on **Run** and go straight to the *ModelResults* (popping them out) upon the model finishing we see the following:



Stat-JR:TREE

Results

Parameters:

parameter	mean	sd	ESS	variable
tau	4.89616171457	0.411975776092	379	
deviance	1107.55744441	61.2129252138	233	
omega_u_0	0.592551802384	0.0775826393639	1402	
omega_u_1	-0.0343437225568	0.0598411893277	208	
omega_u_2	0.381308699428	0.0845698159276	139	
omega_u_3	0.00152713124979	0.0162361595452	195	
omega_u_4	-0.0759792945733	0.0231302135353	99	
omega_u_5	0.0187986455924	0.00692071505982	78	
d_u_0	1.8485580252	0.328985031486	470	
d_u_1	0.790458016821	1.04741678961	347	
d_u_2	16.5493357663	7.41565860944	152	
d_u_3	3.60173256273	6.95041547285	246	
d_u_4	72.3311141697	45.0407838107	117	
d_u_5	386.972139475	283.398291465	96	
beta_0	2.04905732289	0.204221651797	69	cons
beta_1	1.6446737781	0.0595037252996	166	t
beta_2	-0.18623753602	0.0169158480296	221	t2
beta_3	0.0531194953842	0.0216216114644	64	homecog
sigma2	0.205681710928	0.0172325179925	398	

Model:

Statistic	Value
Dbar	1107.55744441
D(thetabar)	710.283400383
pD	397.274044028
DIC	1504.83148844

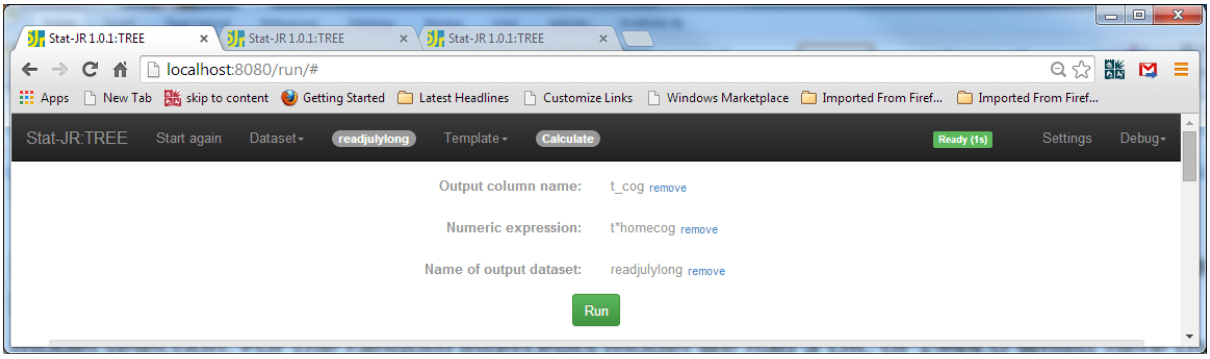
Here we see that *beta_3* (the *homecog* effect) is just significant with an effect of 0.053 (and standard error 0.022) although we have rather a low ESS, and the DIC diagnostic which, at 1504.83, is bigger than the model without *homecog* (which had a DIC of 1492.2). Given the ESS is small (64) we could run for longer, and in fact running for 25,000 per chain gives *beta_3* = 0.050 (with standard error 0.023) which is still just significant though with DIC still at 1501.0. The effect is positive, as one might expect: i.e. more cognitive support at home is associated with higher reading test scores.

Question: Does the addition of *homecog* (a child-level variable) explain much of the between-child variance? (Hint: *omega_u_0* is the child-level intercept variance).

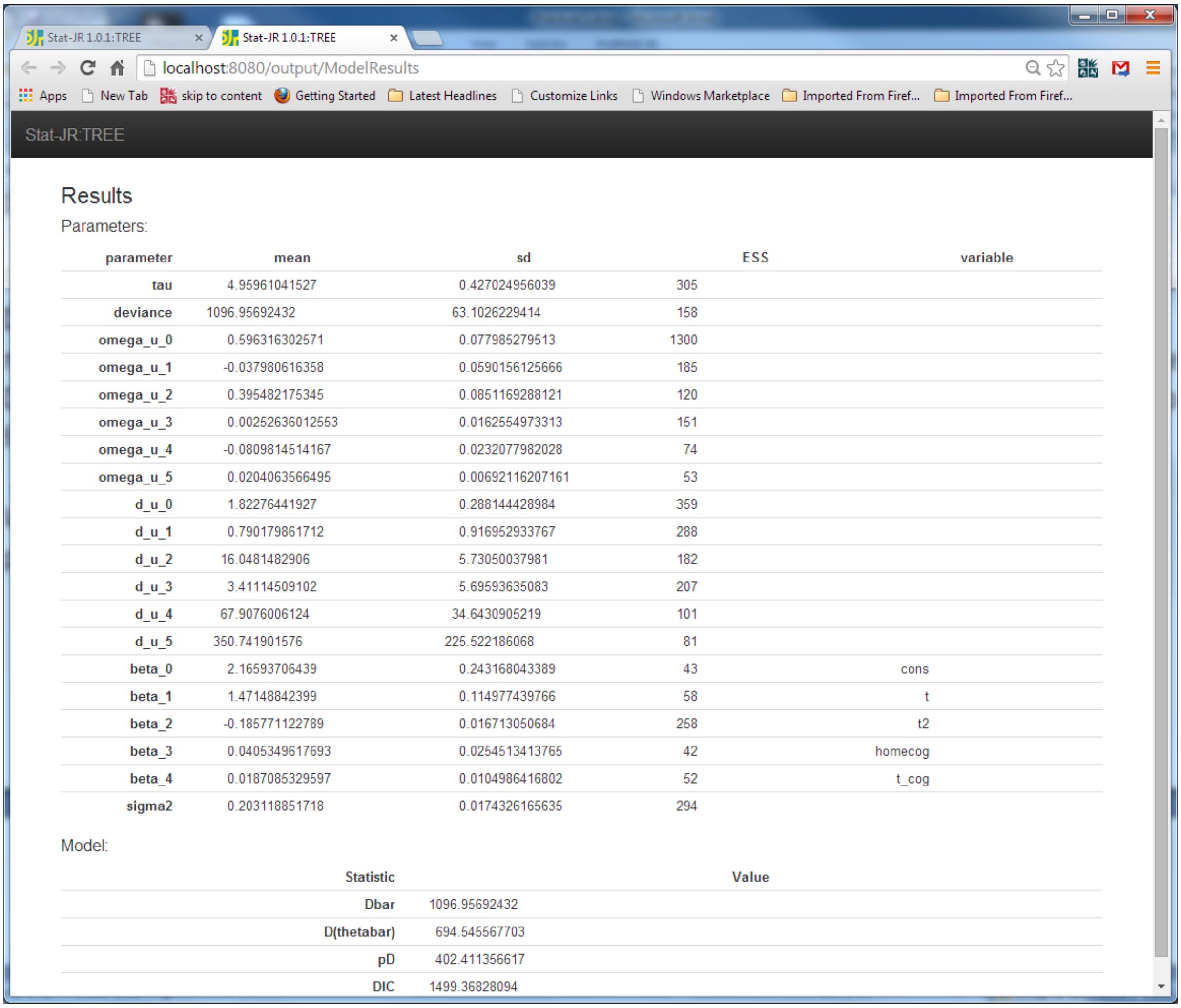
The reason the DIC is not improved (and in fact is worse) with the addition of this predictor is an interesting one. The DIC diagnostic has a ‘focus’ which is where in the model it measures model fit. In this case the ‘focus’ is at level 1 and although *homecog* is a significant predictor it is explaining variation at level 2 which is not the ‘focus’ of DIC – in other words the variation it explained at level 1 had already been explained by the child random effects in the simpler model. This is not to say that including it doesn’t improve the model but simply that the DIC diagnostic is not so useful at establishing the importance of higher level predictors.

We could also look at whether home cognitive support affects reading progress. To do this we will need to include an interaction between the *homecog* variable and time. Return to the main window and select *Calculate* from the template list and click **Use** and **Run**.

To construct the interaction the template inputs should be as follows (**Output column name:** *t_cog*; **Numeric expression:** *t*homecog*; **Name of output dataset:** *readjulylong*):



Clicking on **Run** will construct the interaction variable and name it *t_cog*. We will next return to the template *2LevelRS* and add *t_cog* to the explanatory variable list. If you do this (whilst repeating the other inputs from the last model fit (see page 9)) we see in the *ModelResults*, below, that now, both the main effect for *homecog* (*beta_3*), and the interaction (*beta_4*) are borderline significant.

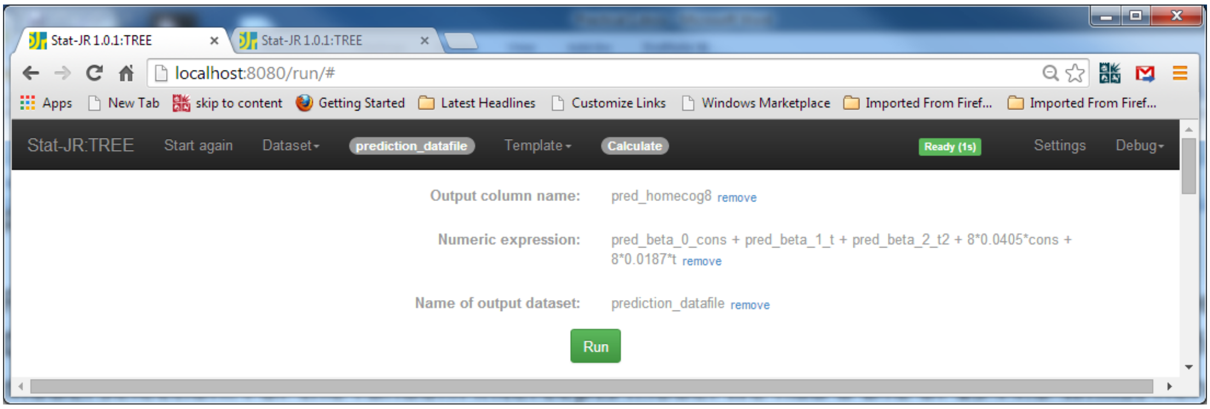


parameter	mean	sd	ESS	variable
tau	4.95961041527	0.427024956039	305	
deviance	1096.95692432	63.1026229414	158	
omega_u_0	0.596316302571	0.077985279513	1300	
omega_u_1	-0.037980616358	0.0590156125666	185	
omega_u_2	0.395482175345	0.0851169288121	120	
omega_u_3	0.00252636012553	0.0162554973313	151	
omega_u_4	-0.0809814514167	0.0232077982028	74	
omega_u_5	0.0204063566495	0.00692116207161	53	
d_u_0	1.82276441927	0.288144428984	359	
d_u_1	0.790179861712	0.916952933767	288	
d_u_2	16.0481482906	5.73050037981	182	
d_u_3	3.41114509102	5.69593635083	207	
d_u_4	67.9076006124	34.6430905219	101	
d_u_5	350.741901576	225.522186068	81	
beta_0	2.16593706439	0.243168043389	43	cons
beta_1	1.47148842399	0.114977439766	58	t
beta_2	-0.185771122789	0.016713050684	258	t2
beta_3	0.0405349617693	0.0254513413765	42	homecog
beta_4	0.0187085329597	0.0104986416802	52	t_cog
sigma2	0.203118851718	0.0174326165635	294	

Statistic	Value
Dbar	1096.95692432
D(thetabar)	694.545567703
pD	402.411356617
DIC	1499.36828094

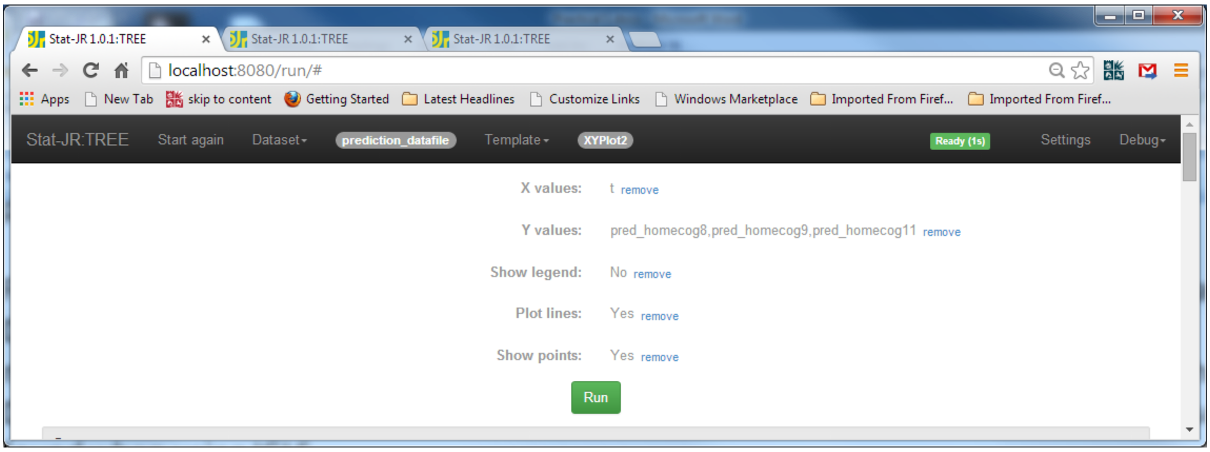
One could ask whether the addition of the interaction between *t* and *homecog* has explained much of the between-child variance in the effect of *t* (i.e. variances associated with *t* (*omega_u_2*) and *t2* (*omega_u_5*)).

We might like to plot the average predicted curves for various levels of the *homecog* variable. In the lecture we did this for values of *homecog* equal to 8, 9 and 11 as these represent the quartiles of the data. We can construct these predictions as follows. Firstly select *Calculate* from the template list and click **Use** and then select *prediction_datafile* from the dataset list and click **Use**. Clicking on **Run** we can construct the prediction for the lower quartile as follows (**Output column name:** *pred_homecog8*; **Numeric expression:** *pred_beta_0_cons + pred_beta_1_t + pred_beta_2_t2 + 8*0.0405*cons + 8*0.0187*t*; **Name of output dataset:** *prediction_datafile*):

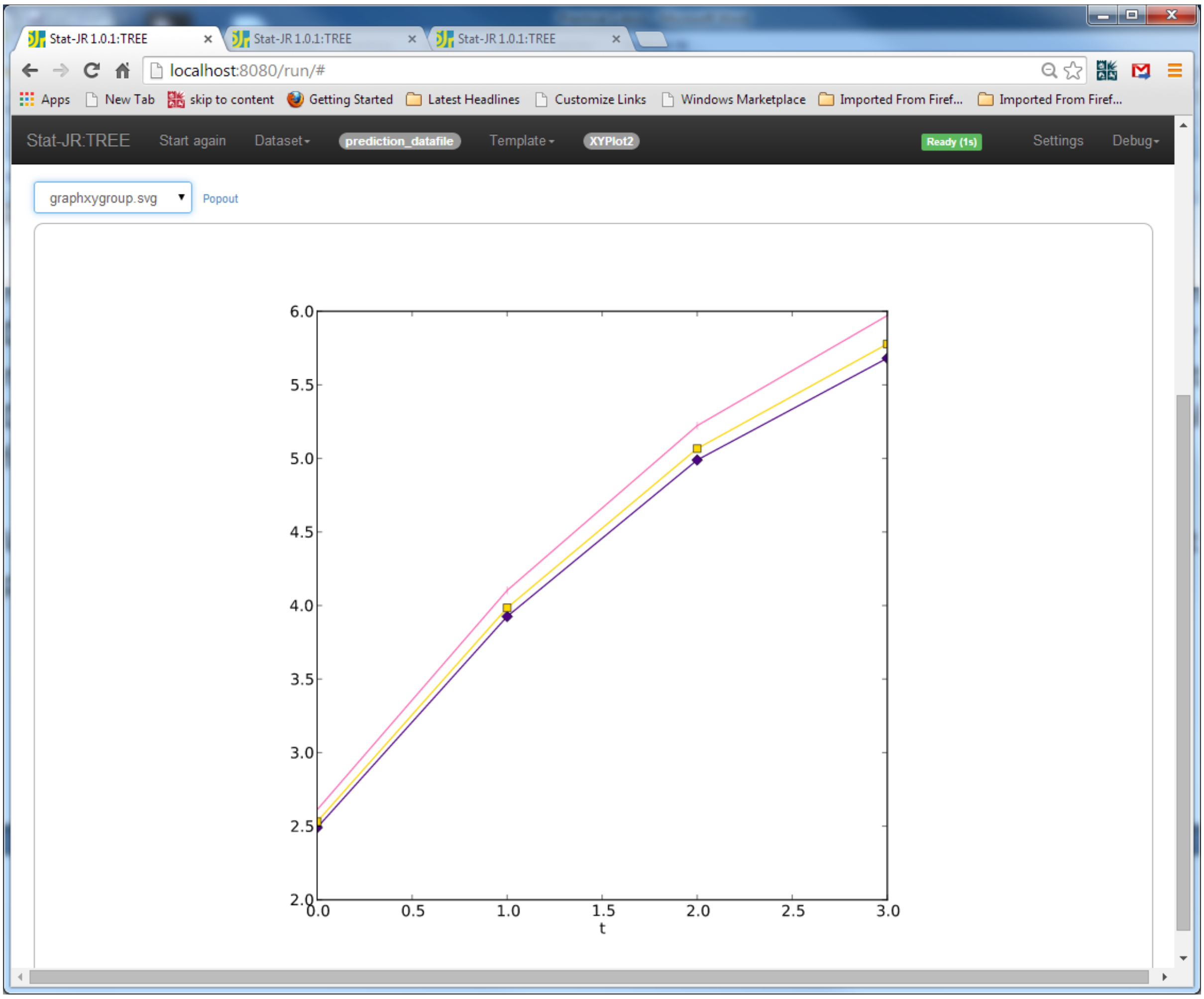


Clicking on **Run** will construct this extra prediction variable and add it to the *prediction_datafile*. To construct the predictors for the median and upper quartile we simply change the name of the output column, and also replace the two 8s in the numeric expression with 9s for the median and 11s for the upper quartile. Each time we add the variable to *prediction_datafile* (i.e. **Output column name:** *pred_homecog**9***; **Numeric expression:** *pred_beta_0_cons + pred_beta_1_t + pred_beta_2_t2 + **9***0.0405*cons + **9***0.0187*t*; **Name of output dataset:** *prediction_datafile*; etc.).

Finally we will plot these 3 curves by selecting the template *XYPlot2* and choosing inputs as shown:



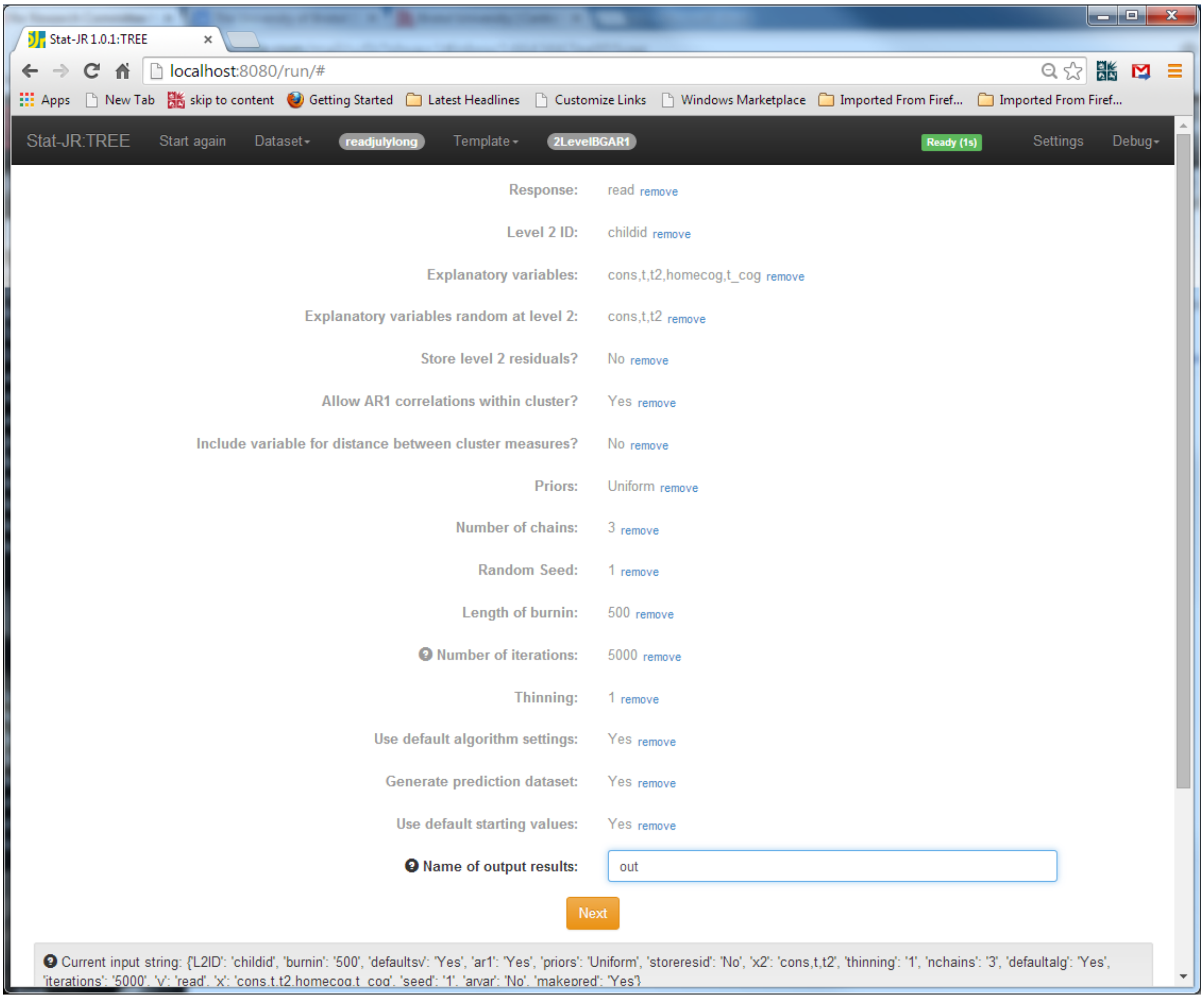
And we get the graphs thus:



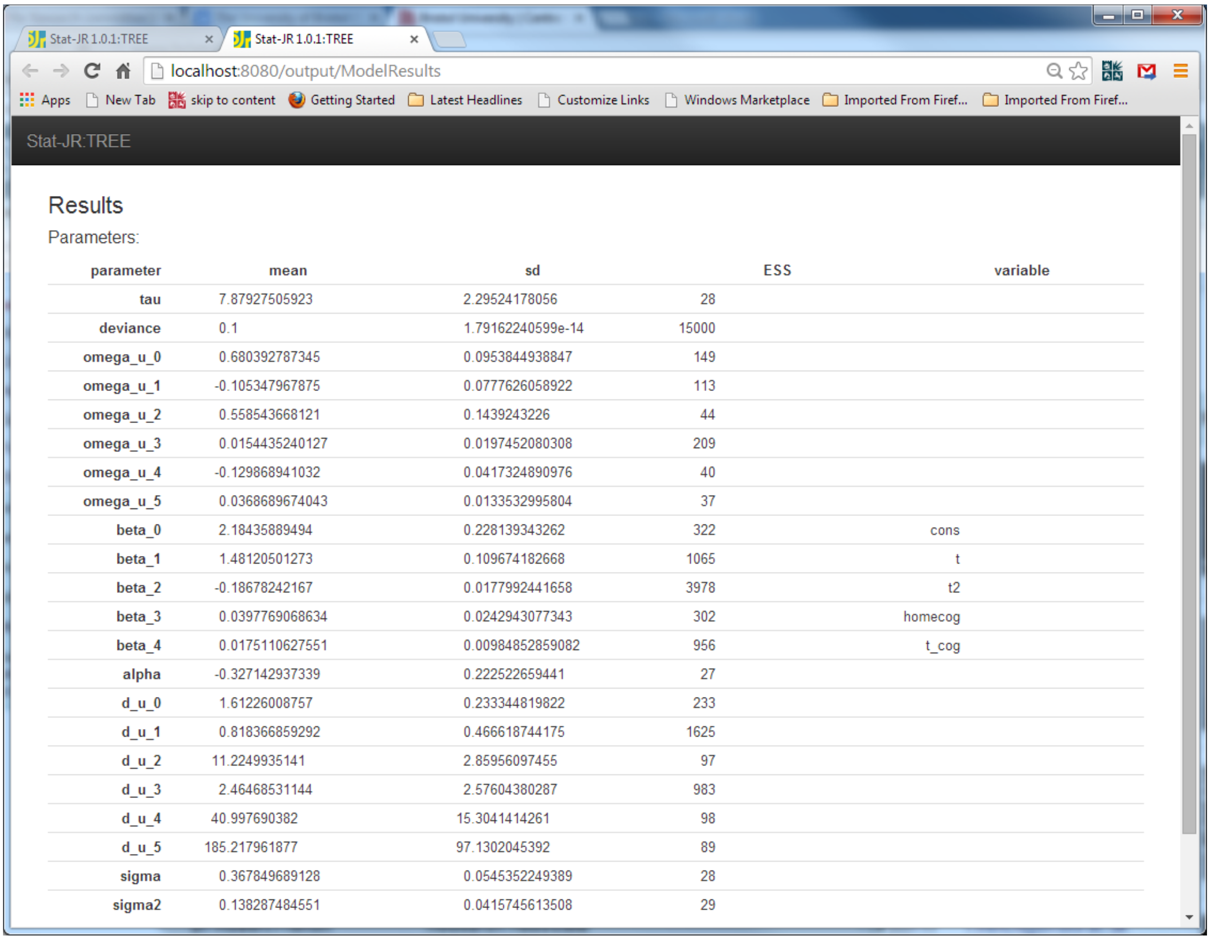
Autocorrelated residuals

So far we have assumed that the occasion-level residuals are independent: i.e. we have assumed that the correlation between a child's responses over time is explained by unmeasured time-invariant child characteristics (represented in the model by the child-level random effects). We now relax this assumption and assume an AR(1) covariance structure at the occasion-level. This is fairly recent work in MCMC and we are following the algorithms given in Browne and Goldstein (2010) so will use a template named *2LevelBGAR1* (where BG stands for Browne and Goldstein). Although this template gives estimates, it doesn't, as yet, give a DIC diagnostic for model comparison, nor LaTeX code for the model structure.

So, to begin, select *2LevelBGAR1* from the template list, and *readjulylong* from the dataset list. Then, after clicking on **Use,** we need to set the inputs to the template as follows:



Note again here that we are running for 5,000 iterations. Clicking on **Next** and **Run** will fit the model. Note that the model code should be ignored as this template doesn't use the algebra system but creates its own bespoke C code. The model takes a little longer to run but the results can be seen in *ModelResults* (popped out):



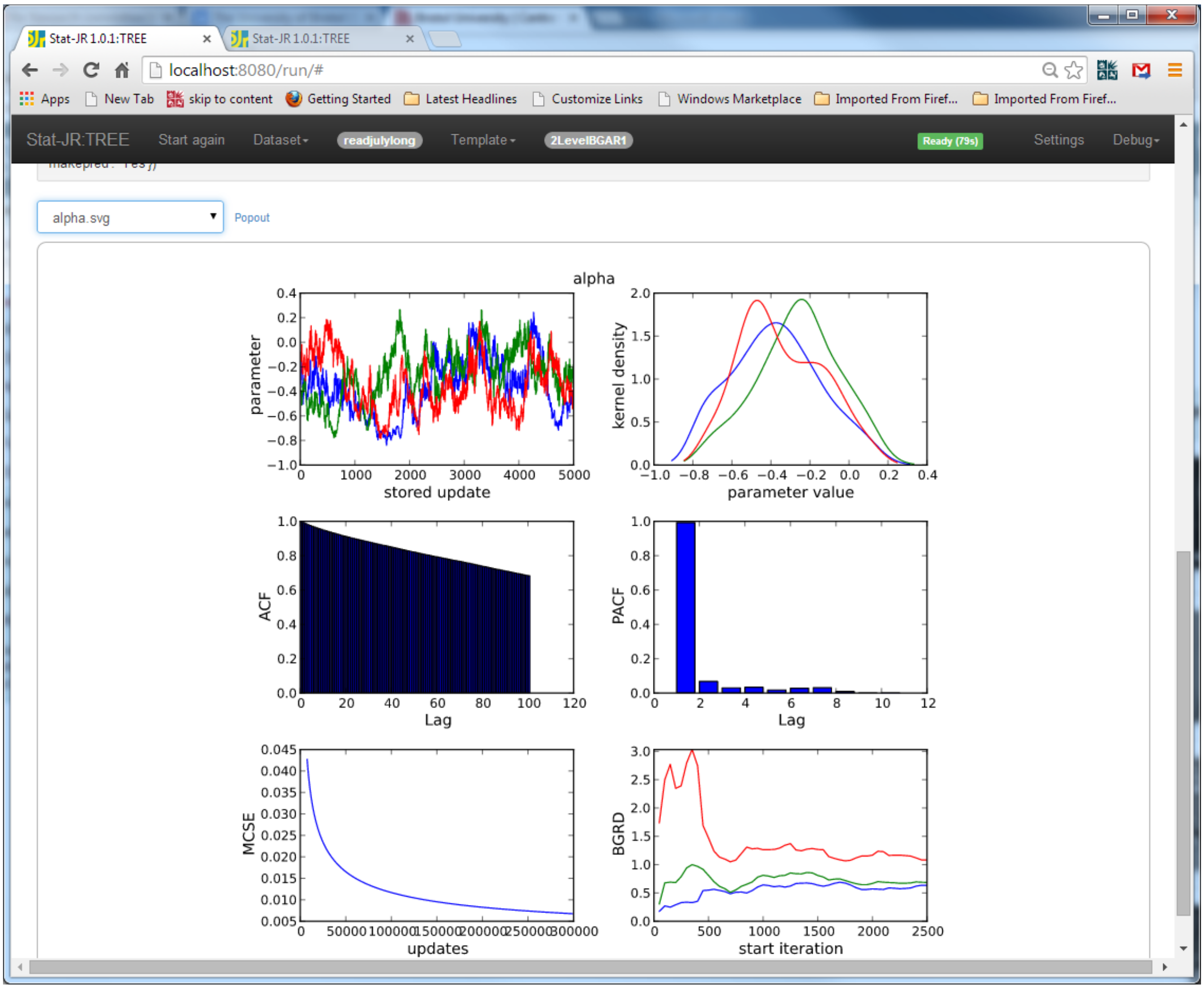
Stat-JR:TREE

Results

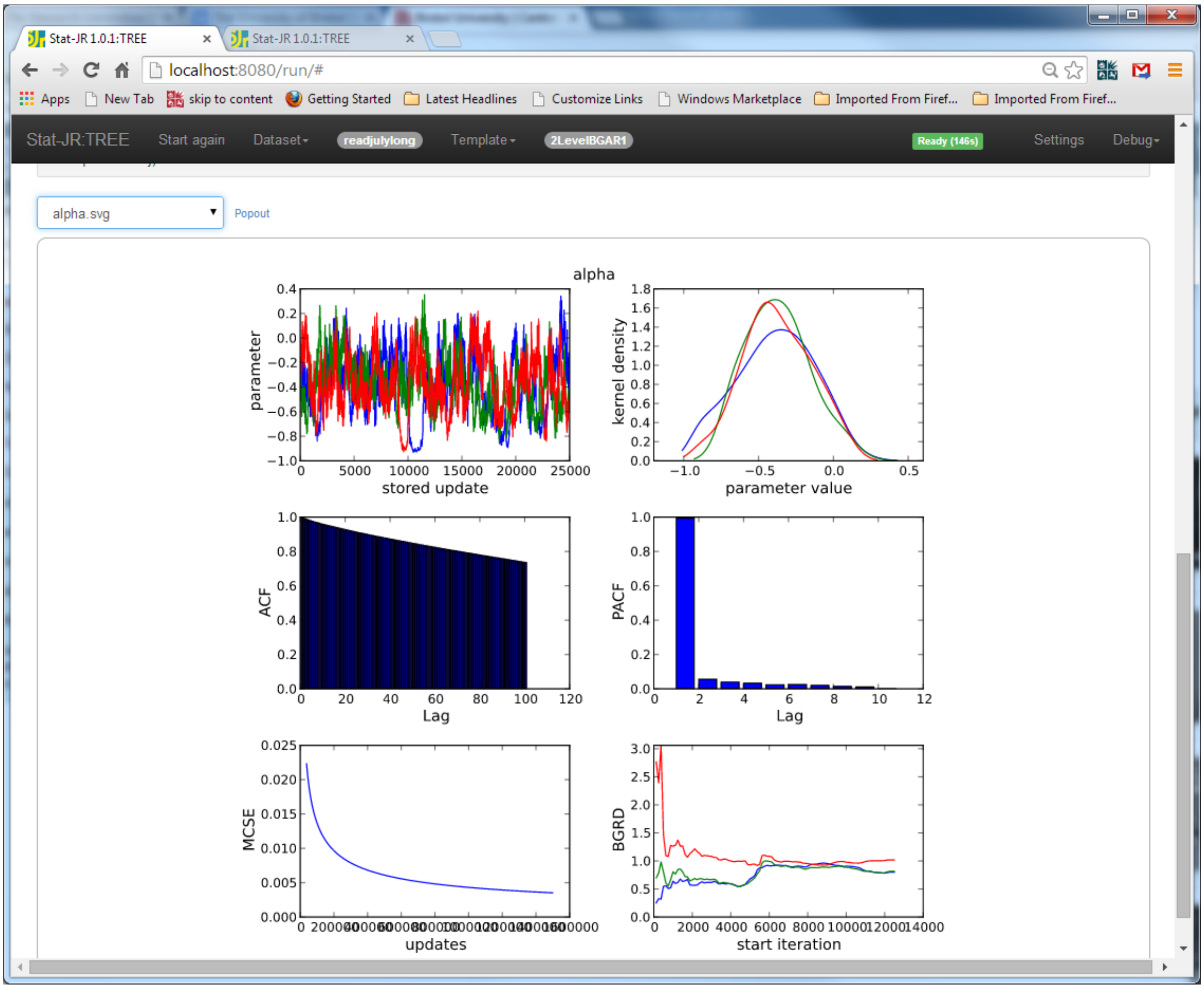
Parameters:

parameter	mean	sd	ESS	variable
tau	7.87927505923	2.29524178056	28	
deviance	0.1	1.79162240599e-14	15000	
omega_u_0	0.680392787345	0.0953844938847	149	
omega_u_1	-0.105347967875	0.0777626058922	113	
omega_u_2	0.558543668121	0.1439243226	44	
omega_u_3	0.0154435240127	0.0197452080308	209	
omega_u_4	-0.129868941032	0.0417324890976	40	
omega_u_5	0.0368689674043	0.0133532995804	37	
beta_0	2.18435889494	0.228139343262	322	cons
beta_1	1.48120501273	0.109674182668	1065	t
beta_2	-0.18678242167	0.0177992441658	3978	t2
beta_3	0.0397769068634	0.0242943077343	302	homecog
beta_4	0.0175110627551	0.00984852859082	956	t_cog
alpha	-0.327142937339	0.222522659441	27	
d_u_0	1.61226008757	0.233344819822	233	
d_u_1	0.818366859292	0.466618744175	1625	
d_u_2	11.2249935141	2.85956097455	97	
d_u_3	2.46468531144	2.57604380287	983	
d_u_4	40.997690382	15.3041414261	98	
d_u_5	185.217961877	97.1302045392	89	
sigma	0.367849689128	0.0545352249389	28	
sigma2	0.138287484551	0.0415745613508	29	

The parameter *alpha* represents the AR1 correlation and takes value -0.327, with standard error 0.222: so not significant, although the ESS is only 27 which suggests we need to run for longer. The software package Stata can fit this same model using the *xtmixed* command: this estimates the correlation as -0.291 with, again, a large standard error. If we look here at the diagnostics for *alpha*, by selecting *alpha.svg*, from the object list, we see:



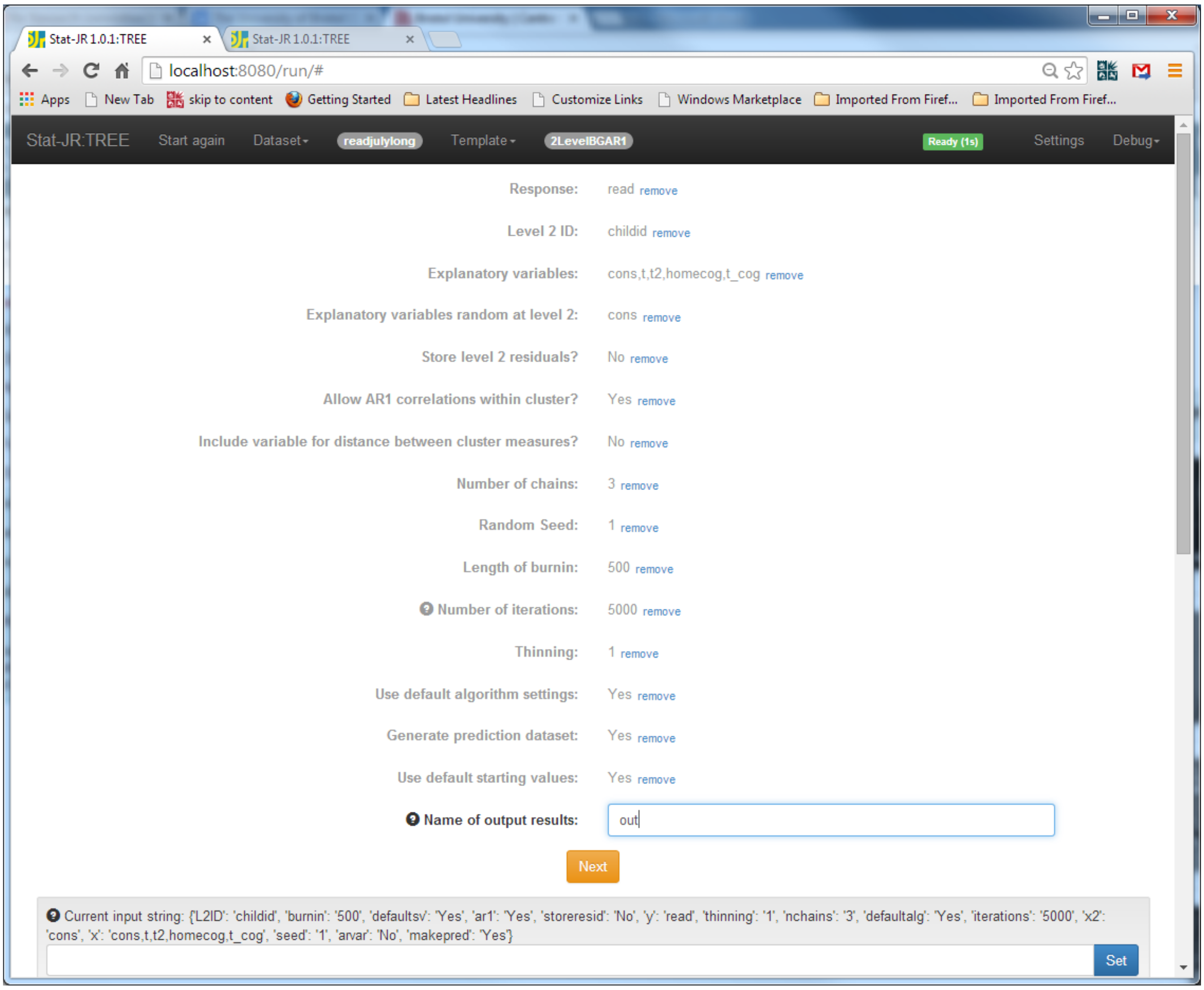
Here we can see the chains are not mixing well but that the estimate of -0.291 from Stata is close to the modes we see for the blue and green chains. In fact if we were to run for a further 20,000 iterations per chain (don't do this yourself as it will take too long!) we see the following:



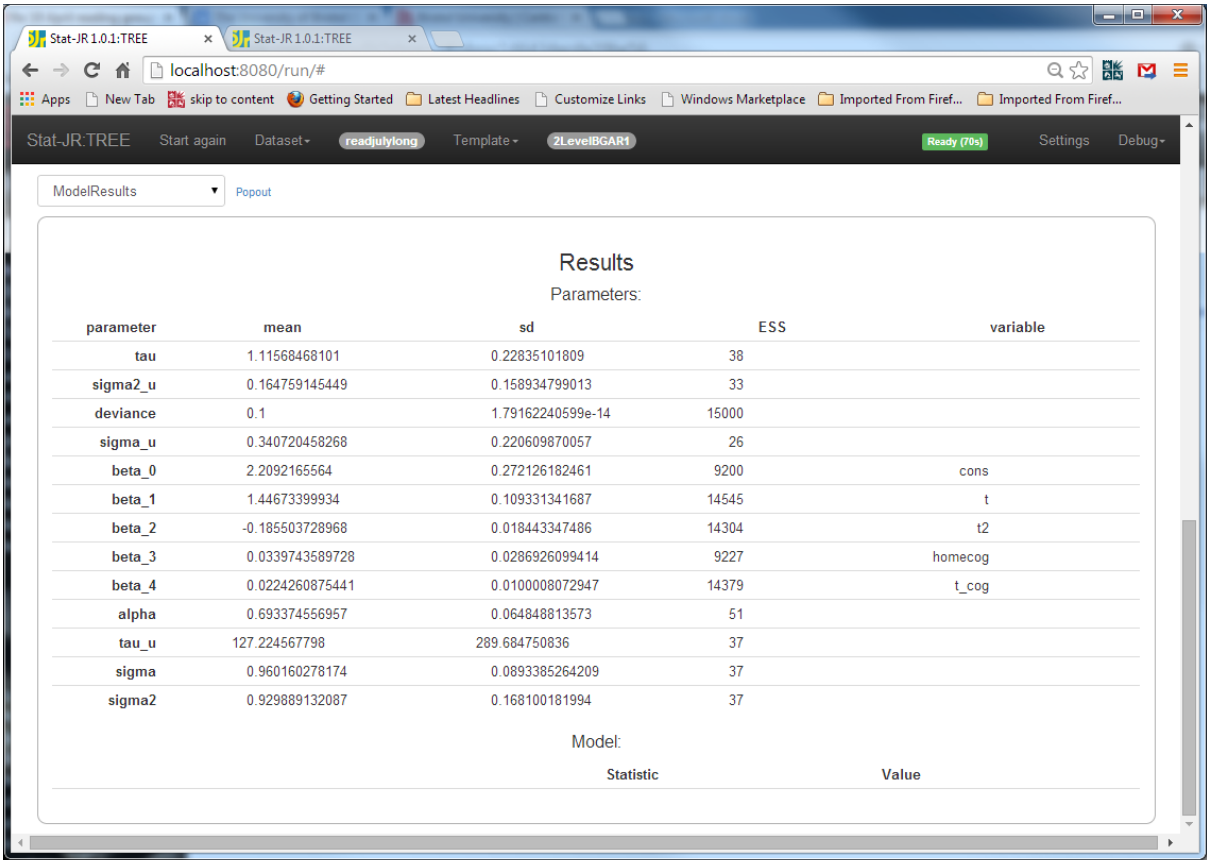
Here the posterior mean estimate is -0.377, with standard error 0.236, which is not quite the Stata estimate, but both methods agree that the correlation is not significant. This may be because, by fitting quadratic terms for time for each child, we have explained much of the autocorrelation in their readings.

To test this we will next fit a simpler growth curve model with a quadratic curve in the fixed part, but only permitting the intercept to vary across children.

Using the same template (*2LevelBGAR1*) and dataset (*readjulylong*) as before, we enter our inputs as follows:



Clicking **Next** and **Run** we get the estimates that you see below, under *ModelResults*:



Results

Parameters:

parameter	mean	sd	ESS	variable
tau	1.11568468101	0.22835101809	38	
sigma2_u	0.164759145449	0.158934799013	33	
deviance	0.1	1.79162240599e-14	15000	
sigma_u	0.340720458268	0.220609870057	26	
beta_0	2.2092165564	0.272126182461	9200	cons
beta_1	1.44673399934	0.109331341687	14545	t
beta_2	-0.185503728968	0.018443347486	14304	t2
beta_3	0.0339743589728	0.0286926099414	9227	homecog
beta_4	0.0224260875441	0.0100008072947	14379	t_cog
alpha	0.693374556957	0.064848813573	51	
tau_u	127.224567798	289.684750836	37	
sigma	0.960160278174	0.0893385264209	37	
sigma2	0.929889132087	0.168100181994	37	

Model:

Statistic	Value

Here we observe a posterior mean estimate of 0.693 (Stata's *xtmixed* gets 0.67 for this parameter) and this value is much bigger than its standard error. So, if we do not fit a realistic random effects model that better accounts for the variability between children in their reading progress, there is unexplained positive autocorrelation.

An estimate of 0.693 implies that, after allowing for an overall quadratic growth trend and effects of cognitive support at home, the residual correlation between a child's residuals on consecutive occasions (2 years apart) is 0.693. The correlation between residuals on occasions 4 years apart (i.e. times 1 and 3, or 2 and 4) decreases to $0.693^2 = 0.480$.

Further Exercises

If you have time you might investigate growth models that answer the following questions:

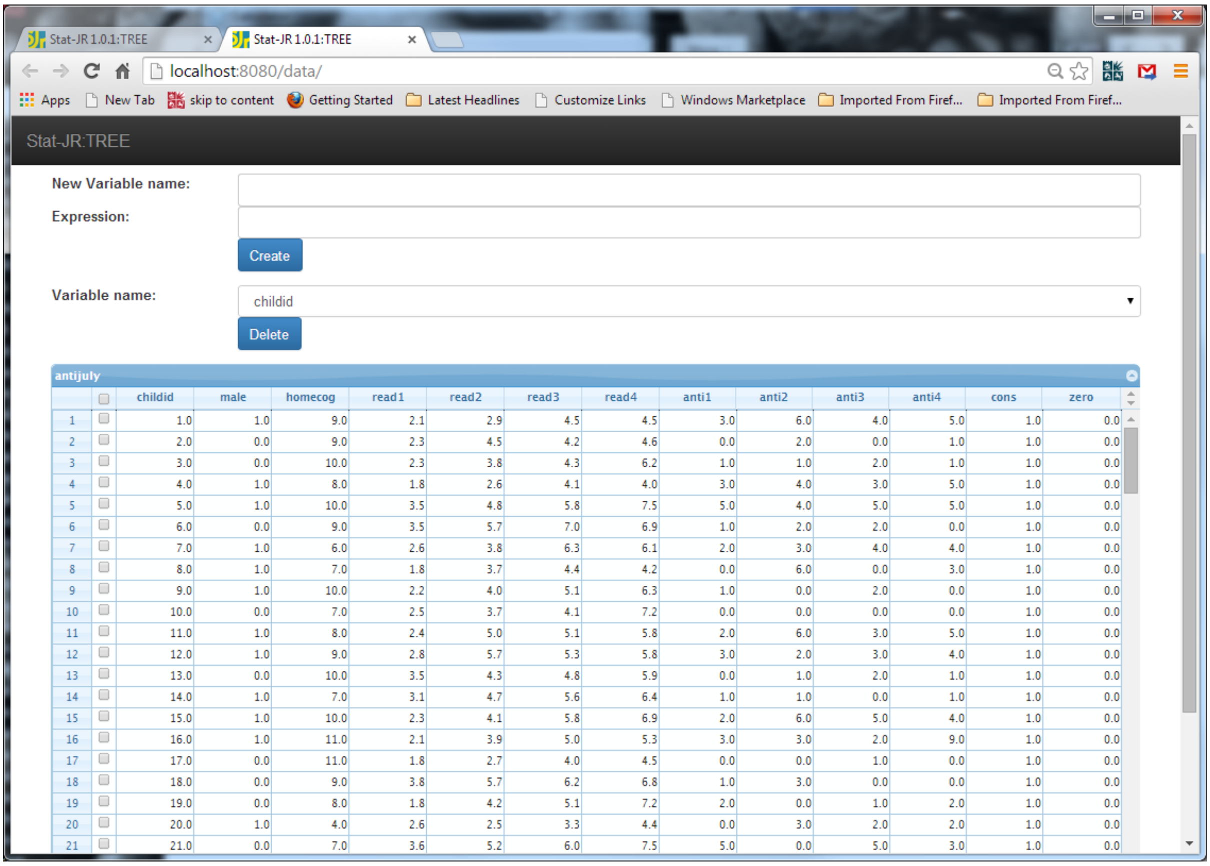
i. Is there a gender difference in the level of reading score (at any occassion)?
ii. Do boys and girls differ in the rate of reading progress?

MODELLING LONGITUDINAL DATA USING THE STAT-JR PACKAGE

Practical 3: Multivariate and Dynamic (Autoregressive) Models

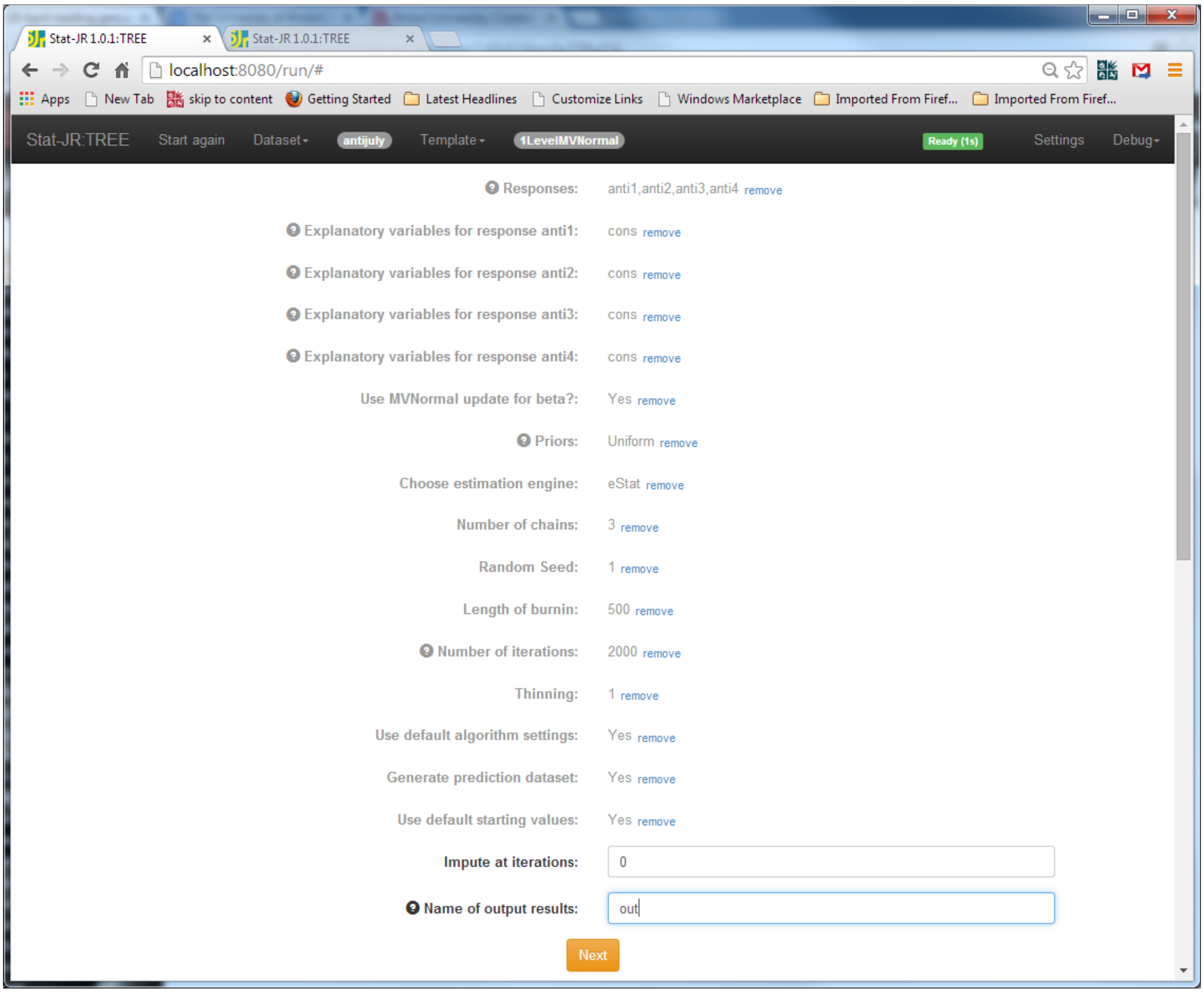
3.1 Multivariate model for antisocial behaviour

In this exercise we begin by analysing repeated measures of antisocial behaviour which are contained in the same dataset as the reading score measures. As for reading, antisocial behaviour was measured on four occasions for 221 US children. See Practical 1 for a description of the dataset. We will use a wide form of the data (with 1 column per year), saved as the dataset *antijuly*. To use this dataset select it from the list and click on the **Use** button. To view the dataset click on the **View dataset** button towards the top of the page, and it will appear thus:

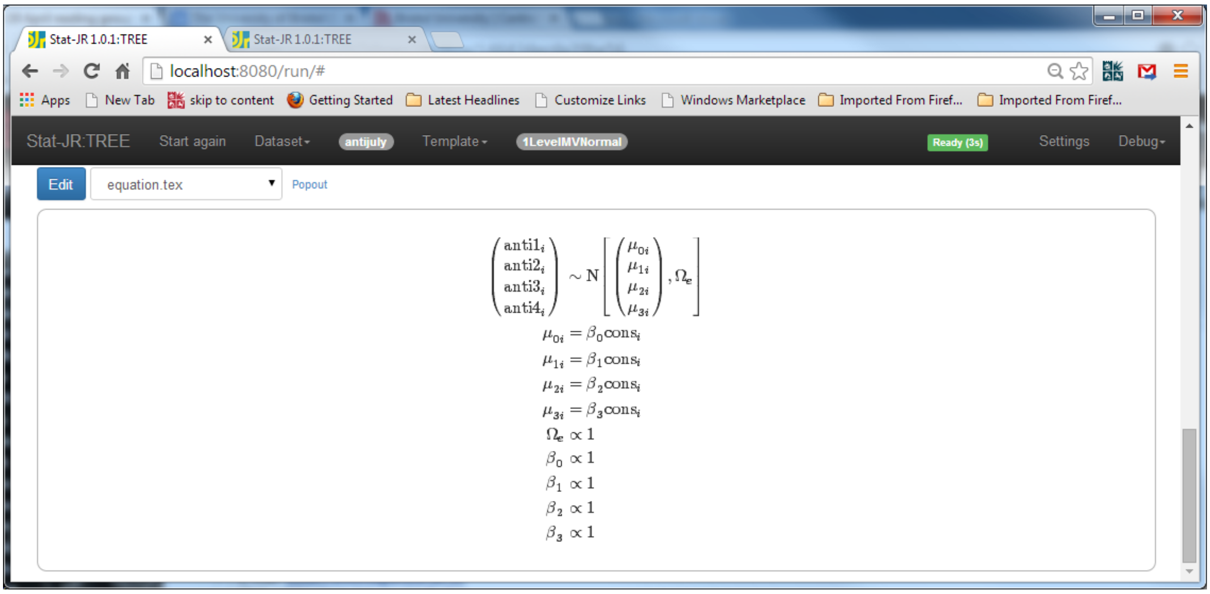


	childid	male	homecog	read1	read2	read3	read4	anti1	anti2	anti3	anti4	cons	zero
1	1.0	1.0	9.0	2.1	2.9	4.5	4.5	3.0	6.0	4.0	5.0	1.0	0.0
2	2.0	0.0	9.0	2.3	4.5	4.2	4.6	0.0	2.0	0.0	1.0	1.0	0.0
3	3.0	0.0	10.0	2.3	3.8	4.3	6.2	1.0	1.0	2.0	1.0	1.0	0.0
4	4.0	1.0	8.0	1.8	2.6	4.1	4.0	3.0	4.0	3.0	5.0	1.0	0.0
5	5.0	1.0	10.0	3.5	4.8	5.8	7.5	5.0	4.0	5.0	5.0	1.0	0.0
6	6.0	0.0	9.0	3.5	5.7	7.0	6.9	1.0	2.0	2.0	0.0	1.0	0.0
7	7.0	1.0	6.0	2.6	3.8	6.3	6.1	2.0	3.0	4.0	4.0	1.0	0.0
8	8.0	1.0	7.0	1.8	3.7	4.4	4.2	0.0	6.0	0.0	3.0	1.0	0.0
9	9.0	1.0	10.0	2.2	4.0	5.1	6.3	1.0	0.0	2.0	0.0	1.0	0.0
10	10.0	0.0	7.0	2.5	3.7	4.1	7.2	0.0	0.0	0.0	0.0	1.0	0.0
11	11.0	1.0	8.0	2.4	5.0	5.1	5.8	2.0	6.0	3.0	5.0	1.0	0.0
12	12.0	1.0	9.0	2.8	5.7	5.3	5.8	3.0	2.0	3.0	4.0	1.0	0.0
13	13.0	0.0	10.0	3.5	4.3	4.8	5.9	0.0	1.0	2.0	1.0	1.0	0.0
14	14.0	1.0	7.0	3.1	4.7	5.6	6.4	1.0	1.0	0.0	1.0	1.0	0.0
15	15.0	1.0	10.0	2.3	4.1	5.8	6.9	2.0	6.0	5.0	4.0	1.0	0.0
16	16.0	1.0	11.0	2.1	3.9	5.0	5.3	3.0	3.0	2.0	9.0	1.0	0.0
17	17.0	0.0	11.0	1.8	2.7	4.0	4.5	0.0	0.0	1.0	0.0	1.0	0.0
18	18.0	0.0	9.0	3.8	5.7	6.2	6.8	1.0	3.0	0.0	0.0	1.0	0.0
19	19.0	0.0	8.0	1.8	4.2	5.1	7.2	2.0	0.0	1.0	2.0	1.0	0.0
20	20.0	1.0	4.0	2.6	2.5	3.3	4.4	0.0	3.0	2.0	2.0	1.0	0.0
21	21.0	0.0	7.0	3.6	5.2	6.0	7.5	5.0	0.0	5.0	3.0	1.0	0.0

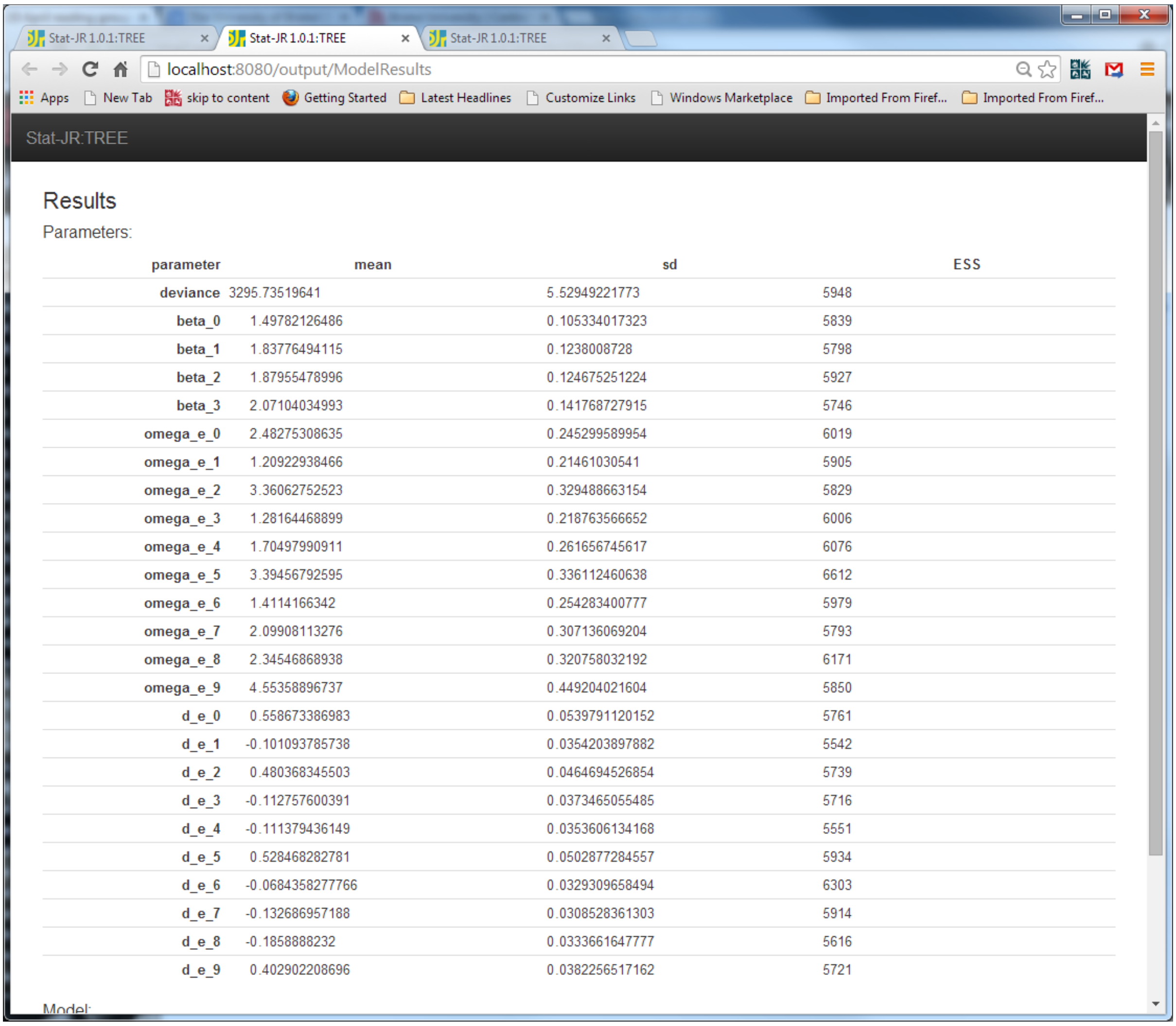
This dataset is identical to that used in Practical 1, apart from an additional column of 0s named *zero* which we will need later. It is in wide format and therefore there are four variables for the four years of antisocial behaviour measures. In the lecture we saw that antisocial behaviour trajectories are highly nonlinear and variable across children, so a linear or quadratic model is unlikely to fit well. We will therefore fit a multivariate model using another template named *1LevelMVNormal*. So, select *1LevelMVNormal* from the template list, and click **Use** and set-up inputs as follows:



Clicking on **Next** will give the LaTeX model description:



Note that the algebra system has some limited ability to fit multivariate models; when we answered *Yes* to "**Use MVNormal update for beta?**", all steps are in fact done by custom C code for this model. The equations show that all we are doing in this model is basically estimating the means and covariance matrix for the 4 variables. Clicking on **Run** gives the following results:



Results

Parameters:

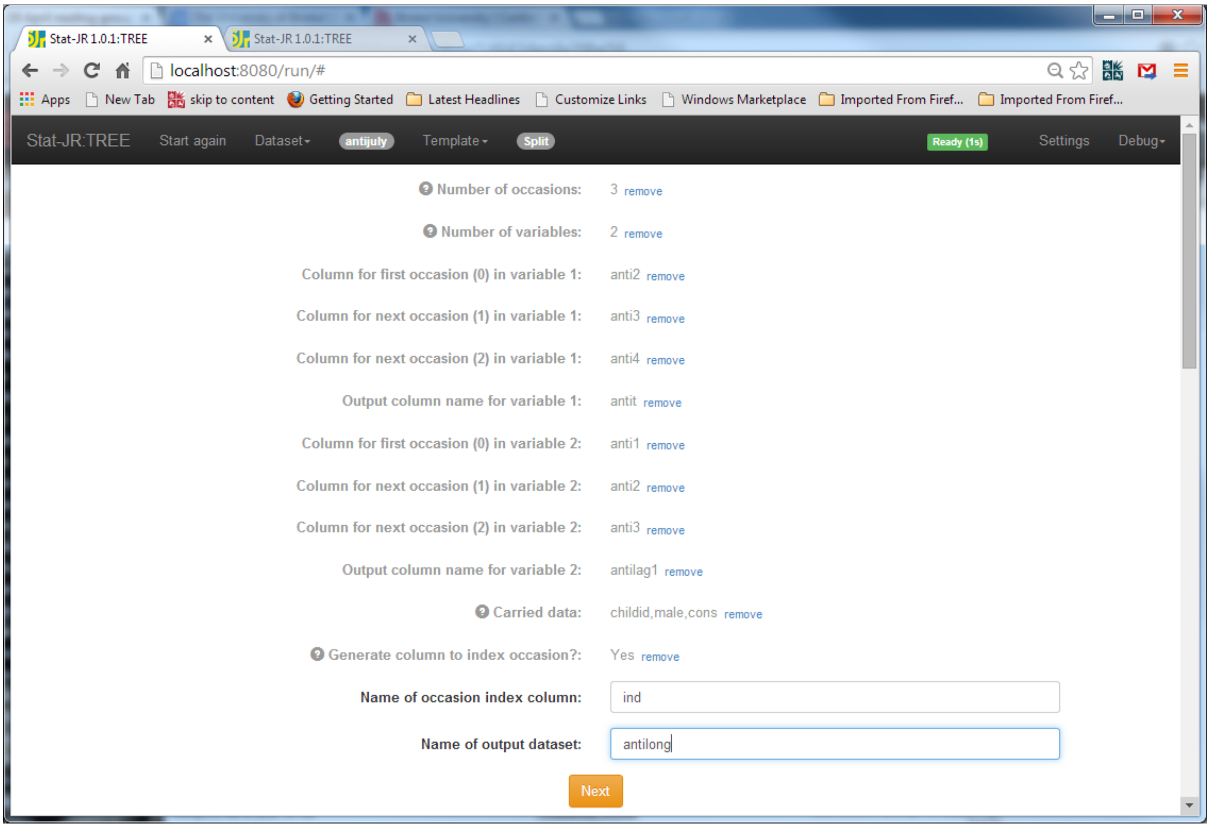
parameter	mean	sd	ESS
deviance	3295.73519641	5.52949221773	5948
beta_0	1.49782126486	0.105334017323	5839
beta_1	1.83776494115	0.1238008728	5798
beta_2	1.87955478996	0.124675251224	5927
beta_3	2.07104034993	0.141768727915	5746
omega_e_0	2.48275308635	0.245299589954	6019
omega_e_1	1.20922938466	0.21461030541	5905
omega_e_2	3.36062752523	0.329488663154	5829
omega_e_3	1.28164468899	0.218763566652	6006
omega_e_4	1.70497990911	0.261656745617	6076
omega_e_5	3.39456792595	0.336112460638	6612
omega_e_6	1.4114166342	0.254283400777	5979
omega_e_7	2.09908113276	0.307136069204	5793
omega_e_8	2.34546868938	0.320758032192	6171
omega_e_9	4.55358896737	0.449204021604	5850
d_e_0	0.558673386983	0.0539791120152	5761
d_e_1	-0.101093785738	0.0354203897882	5542
d_e_2	0.480368345503	0.0464694526854	5739
d_e_3	-0.112757600391	0.0373465055485	5716
d_e_4	-0.111379436149	0.0353606134168	5551
d_e_5	0.528468282781	0.0502877284557	5934
d_e_6	-0.0684358277766	0.0329309658494	6303
d_e_7	-0.132686957188	0.0308528361303	5914
d_e_8	-0.1858888232	0.0333661647777	5616
d_e_9	0.402902208696	0.0382256517162	5721

Model:

Here we see the four *beta* variables giving the mean levels of antisocial behaviour in the four time points, indicating an increasing trend of antisocial behaviour with time. We also see that all covariances are positive, meaning positive correlations for individual children's levels of antisocial behaviour over time. The variances (*omega_e* elements *0, 2, 5* and *9*) are also increasing over time, so not only is the level of antisocial behaviour rising but so is the variability across the group. This multivariate model does not include any child-level random effects. This is because we are estimating the full covariance matrix for the occasion-level residuals, which completely allows for correlation between a child's responses over time: there is no further correlation to explain by a child-level random effect.

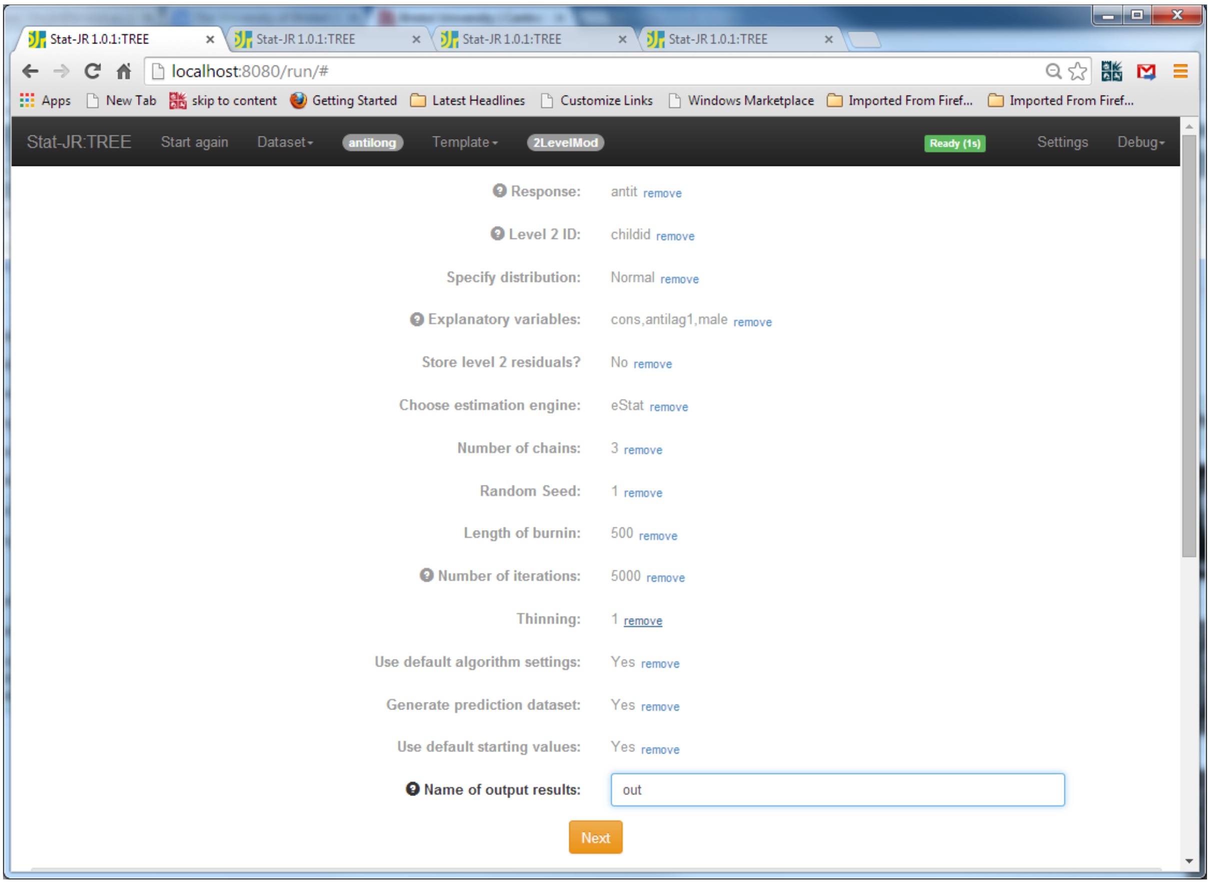
3.2 Dynamic AR(1) model for antisocial behaviour

We will next fit an AR(1) model for antisocial behaviour, ignoring initial conditions to begin with. To do this we need to construct a long version of the dataset that ignores the first time-point in the response variable but also includes a lagged response. We will do so by using the *Split* template that we used in Practical 1. Select *Split* from the template list and click on the **Use** button. Set up the inputs as follows:

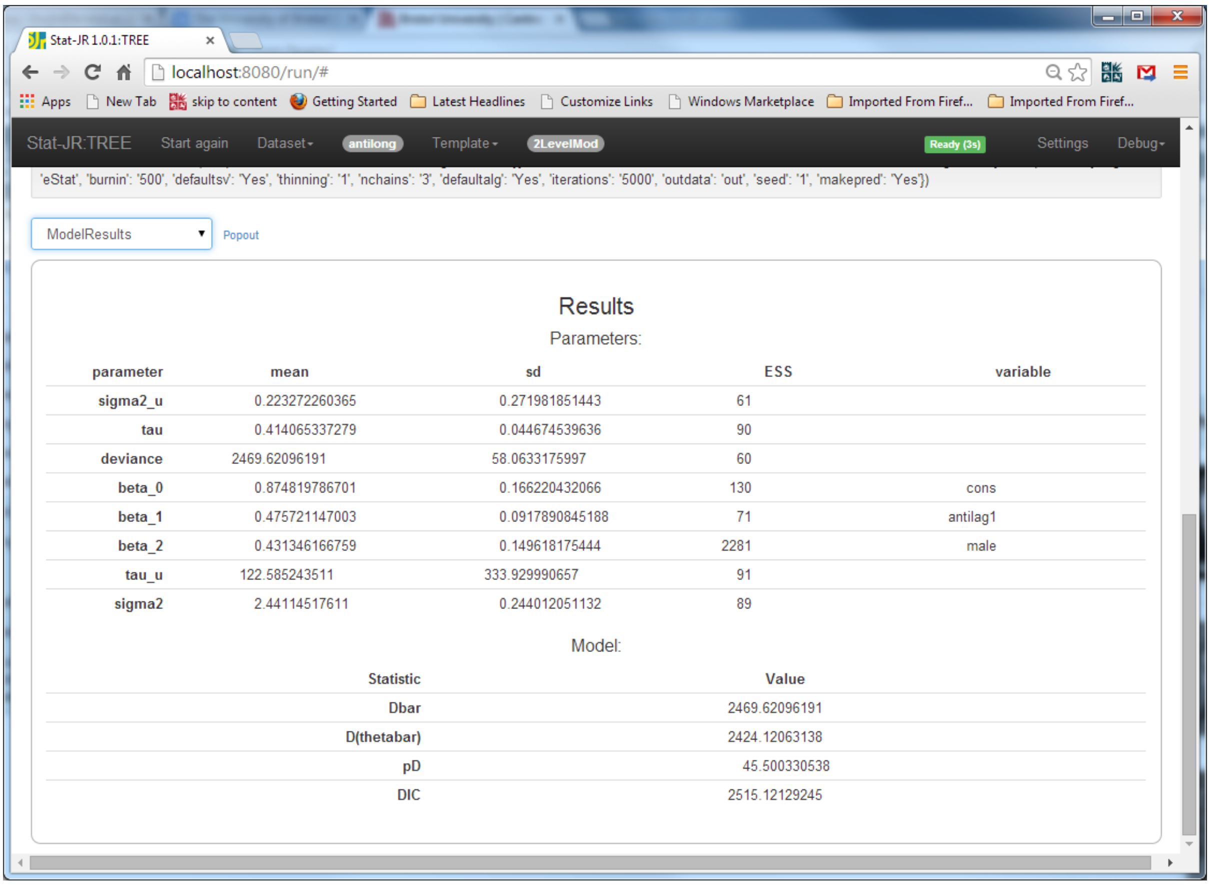


Clicking on **Next** and **Run** will now create the long version of the dataset.

We will now use *2LevelMod* to fit a model, so return to the main window and select *2LevelMod* from the template list and *antilong* from the dataset list, clicking on **Use** after each. Next we will set up the template inputs as follows:



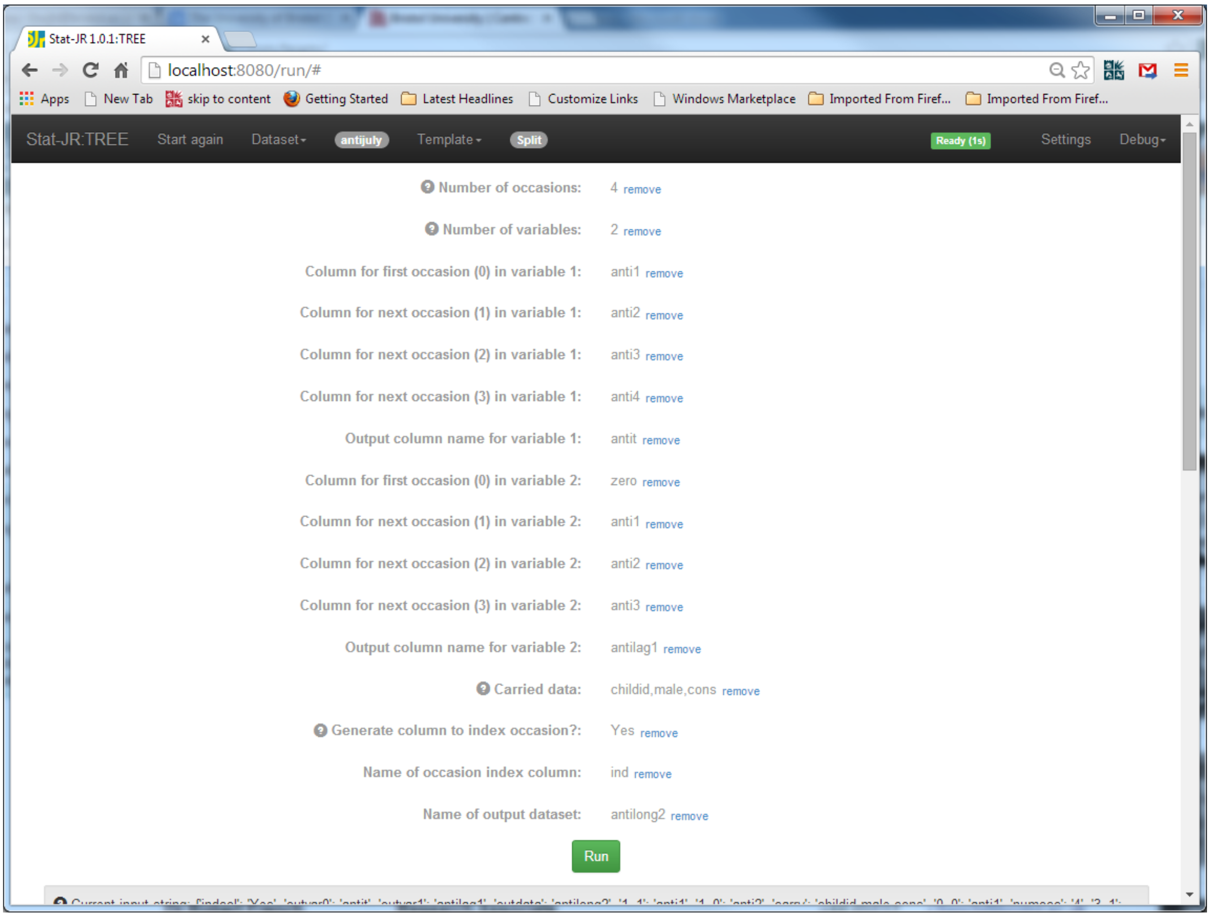
Clicking on **Next** and **Run** will fit the model and give the following results:



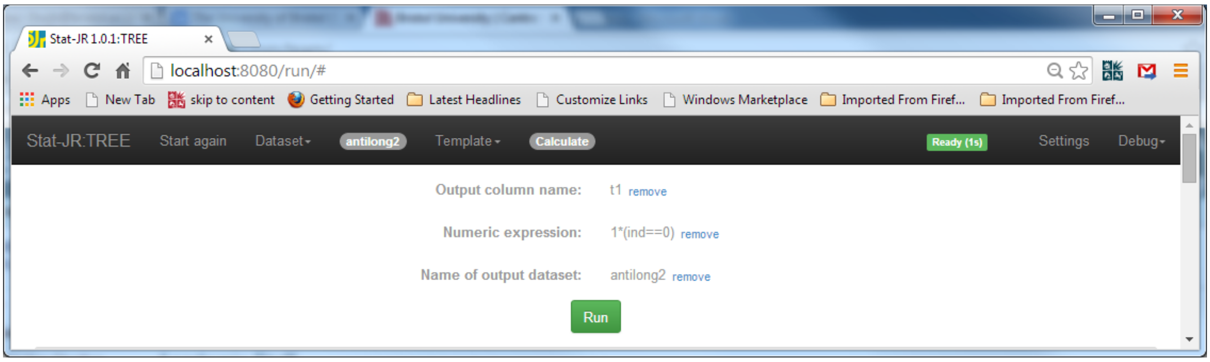
Here we get the same results as shown in the lecture slides (for the 'No IC' model) and we observe that the level 2 variance (*sigma2_u*) is small whilst there appears to be a significant lag effect (*beta_1*).

We next want to allow initial conditions by specifying an additional equation for the first time-point. To do this we need to return to the wide version of the data and construct a new long form that includes time point 1, and then we need to set-up various dummy variables and interactions to identify each line as belonging to either year 1 or a later year. In fact, most of the work in fitting this type of model is initial data manipulation to get the data in a form that can then be fitted using the standard template.

Firstly return to the main screen and set the template to *Split***,** and change the dataset to *antijuly* (i.e. wide form), clicking **Use** after each. Click on **Run** and set-up the template inputs as follows:



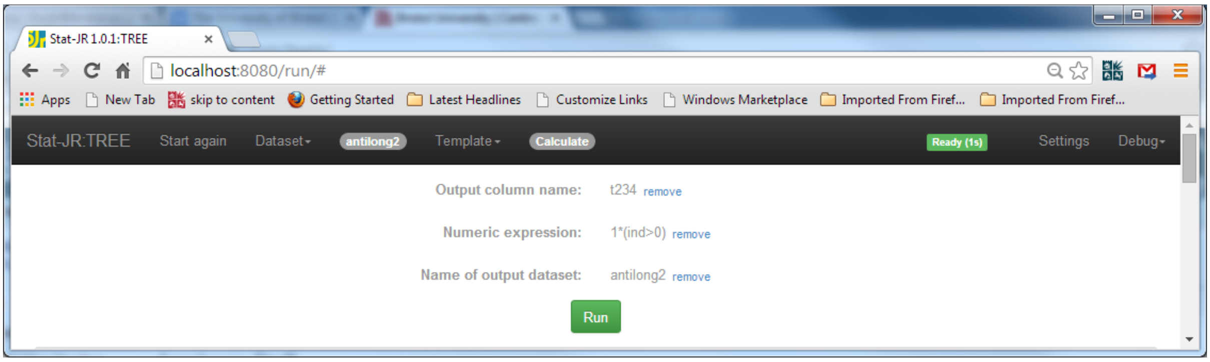
Here we are using the *zero* column as the lagged values for time point 1. Clicking on **Run** will now create the basic structure for our dataset. We next need to add some additional indicator variables, and so return to the top of the screen and change the dataset to *antilong2* (which we have just created) and the template to *Calculate,* clicking **Use**, as usual, between each choice. We will firstly generate a dummy variable to indicate when the data line is the first time point, as follows:



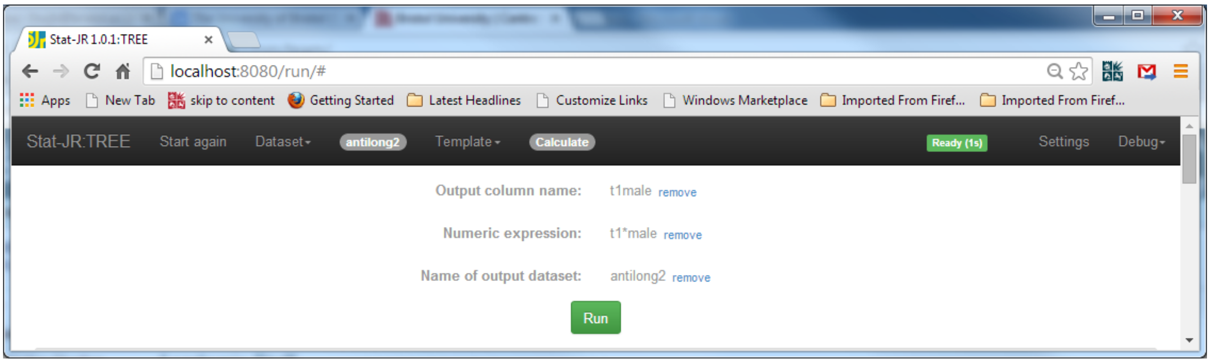
Clicking **Run** will add this variable to the dataset. We will now add three more variables to the dataset: *t234,* which takes a value of 0 when *ind* = 0, and a value of 1 when *ind* = 1, 2 or 3 (i.e. it indicates whether it is the first time point or not), and interactions for both these variables with the male variable (*t1male* and *t234male*).

To generate each of these, click on **Start Again** (in the black bar at the top) and fill in the inputs as shown below, clicking on **Next** and **Run**. Note *don't* do this too quickly: in particular make sure that the dataset appears in the right-hand pane before clicking **Start Again** or that variable will *not* be added to the dataset.

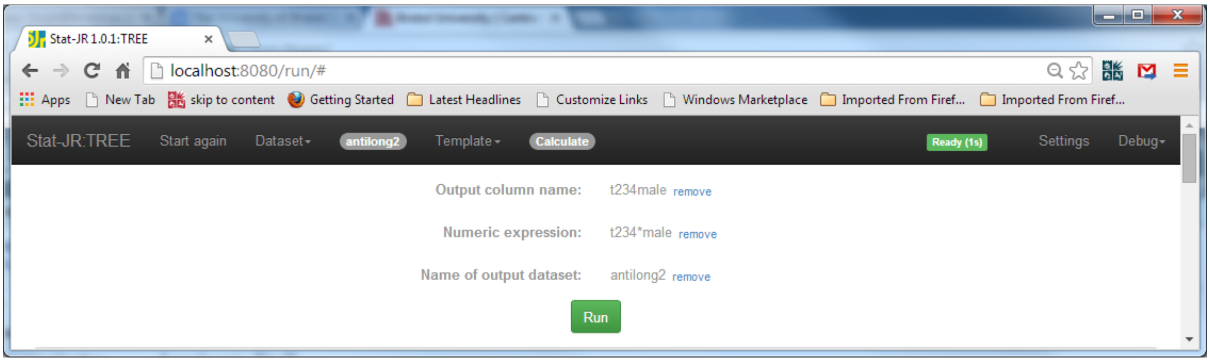
Firstly inputs for *t234*:



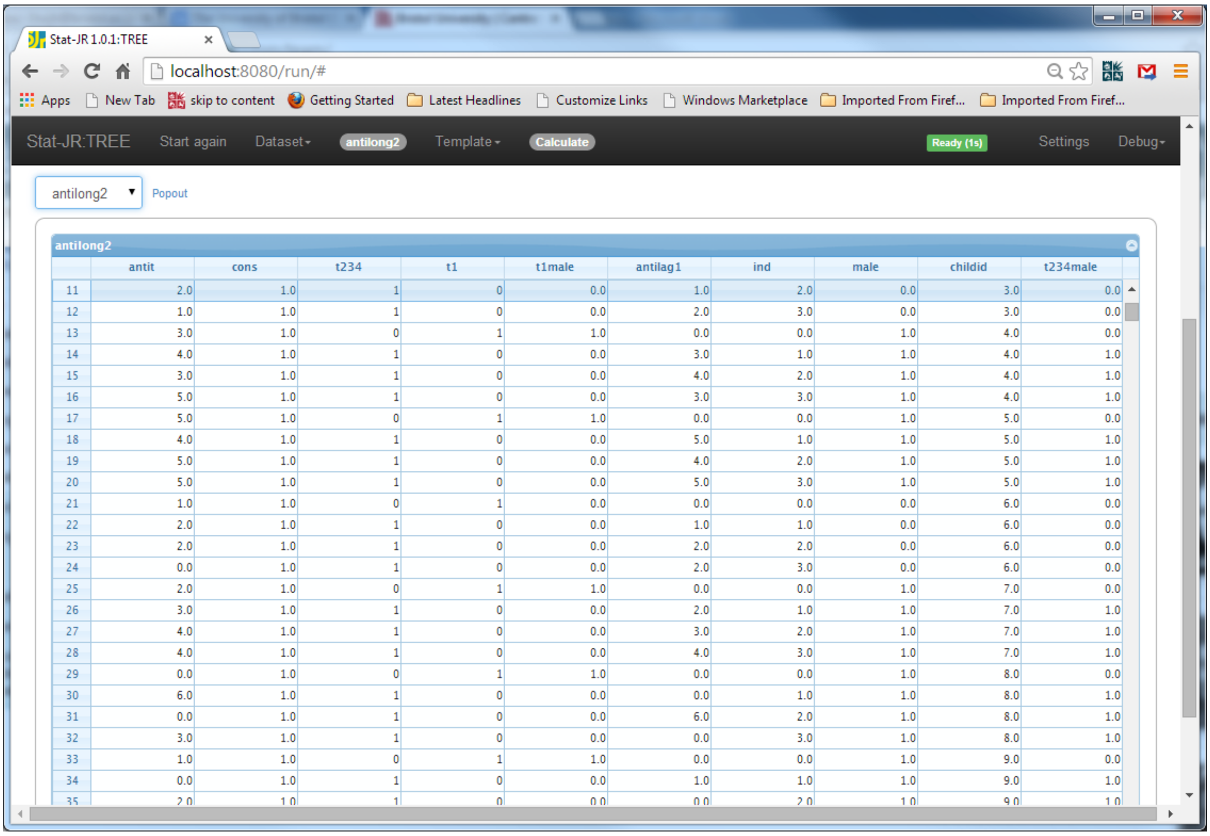
Next for *t1male*:



And finally for *t234male*:



After all the calculations, *antilong2* should look something like the window below (although note the column order may differ) :



antilong2

	antit	cons	t234	t1	t1male	antilag1	ind	male	childid	t234male
11	2.0	1.0	1	0	0.0	1.0	2.0	0.0	3.0	0.0
12	1.0	1.0	1	0	0.0	2.0	3.0	0.0	3.0	0.0
13	3.0	1.0	0	1	1.0	0.0	0.0	1.0	4.0	0.0
14	4.0	1.0	1	0	0.0	3.0	1.0	1.0	4.0	1.0
15	3.0	1.0	1	0	0.0	4.0	2.0	1.0	4.0	1.0
16	5.0	1.0	1	0	0.0	3.0	3.0	1.0	4.0	1.0
17	5.0	1.0	0	1	1.0	0.0	0.0	1.0	5.0	0.0
18	4.0	1.0	1	0	0.0	5.0	1.0	1.0	5.0	1.0
19	5.0	1.0	1	0	0.0	4.0	2.0	1.0	5.0	1.0
20	5.0	1.0	1	0	0.0	5.0	3.0	1.0	5.0	1.0
21	1.0	1.0	0	1	0.0	0.0	0.0	0.0	6.0	0.0
22	2.0	1.0	1	0	0.0	1.0	1.0	0.0	6.0	0.0
23	2.0	1.0	1	0	0.0	2.0	2.0	0.0	6.0	0.0
24	0.0	1.0	1	0	0.0	2.0	3.0	0.0	6.0	0.0
25	2.0	1.0	0	1	1.0	0.0	0.0	1.0	7.0	0.0
26	3.0	1.0	1	0	0.0	2.0	1.0	1.0	7.0	1.0
27	4.0	1.0	1	0	0.0	3.0	2.0	1.0	7.0	1.0
28	4.0	1.0	1	0	0.0	4.0	3.0	1.0	7.0	1.0
29	0.0	1.0	0	1	1.0	0.0	0.0	1.0	8.0	0.0
30	6.0	1.0	1	0	0.0	0.0	1.0	1.0	8.0	1.0
31	0.0	1.0	1	0	0.0	6.0	2.0	1.0	8.0	1.0
32	3.0	1.0	1	0	0.0	0.0	3.0	1.0	8.0	1.0
33	1.0	1.0	0	1	1.0	0.0	0.0	1.0	9.0	0.0
34	0.0	1.0	1	0	0.0	1.0	1.0	1.0	9.0	1.0
35	2.0	1.0	1	0	0.0	0.0	2.0	1.0	9.0	1.0

We will fit a model where the fixed part is as follows:

$$\beta_0 t1 + \beta_1 t1male + \beta_2 t234 + \beta_3 t234antil1 + \beta_4 t234male$$

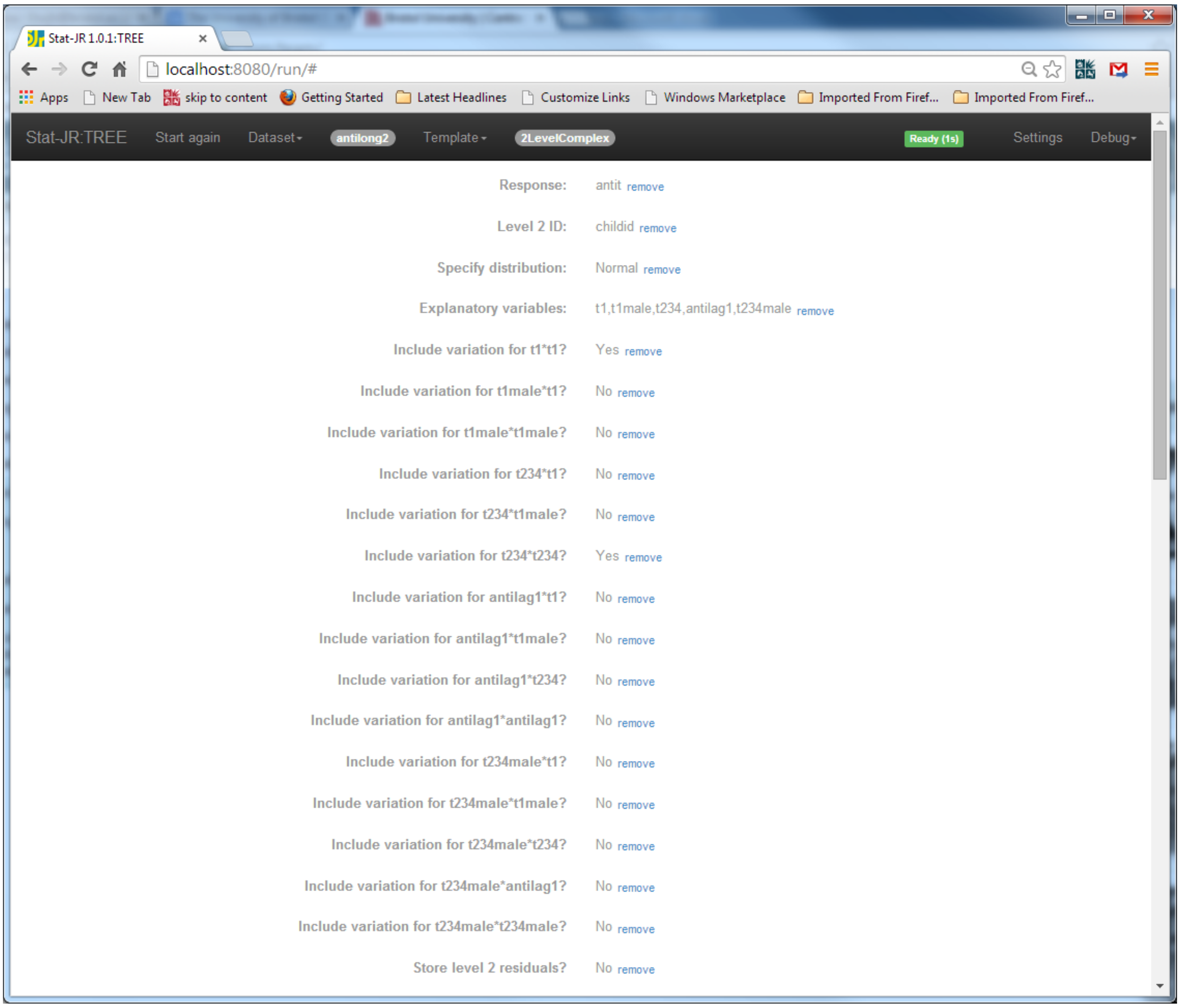
This breaks down into two equations for t=1 and t>1:

For t = 1 *t1* = 1 and *t234* = 0 thus we have $\beta_0 + \beta_1 male$

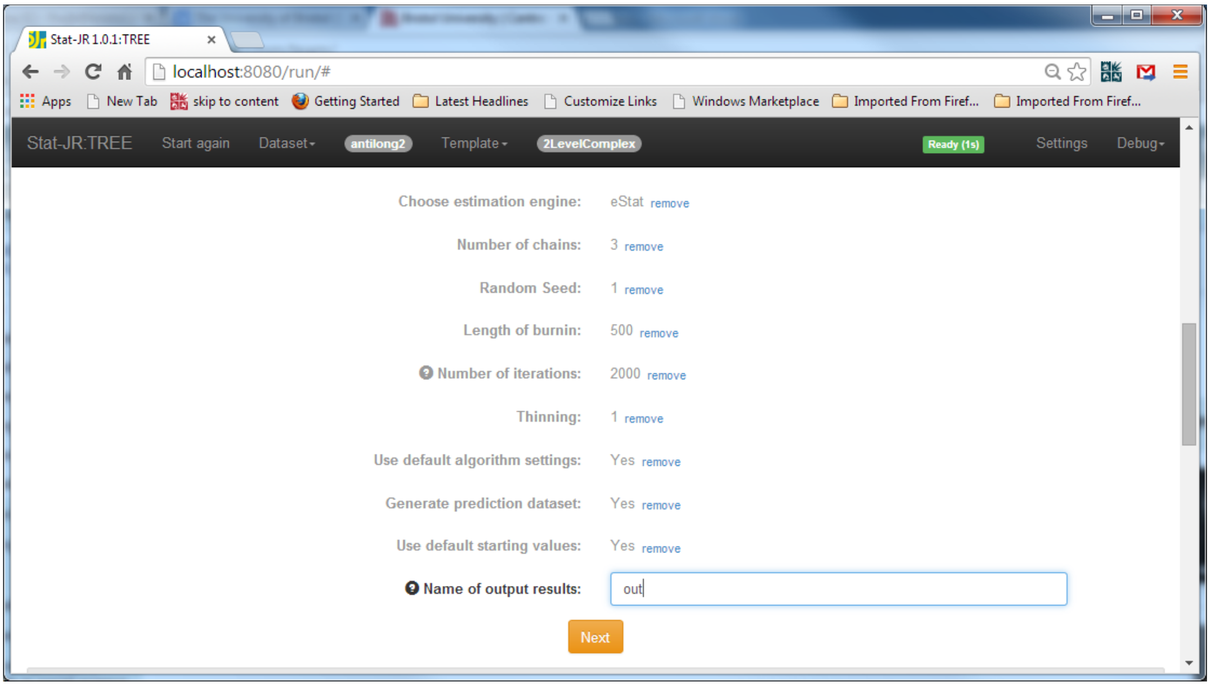
For t > 1 *t1* = 0 and *t234* = 1 thus we have $\beta_2 + \beta_3 antil1 + \beta_4 male$

Turning now to the random part of the model, we will allow a single child-level random effect for each child but we will allow the variability of the residuals at level 1 to be different for the first, and later, time-points. In order to fit different variances in Stat-JR we need to use the template *2LevelComplex* which allows complex variability at level 1.

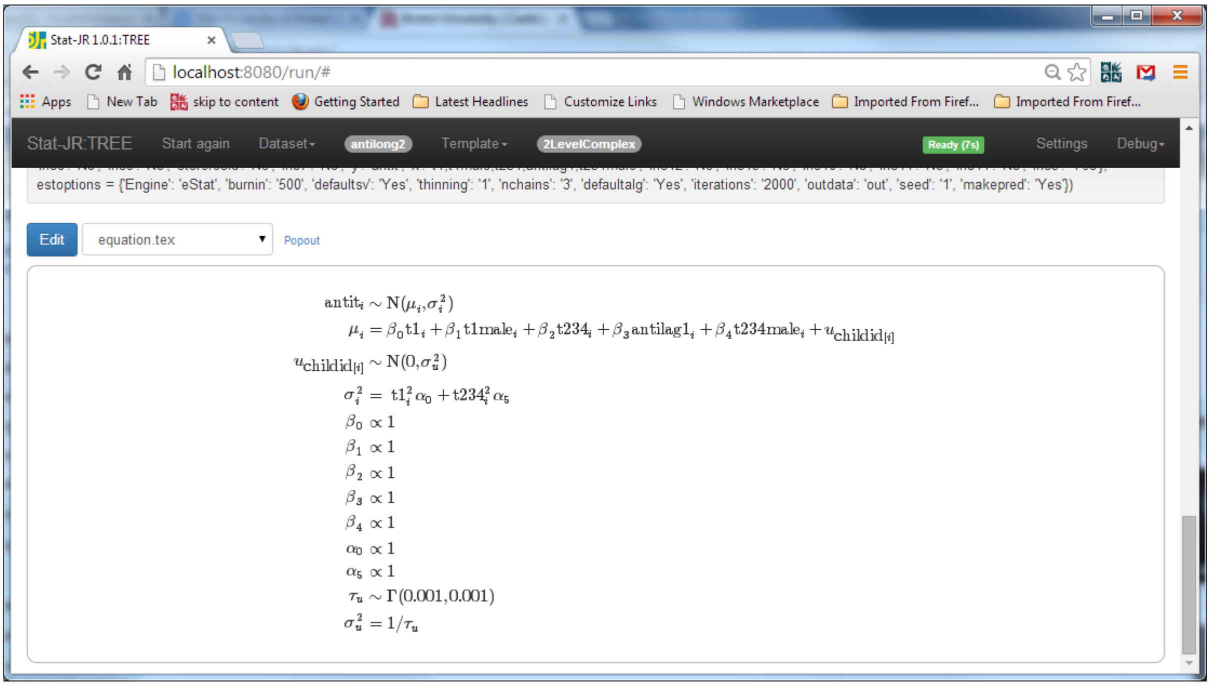
So, select *2LevelComplex* from the template list, and click on **Use** and set up the inputs as follows (note that the only two terms (of the large number) that we want variation at level 1 for are *t1*t1* and *t234*t234*):



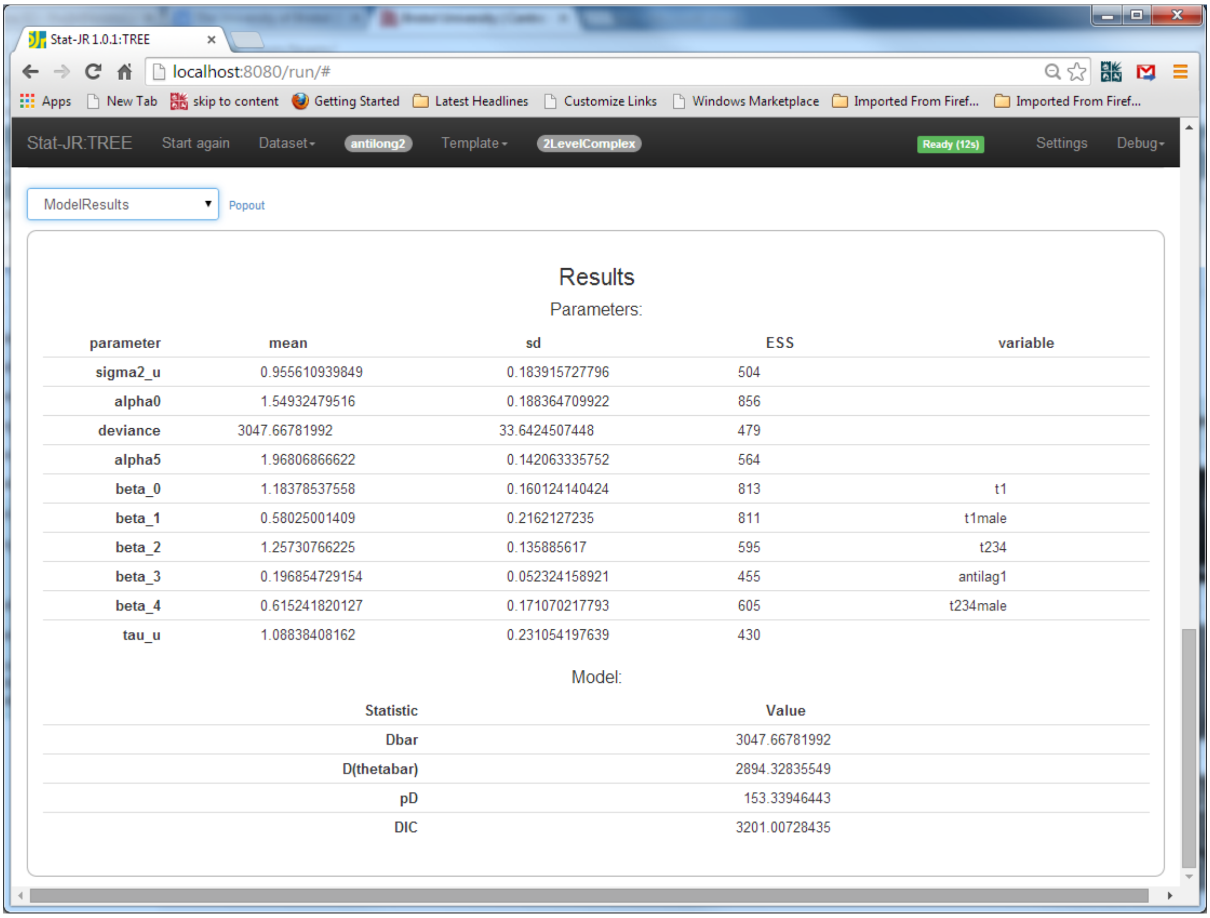
and



Clicking **Next** brings up the LateX for the model, as follows:



Here you can see that *alpha0* and *alpha5* are the variances for the first and subsequent time points, respectively. Clicking on **Run** will fit the model, after which the *ModelResults* appear as follows:



Results

Parameters:

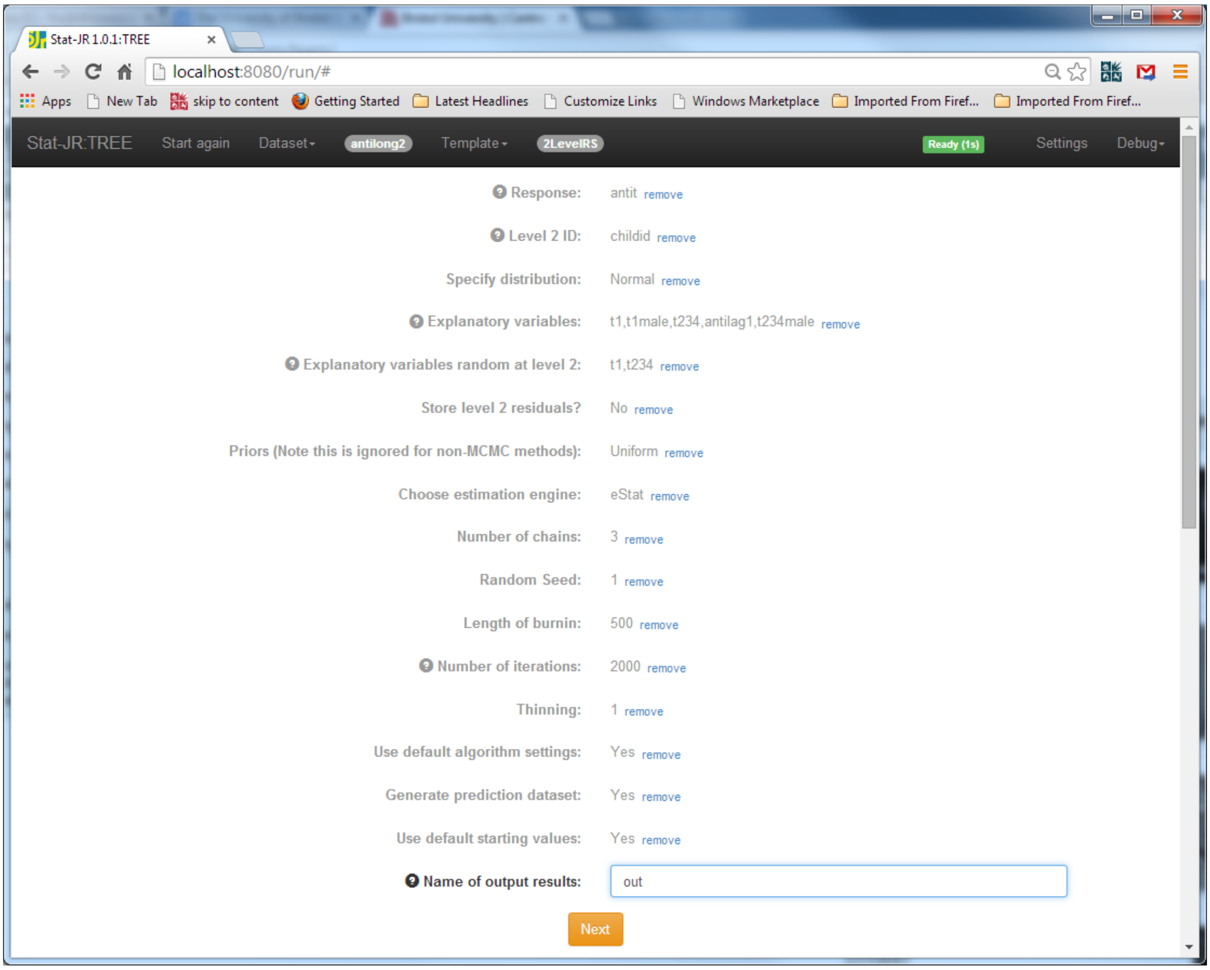
parameter	mean	sd	ESS	variable
sigma2_u	0.955610939849	0.183915727796	504	
alpha0	1.54932479516	0.188364709922	856	
deviance	3047.66781992	33.6424507448	479	
alpha5	1.96806866622	0.142063335752	564	
beta_0	1.18378537558	0.160124140424	813	t1
beta_1	0.58025001409	0.2162127235	811	t1male
beta_2	1.25730766225	0.135885617	595	t234
beta_3	0.196854729154	0.052324158921	455	antilag1
beta_4	0.615241820127	0.171070217793	605	t234male
tau_u	1.08838408162	0.231054197639	430	

Model:

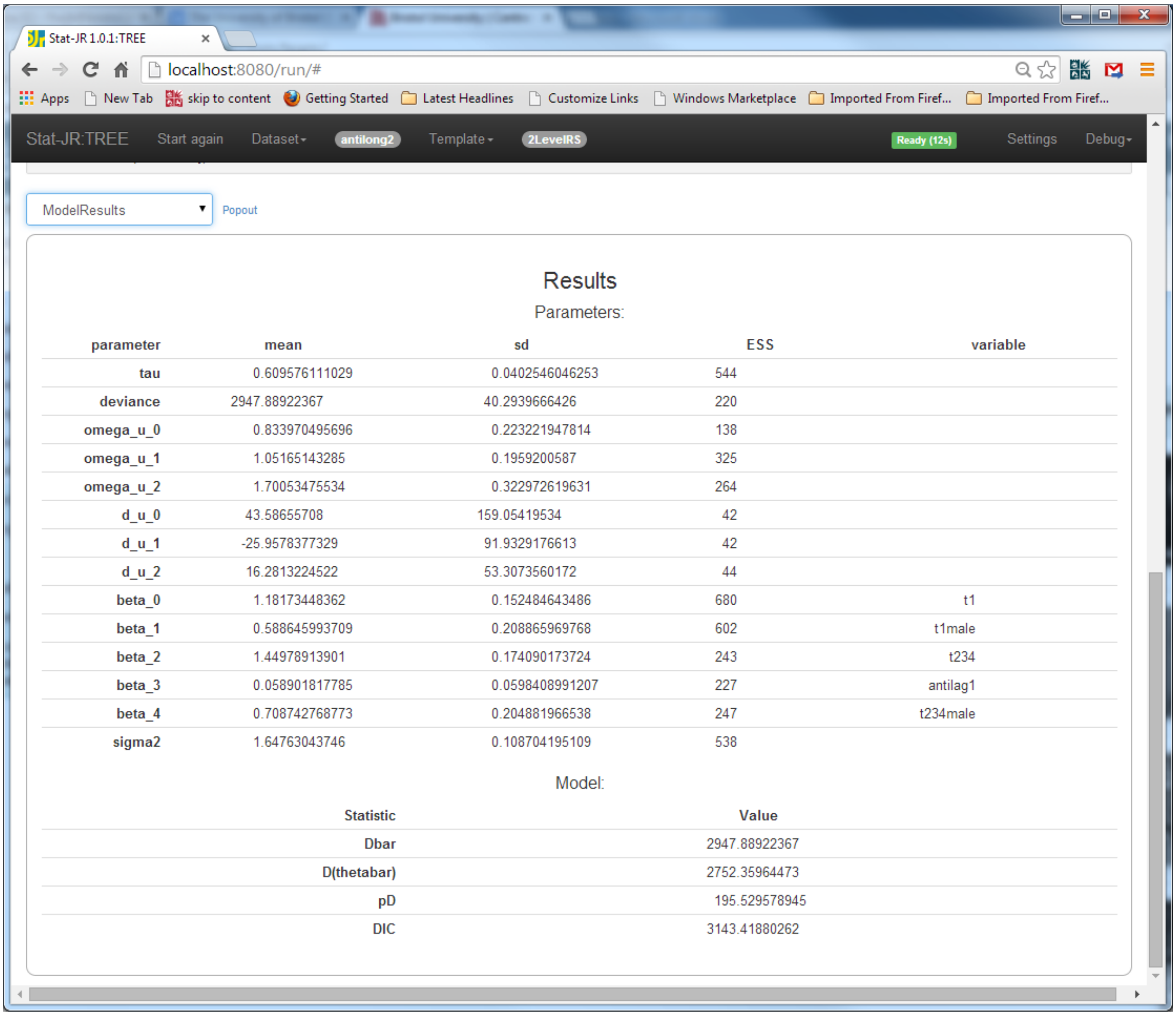
Statistic	Value
Dbar	3047.66781992
D(thetabar)	2894.32835549
pD	153.33946443
DIC	3201.00728435

These results should correspond to initial conditions 1 (IC1) in the lecture notes, and we can now see a significantly larger level 2 variance, whilst the effect of the lagged variable has reduced. The alternative approach is to fit an extended model with separate child-level random effects for $t = 1$ and $t > 1$ whilst maintaining the same fixed effects in the model. We do this by using the template *2LevelRS* which assumes the same variance at level 1 for each time point whilst allowing a different variance at level 2 for *t1* and *t234* and a covariance between them.

Return to the main window and choose *2LevelRS* as the template and click **Use** and fill in the inputs as shown below:



Clicking **Next** and **Run** will fit the model and the *ModelResults* can be found from the objects list:



Results

Parameters:

parameter	mean	sd	ESS	variable
tau	0.609576111029	0.0402546046253	544	
deviance	2947.88922367	40.2939666426	220	
omega_u_0	0.833970495696	0.223221947814	138	
omega_u_1	1.05165143285	0.1959200587	325	
omega_u_2	1.70053475534	0.322972619631	264	
d_u_0	43.58655708	159.05419534	42	
d_u_1	-25.9578377329	91.9329176613	42	
d_u_2	16.2813224522	53.3073560172	44	
beta_0	1.18173448362	0.152484643486	680	t1
beta_1	0.588645993709	0.208865969768	602	t1male
beta_2	1.44978913901	0.174090173724	243	t234
beta_3	0.058901817785	0.0598408991207	227	antilag1
beta_4	0.708742768773	0.204881966538	247	t234male
sigma2	1.64763043746	0.108704195109	538	

Model:

Statistic	Value
Dbar	2947.88922367
D(thetabar)	2752.35964473
pD	195.529578945
DIC	3143.41880262

These results are for the model IC(2) in the slides.