

eBook Writing Workshop – Practical 3: Using Supertemplates and incorporating html outputs into eBooks.

Introduction

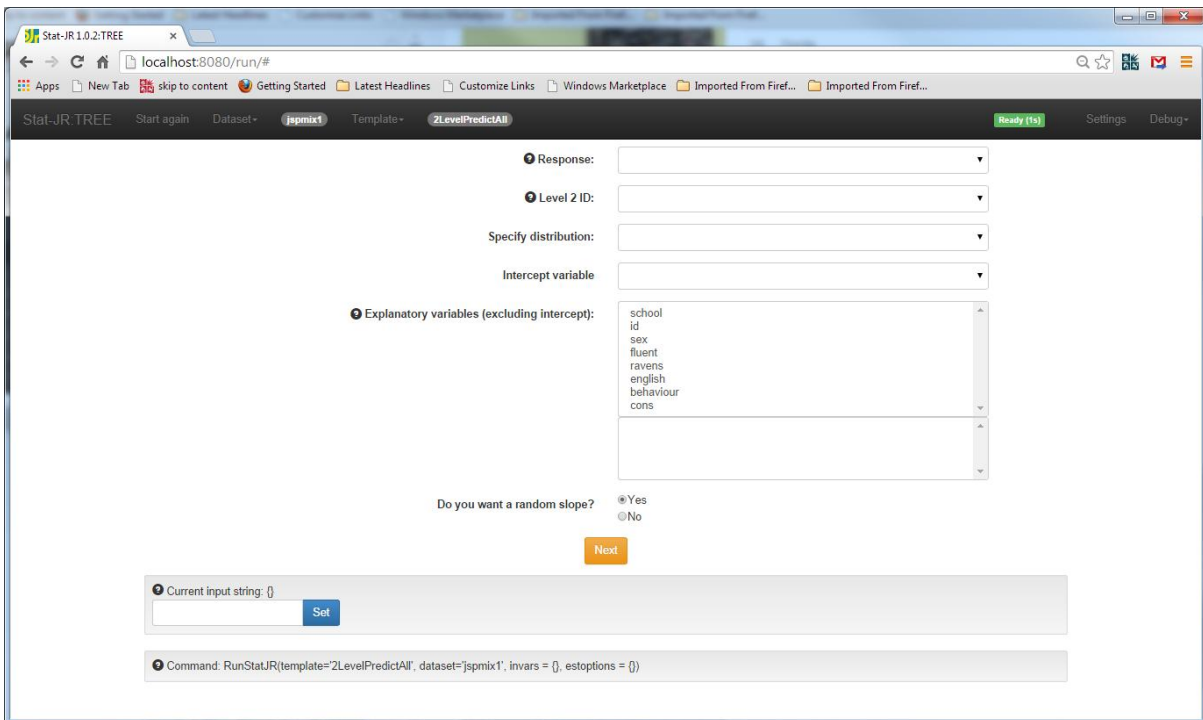
In this third practical we firstly are going to look at how we might construct an eBook that allows the reader to fit multilevel models to their dataset. We could perform this using the skills we have already learnt in practicals 1 and 2 by simply choosing a 2 level modelling template e.g. **2LevelMod** in Stat-JR TREE and using the eBook-writer. There are, however, several post-estimation plots associated with fitting multilevel models it would also be useful to include in an eBook. These are available in other Stat-JR templates, which due to current limitations discussed in practical 2 cannot be linked directly within the DEEP system. To use such plots in an eBook we require a super-template that calls both the model fitting and plotting templates, passing the output from one as input into the other. We will start by using the eBook-writer but will then look in a little detail at how a super-template works and how we might modify it to include textual output.

Creating a Multilevel Modelling eBook

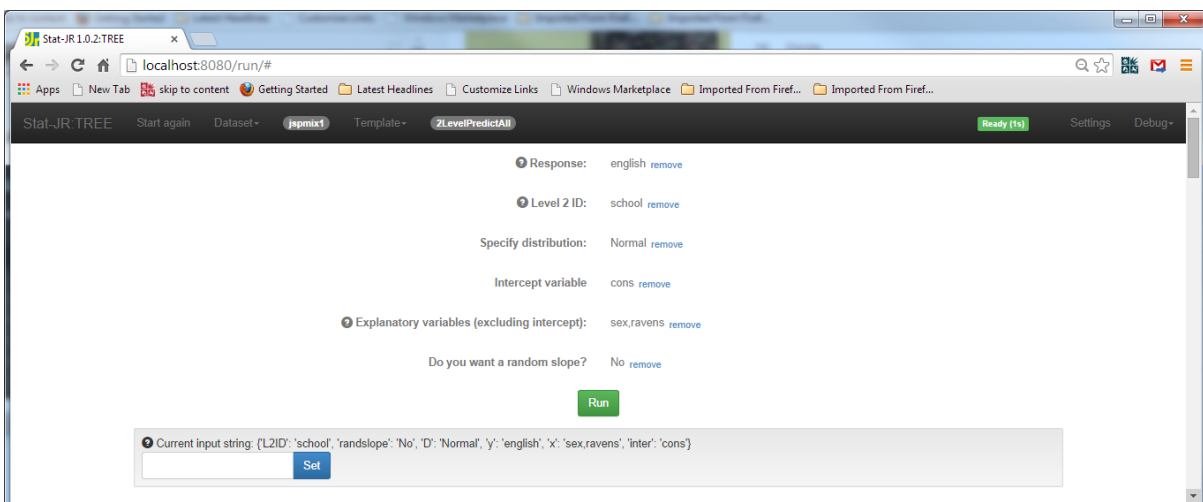
To fit multilevel models in Stat-JR's TREE interface there are several templates including **2LevelMod**, which fits 2 level random intercepts models, and **2LevelRS**, which fits 2 Level random slopes models (with random intercepts as a special case). These templates simply fit the model and return estimates along with residuals and predictions in the form of data files. We have therefore constructed a super template called **2LevelPredictAll** which combines the model-fitting with graphical outputs by chaining together templates. We will begin by creating an eBook using only this super template. As with the other practicals, we begin by starting up Stat-JR TREE afresh which produces the following:



We click **Begin** and select the dataset **jspmix1** (here you will choose your own dataset) from the **Dataset** pull-down list and select the template **2LevelPredictAll** from the **Template** pull down list. Having done this the screen will look as follows:



Here we require the response and predictor (explanatory) variables as we have seen for regression models previously, although in this case the intercept column is specified separately. We then require the column containing the level 2 identifiers and the distribution type (which we will here choose as **Normal**). We are also asked about random slopes as the super-template has different behaviour based on this. We will fill in the inputs as follows:



This super-template is currently intended for demonstration purposes only so, for example, it doesn't ask for the (MCMC) estimation engine settings but uses some defaults. Here I am trying one binary predictor (**sex**) and one continuous predictor (**ravens**) and, for now, running a simple random intercepts model. We suggest you choose two predictors in your own dataset. Clicking on **Run** will run the super-template and eventually the timer will return to the green "Ready" state and the object list will be filled. The Python script that makes up much of the code for the template is what you will initially see in the object browser. If we pop it out then we can see the code (here scrolled down from the top by a few lines):

```

modelInputs['priors'] = 'Uniform'
modelInputs['storeressid'] = 'No'

estinputs = {}
estinputs['engine'] = 'eStat'
estinputs['defaults'] = 'Yes'
estinputs['defaultalg'] = 'Yes'
estinputs['burnin'] = '500'
estinputs['iterations'] = '2000'
estinputs['seed'] = '1'
estinputs['thinning'] = '1'
estinputs['nchains'] = '3'
estinputs['makepred'] = 'Yes'
estinputs['outdata'] = 'chains'

m = RunStatJR(template='2LevelRS', dataset='workdata', invars = modelInputs, estoptions = estinputs)
for k in m.eng.inputs.keys():
    outputs['Model_' + k] = m.eng.inputs[k]

for k in m.eng.outputs.keys():
    outputs['Model_' + k] = m.eng.outputs[k]

for i in range(0, len(x)):
    calcinputs = {}
    calcinputs['outcol'] = 'prediction_'+str(x[i])
    calcinputs['expr'] = 'pred_beta_0_'+str(inter)+'pred_beta_'+str(i+1)+'_'+str(x[i])+'+pred_u0'
    calcinputs['outdata'] = 'prediction_datafile'
    m = RunStatJR(template='Calculate', dataset='prediction_datafile', invars = calcinputs, estoptions = {})

    if randslope:
        if str(x[i]) in list(x2):
            calcinputs['expr'] = 'prediction_'+str(x[i])+'+prediction_'+str(x[i])+ 'pred_u' + str(i+list(x2).index(str(x[i])))
            m = RunStatJR(template='Calculate', dataset='prediction_datafile', invars = calcinputs, estoptions = {})

    if D == 'Binomial':
        calcinputs['expr'] = 'exp(prediction_'+str(x[i])+')/(1+exp(prediction_'+str(x[i])+'))'
        m = RunStatJR(template='Calculate', dataset='prediction_datafile', invars = calcinputs, estoptions = {})
    if D == 'Poisson':
        calcinputs['expr'] = 'exp(prediction_'+str(x[i])+')'
        m = RunStatJR(template='Calculate', dataset='prediction_datafile', invars = calcinputs, estoptions = {})

for i in range(0, len(x)):
    graphinputs = {}
    graphinputs['xaxis'] = str(x[i])
    graphinputs['group'] = str(L2ID)
    graphinputs['filter'] = str(L2ID) + '=' + str(L2ID)

```

You may notice towards the top of the screen that the estimation inputs that have been hard-wired, so for example this template always runs 3 chains (**nchains**) for 2000 iterations. The individual templates that make up the super-template are called via the **RunStatJR** command within the Python code, so here we can see both the **2LevelRS** template and the **Calculate** template called in the code. We use the identifier **m** to store the list of objects that are output from each template execution and these are then uniquely named and copied into the global super-template objects list via the **outputs** object.

We will now look at some of the objects that are returned by the super-template. If we want to look at the model we have fitted then we choose the object **Model_equation.tex** and see:

```

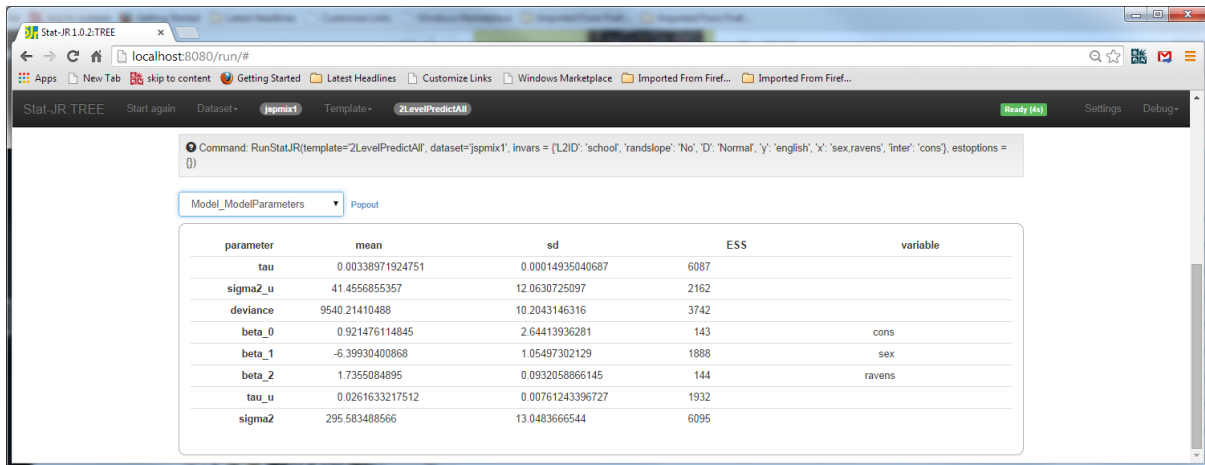
Command: RunStatJR(template='2LevelPredictAll', dataset='jspmix1', invars = (L2ID:'school', 'randslope': 'No', 'D': 'Normal', 'y': 'english', 'x': 'sex,ravens', 'inter': 'cons'), estoptions =
{}

Model_equation.tex
Popout

englishi ~ N(μi, σ2)
μi = β0consi + β1sexi + β2ravensi + u0,school[i]}(2)consi
u0,school[i]}(2) ~ N(0, σ02)
τ02 ~ Γ(0.001, 0.001)
σ02 = 1/τ0
β0 ∝ 1
β1 ∝ 1
β2 ∝ 1
τ ~ Γ(0.001, 0.001)
σ2 = 1/τ

```

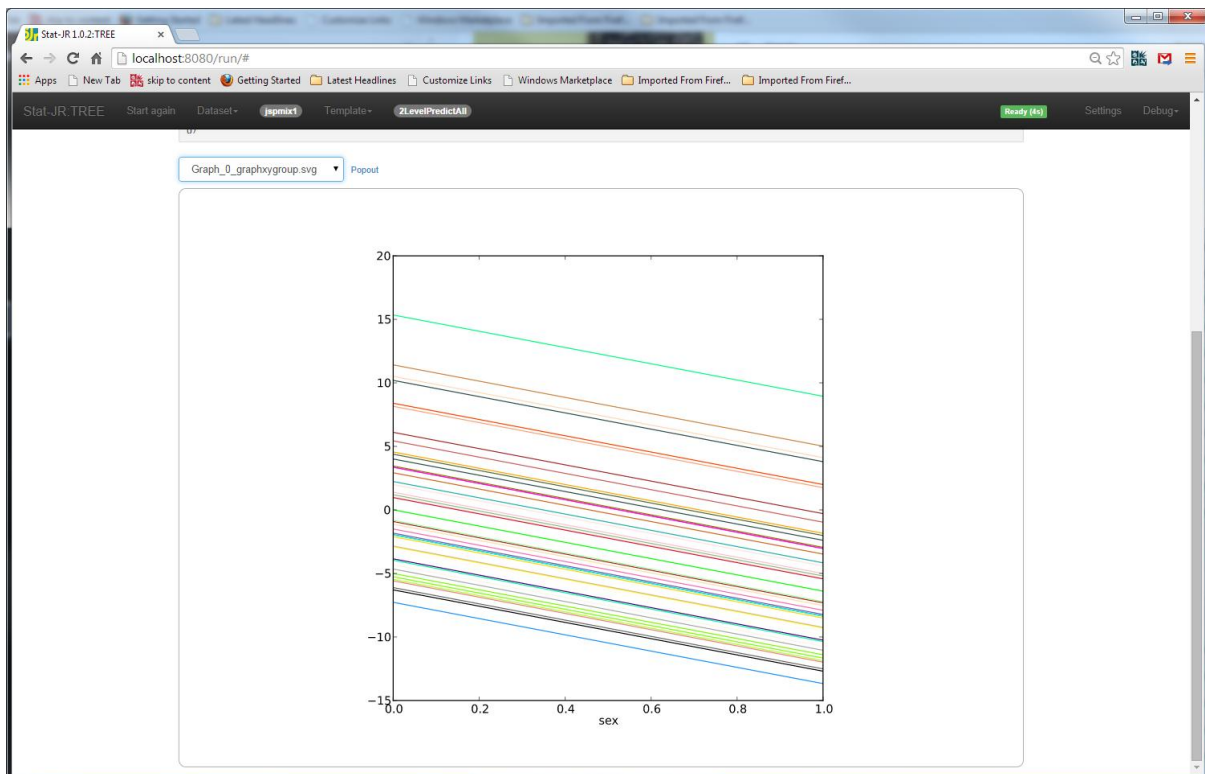
In fact all objects generated by the template **2LevelIRS** will begin with the string **Model_**, for example the parameter estimates are **Model_ModelParameters** as shown below:



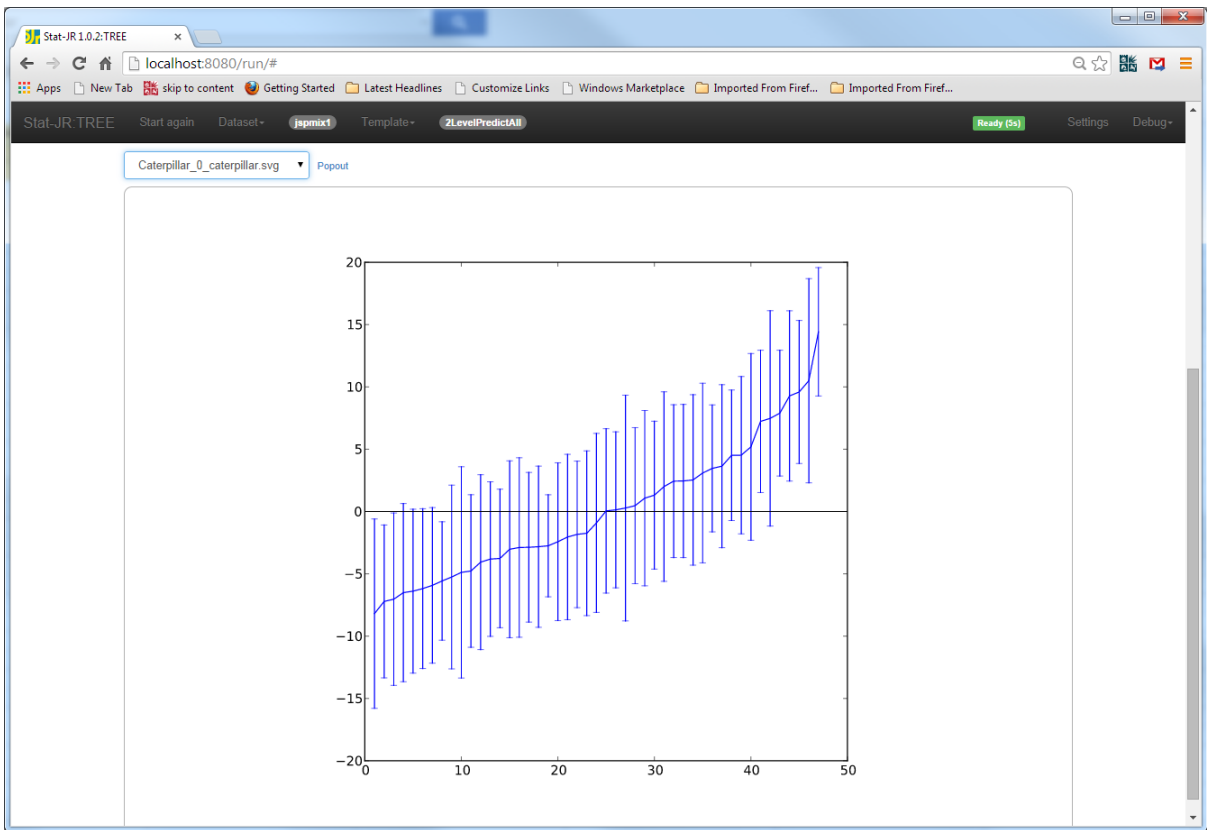
Command: RunStatJR(template="2LevelPredictAll", dataset="jspmix1", invars = (L2ID: 'school', 'randslope', 'No', 'D': 'Normal', 'Y': 'english', 'X': 'sex,ravens', 'inter', 'cons'), estoptions = {})

parameter	mean	sd	ESS	variable
tau	0.00338971924751	0.00014935040687	6087	
sigma2_u	41.4556855357	12.0630725097	2162	
deviance	9540.21410488	10.2043146316	3742	
beta_0	0.921476114845	2.64413936281	143	cons
beta_1	-6.39930400868	1.05497302129	1888	sex
beta_2	1.7355084895	0.0932058866145	144	ravens
tau_u	0.0261633217512	0.00761243396727	1932	
sigma2	295.583488566	13.0483666544	6095	

Here we see (by looking at the **mean** and **sd** columns and comparing the values) that the predictors **sex** and **ravens** are both significant predictors of **english**. We can also look at prediction plots, so for example the object **Graph_0_graphxygroup.svg** is the plot of the individual cluster (school in this case) lines against gender for the predictor **sex** as shown below:



Here the lines are all parallel as we are fitting a random intercept model, and they have a negative slope due to the negative **sex** effect. We can also look at the school-level residuals in a caterpillar plot by selecting **Caterpillar_0_caterpillar.svg**

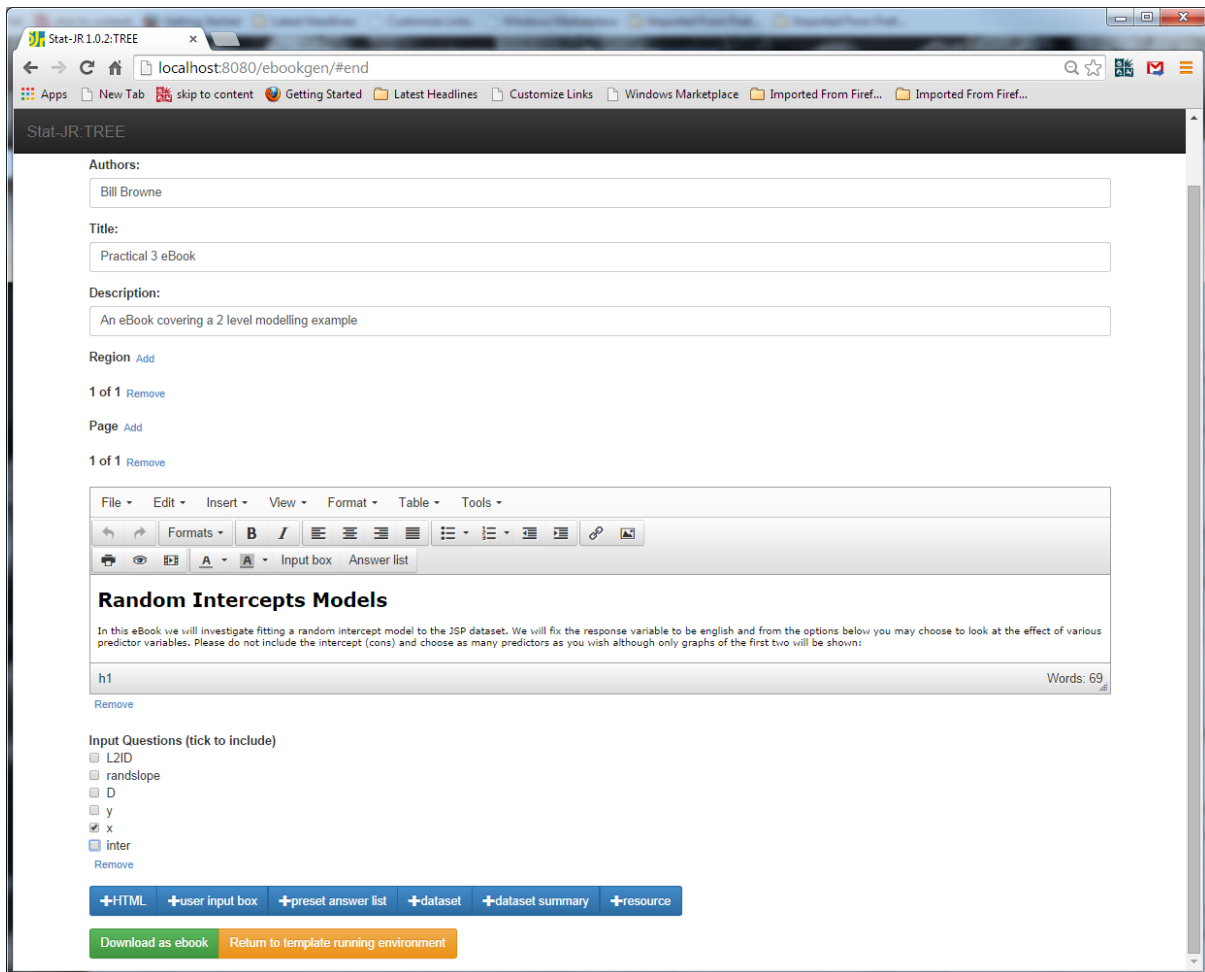


This shows the variability in the schools and you will note that some of the confidence intervals do not overlap with 0 which will explain why it is important to fit the school effects in the model. Let's now try and put some of these objects into an eBook. Scrolling up a bit we can find the **Add to eBook** button and click on it to start writing our eBook. We firstly fill in the top level information:

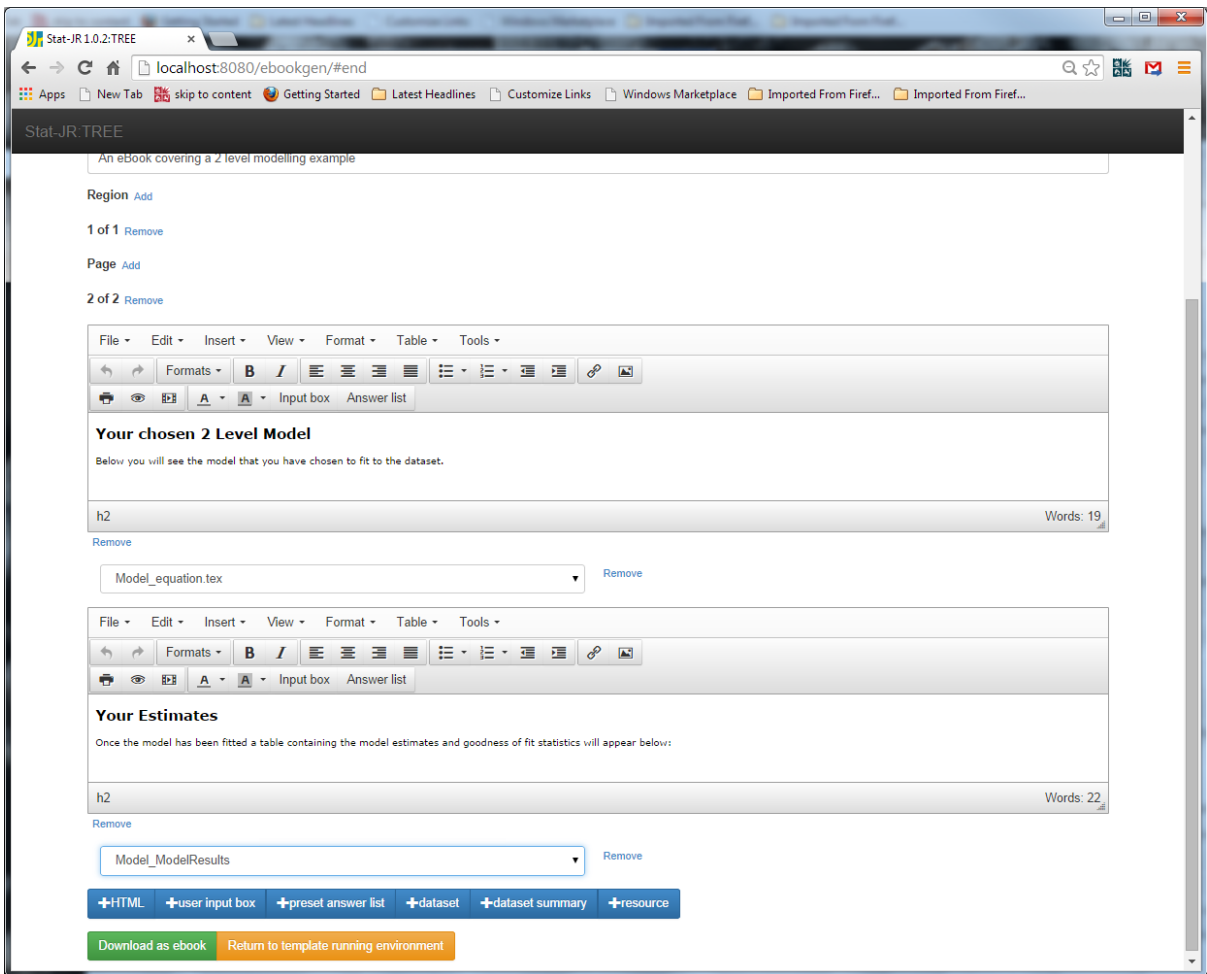
The screenshot shows a web browser window with the URL 'localhost:8080/ebookgen/#end'. The page title is 'Stat-JR TREE'. The form contains the following fields and buttons:

- Authors:** A text input field containing 'Bill Browne'.
- Title:** A text input field containing 'Practical 3 eBook'.
- Description:** A text input field containing 'An eBook covering a 2 level modelling example'.
- Region:** A label with a small 'Add' link next to it.
- Buttons:** A green button labeled 'Download as eBook' and an orange button labeled 'Return to template running environment'.

We will then click on **Add Region** and **Add Page** and start the eBook off with a HTML box and the opportunity for the user to input their settings as shown below:



In this case we are allowing the user to only choose a random intercept model (by not including the **randslope** input which will then be fixed at **No** as we specified in TREE). We also have decided to only allow the reader to input which predictors to test (**x**) as this is the only ticked input. We will next write a second page with the equations and the results (both estimates and fit statistics) by clicking **Add** next to **Page** and filling in the details as shown overleaf, where we add 2 HTML boxes with a resource after each of them:



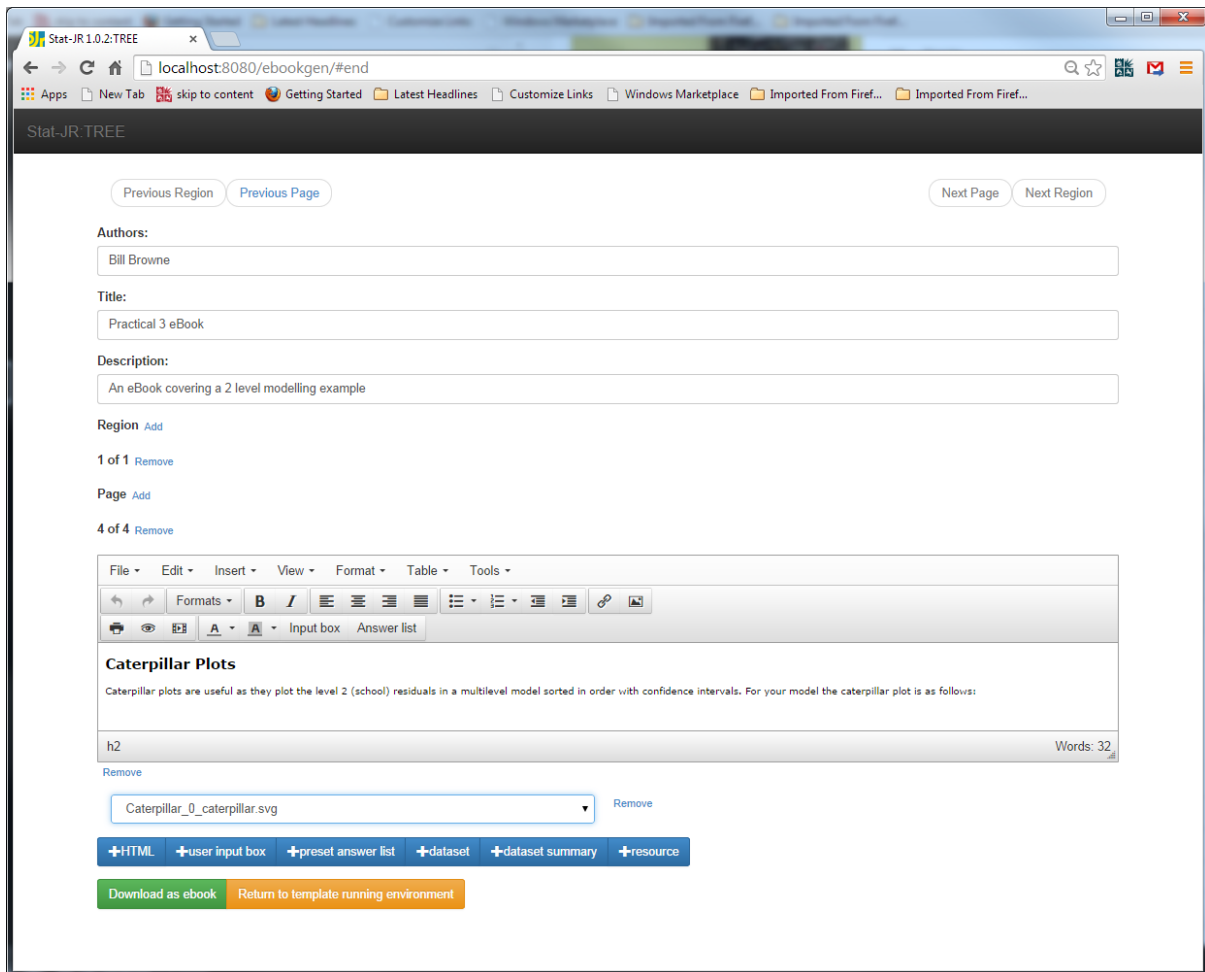
Next we demonstrate the model predictions by adding in a third page with those objects (again two HTML boxes followed by resources) as shown below:

The screenshot shows a web browser window with the URL `localhost:8080/ebookgen/#end`. The page title is "Stat-JR TREE". At the top, there is a navigation bar with "Region Add" and "Page Add" options. Below this, there are two sections for editing content:

- Section 1:** "Predictions for Explanatory Variable 1". The text below the title says: "Below you should see a graph showing lines for each school plotting english against your first explanatory variable (holding all others at the value 0)". Below the text is a text input field containing "h2" and a "Words: 28" indicator. A "Remove" button is located below the input field. A dropdown menu shows "Graph_0_graphxygroup.svg" with a "Remove" button next to it.
- Section 2:** "Predictions for Explanatory Variable 2". The text below the title says: "Similarly if you have 2 predictors in your model below you should see a graph showing lines for each school plotting english against your second explanatory variable (holding all others at the value 0)". Below the text is a text input field containing "h2" and a "Words: 36" indicator. A "Remove" button is located below the input field. A dropdown menu shows "Graph_1_graphxygroup.svg" with a "Remove" button next to it.

At the bottom of the page, there are several utility buttons: "+HTML", "+user input box", "+preset answer list", "+dataset", "+dataset summary", and "+resource". At the very bottom, there are two buttons: "Download as ebook" and "Return to template running environment".

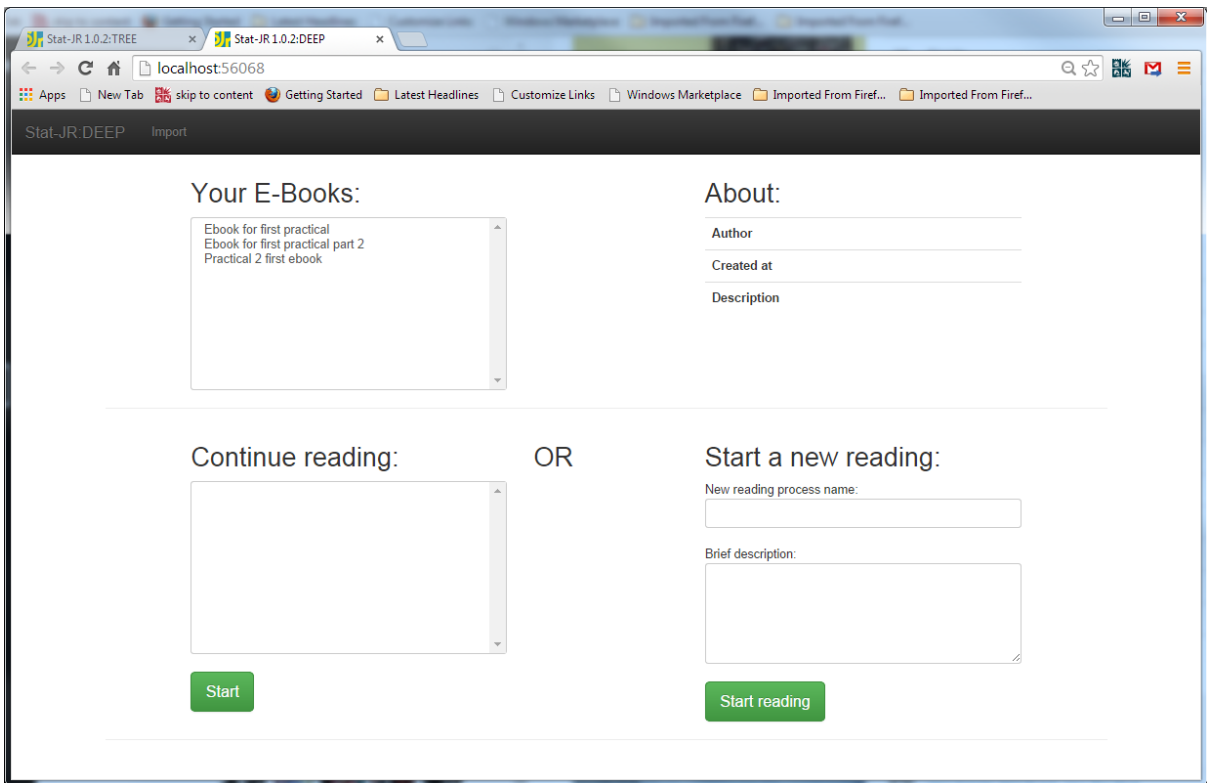
Finally for now we will add a fourth page and include the caterpillar plot thus:



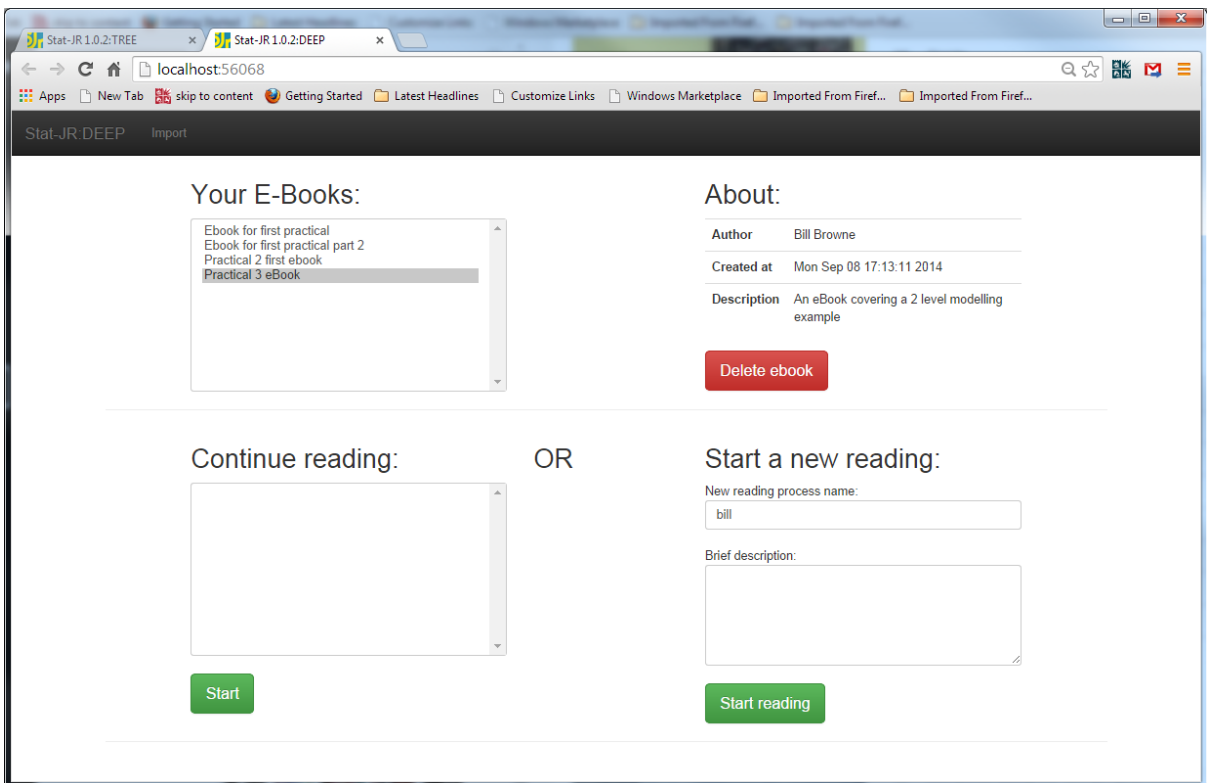
We have now finished our eBook and we can now download this eBook by clicking on the **Download as eBook** button and saving the eBook as **practical3.zip**.

Viewing our first eBook

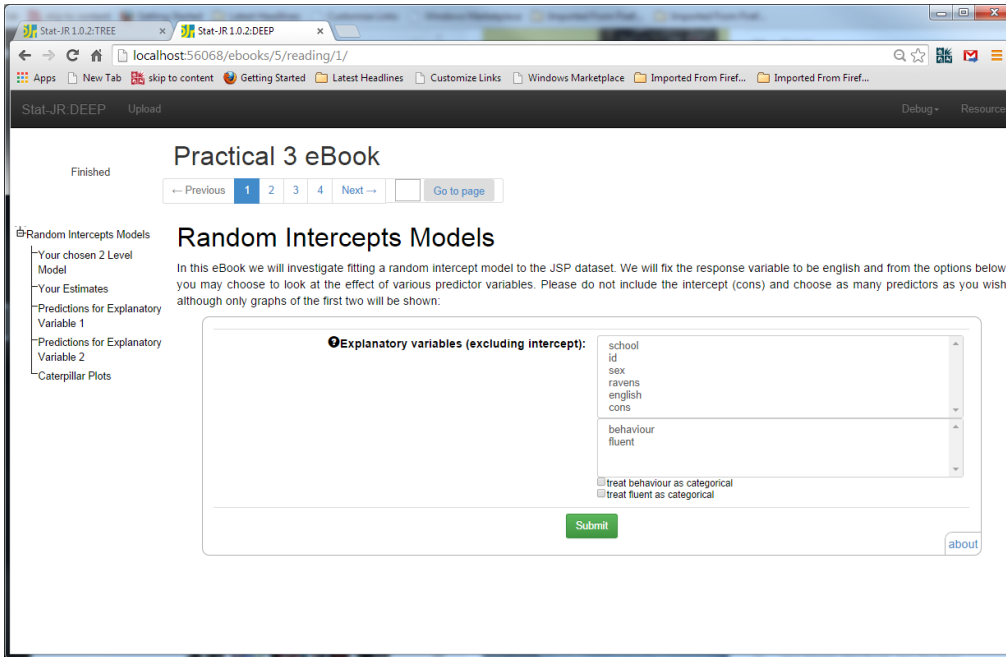
Having created our zip file we are now ready to view our first eBook of this practical. If we load up the Stat-JR DEEP system we will be greeted by the familiar screen:



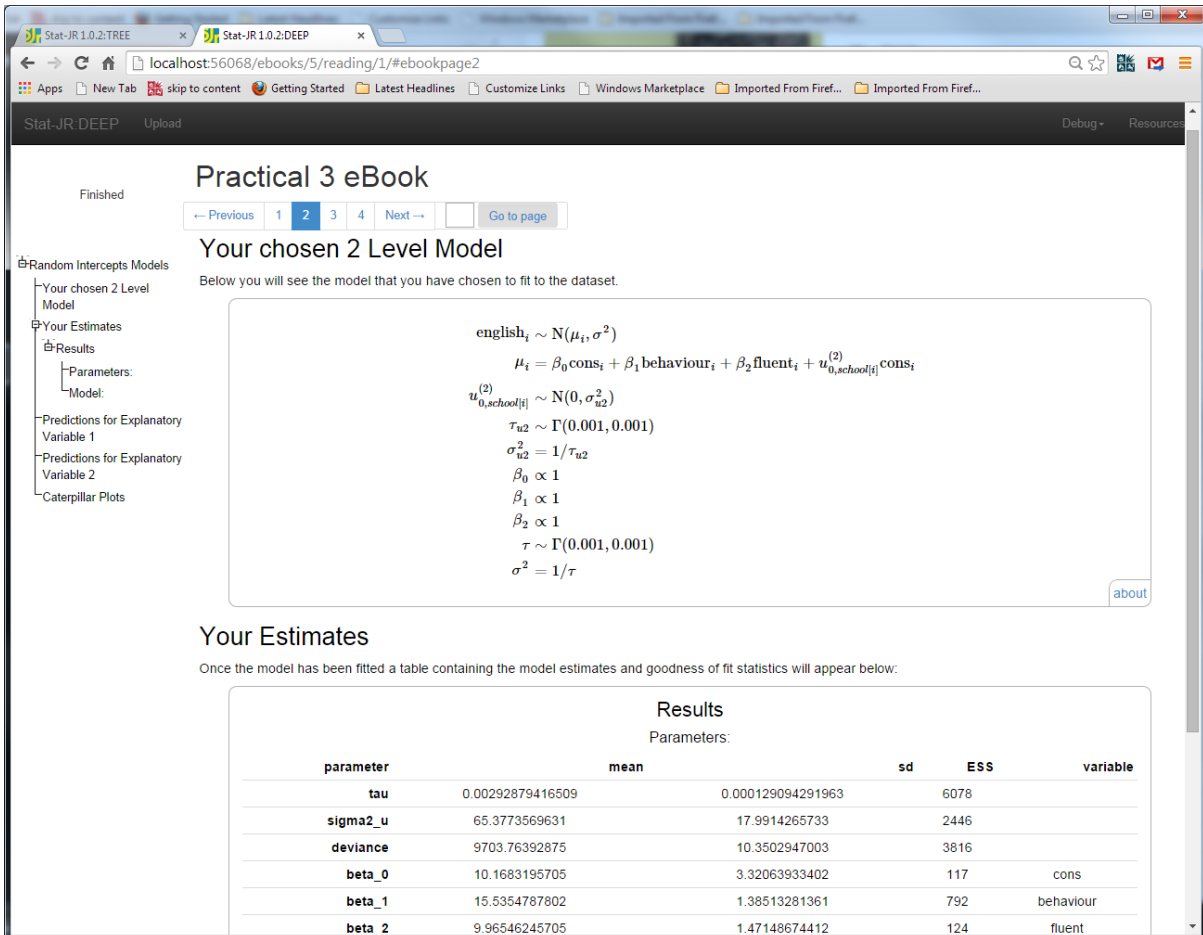
We will select **Import** and find **practical3.zip**. Having imported the ebook we can click on it and it will appear in the list and we can then set up a reading process:



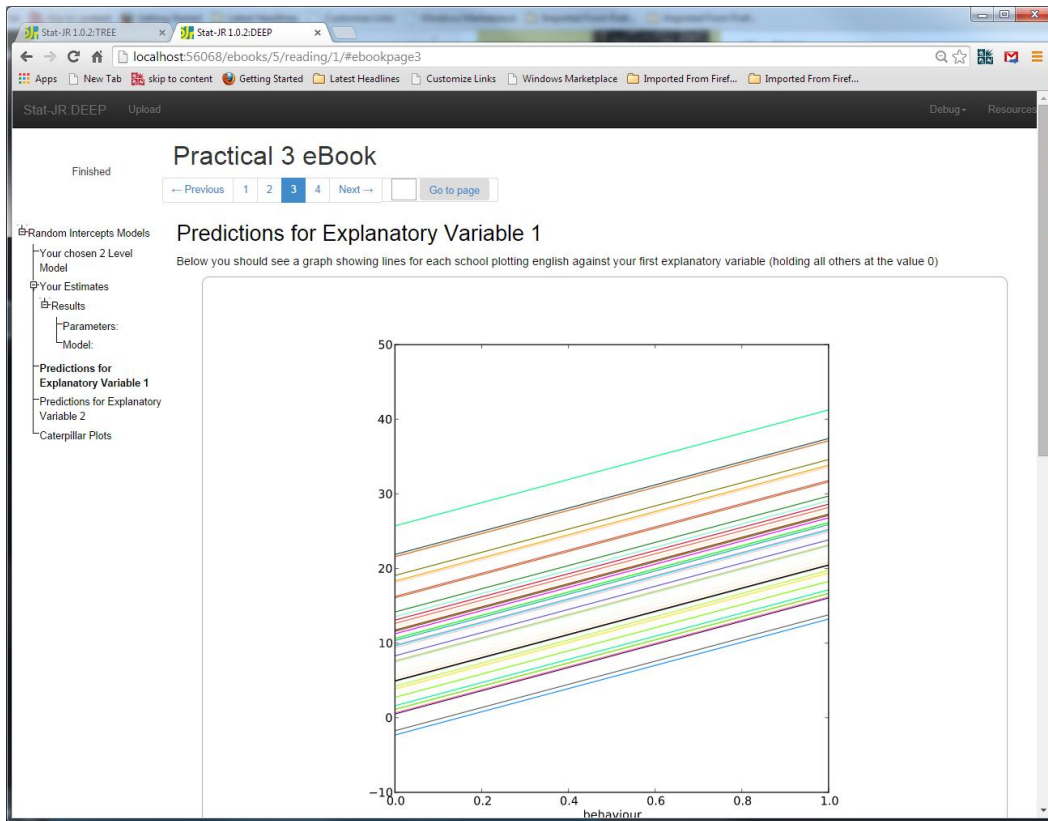
We click on **Start Reading** and are greeted by the first screen and the input box:



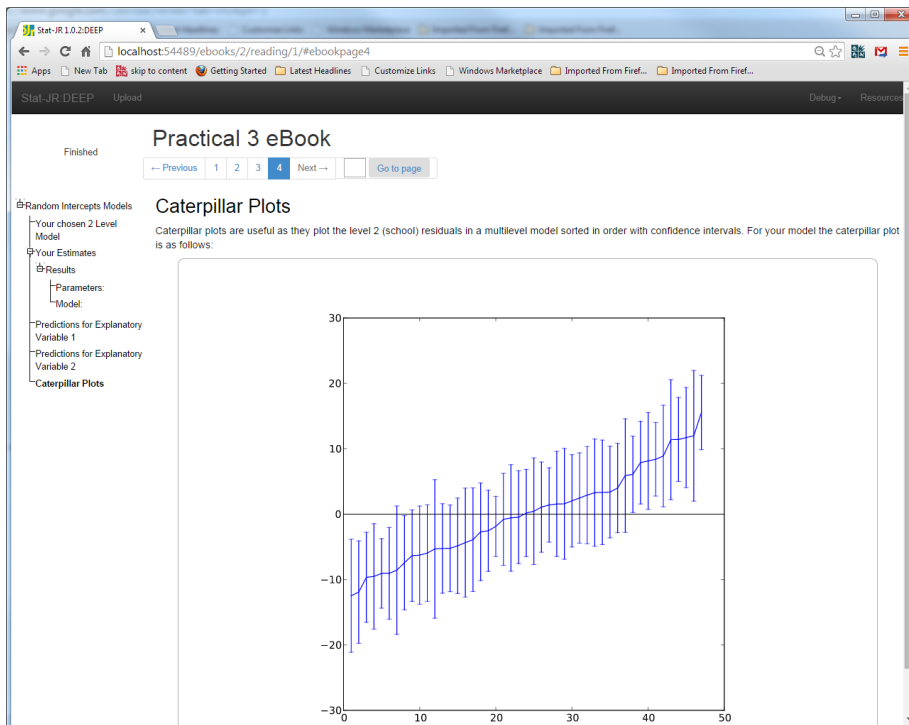
Here I will (for variety) choose **behaviour** and **fluent** (as shown above) as explanatory variables but you can choose appropriate predictors from your own dataset. We then click on **Submit** and DEEP will say “Running Python Script” in the timer in the top-left before eventually saying “Finished”. The results can then be seen on the pages 2 through to 4 with page 2 appearing thus:



Here in the results we see that **behaviour** and **fluent** are both significant predictors and positive effects and this is backed up by the prediction graphs shown on page 3:



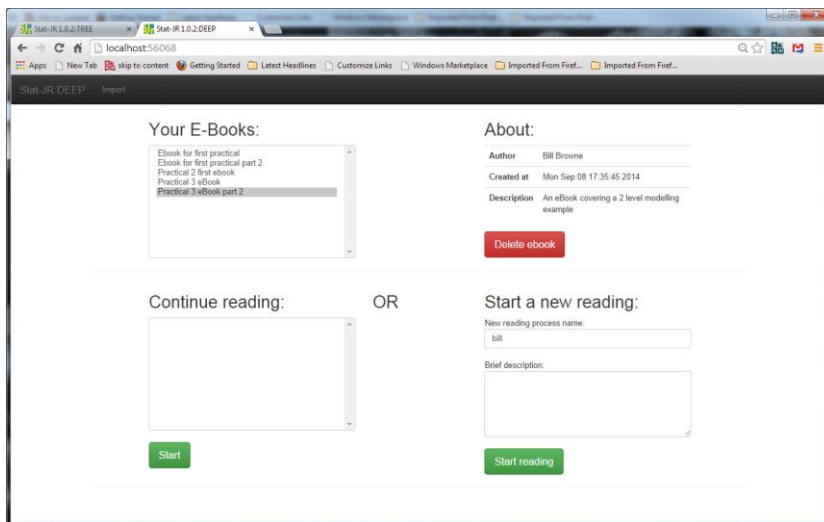
We see **behaviour** here but we can also scroll down to see **fluent**. Finally on page 4 is the caterpillar plot:



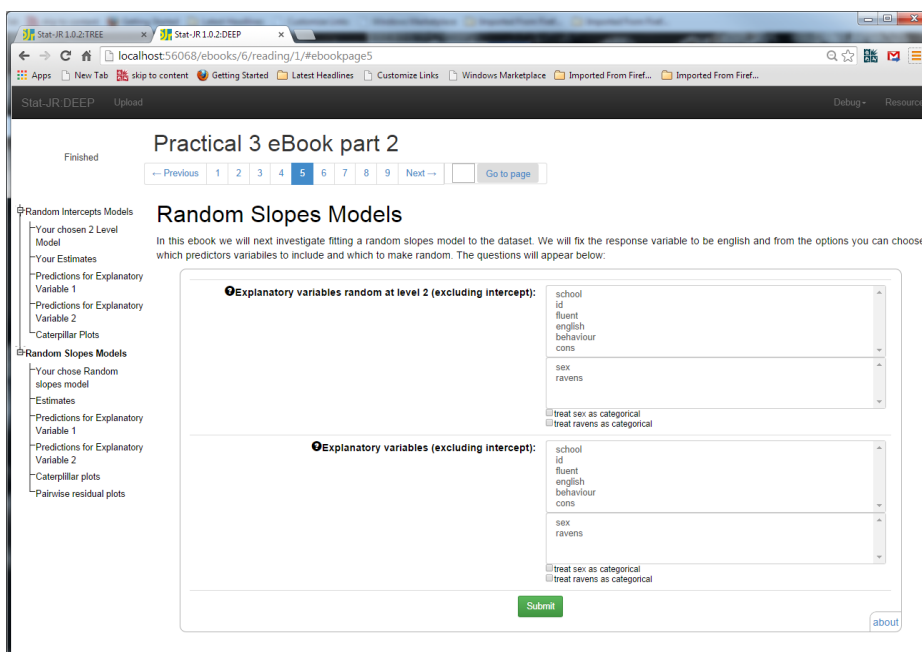
As in previous practicals if you have TREE still open you can now add more material to the eBook and create a longer eBook. One possibility would be to use the same template but fit a random slopes model and create the same four pages for this model. We could perhaps include an additional page with pairwise residual plots as we now have both intercept and slope residuals. Rather than laboriously talk you through this in detail we leave it as an exercise (if you have time) and instead show the output of our attempt which uses a second activity region.

Random slopes screens

Below are the additional pages that I have personally added to extend the eBook. Please try this yourself and see how you get on:



Here is the input screen for the random slopes model where I have allowed the user to include both explanatory variables (x) and those random at level 2 (x_2) (page 5):



Here we have chosen the predictors **sex** and **ravens** for both inputs. Next we see on page 6 the model equations and estimates for this random slopes model:

Stat-JR 1.0.2: TREE x Stat-JR 1.0.2: DEEP x
localhost:56068/ebooks/6/reading/1/#ebookpage6

Stat-JR DEEP Upload Debug Resources

Finished

Practical 3 eBook part 2

← Previous 1 2 3 4 5 **6** 7 8 9 Next → Go to page

Your chose Random slopes model

Below you can see the random slopes model

$$\text{english}_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = \beta_0 \text{cons}_i + \beta_1 \text{sex}_i + \beta_2 \text{ravens}_i + u_{0,\text{school}[i]}^{(2)} \text{cons}_i + u_{1,\text{school}[i]}^{(2)} \text{sex}_i + u_{2,\text{school}[i]}^{(2)} \text{ravens}_i$$

$$\begin{pmatrix} u_{0,\text{school}[i]}^{(2)} \\ u_{1,\text{school}[i]}^{(2)} \\ u_{2,\text{school}[i]}^{(2)} \end{pmatrix} \sim N \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \Omega_u^{(2)} \right]$$

$$\Omega_u^{(2)} \propto 1$$

$$\beta_0 \propto 1$$

$$\beta_1 \propto 1$$

$$\beta_2 \propto 1$$

$$\tau \sim \Gamma(0.001, 0.001)$$

$$\sigma^2 = 1/\tau$$

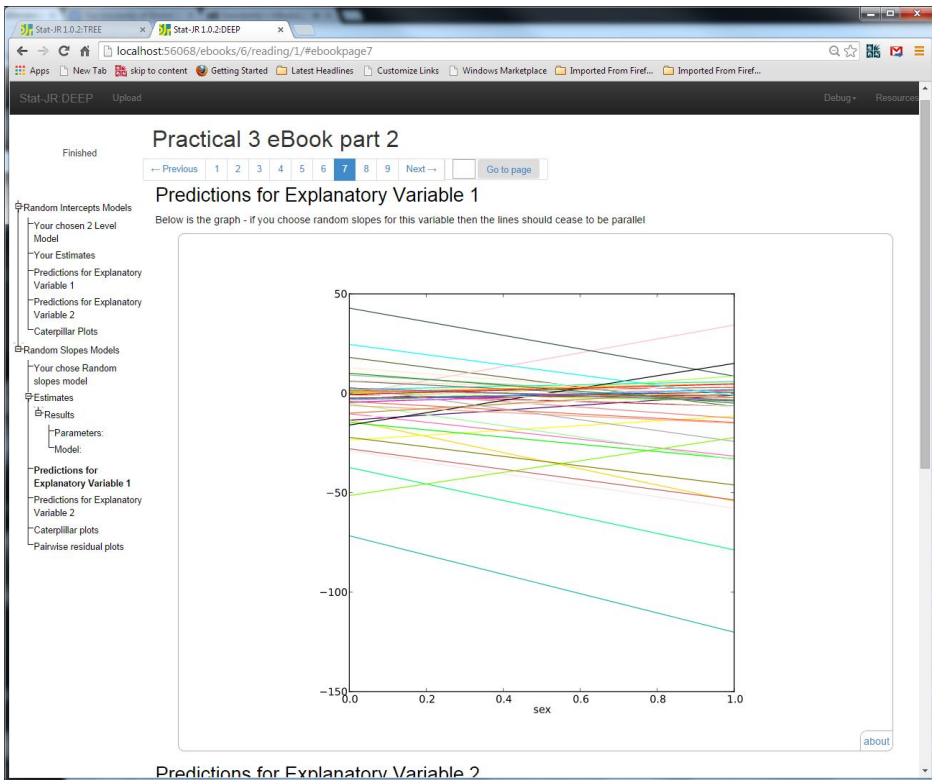
about

Estimates

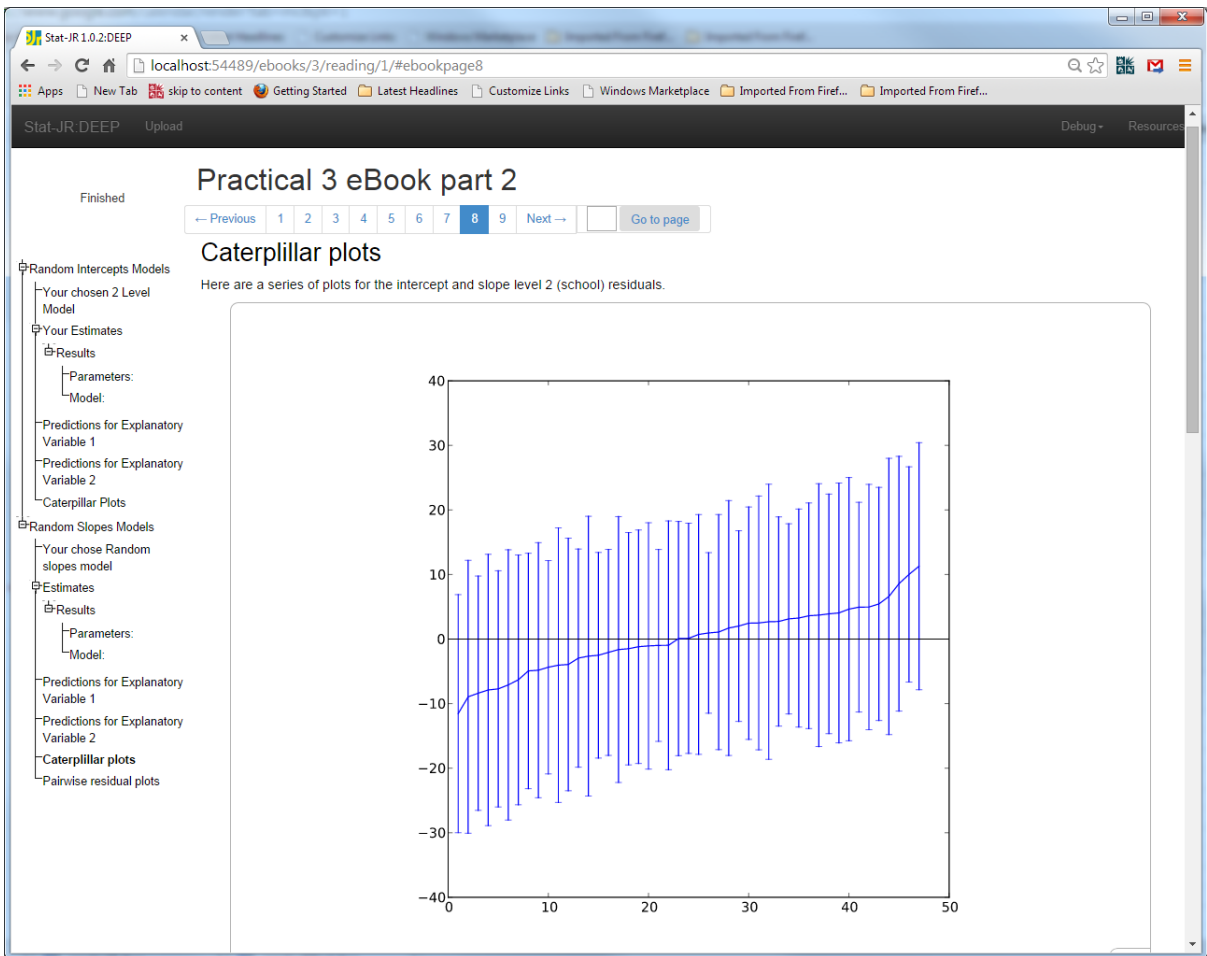
Once the model is finished fitting the model estimates and goodness of fit statistics will appear below:

Results					
Parameters:					
parameter	mean	sd	ESS	variable	
tau	0.00351136492848	0.000154785222959	3872		
deviance	9500.87427898	14.6523881821	601		
omega_u_0	138.062618686	93.9089994664	62		
omega_u_1	-24.6684433268	29.1604191939	161		
omega_u_2	35.2028652738	20.2138734377	305		
omega_u_3	-5.27167849764	3.65724900031	71		
omega_u_4	0.477496319731	1.16123310168	148		

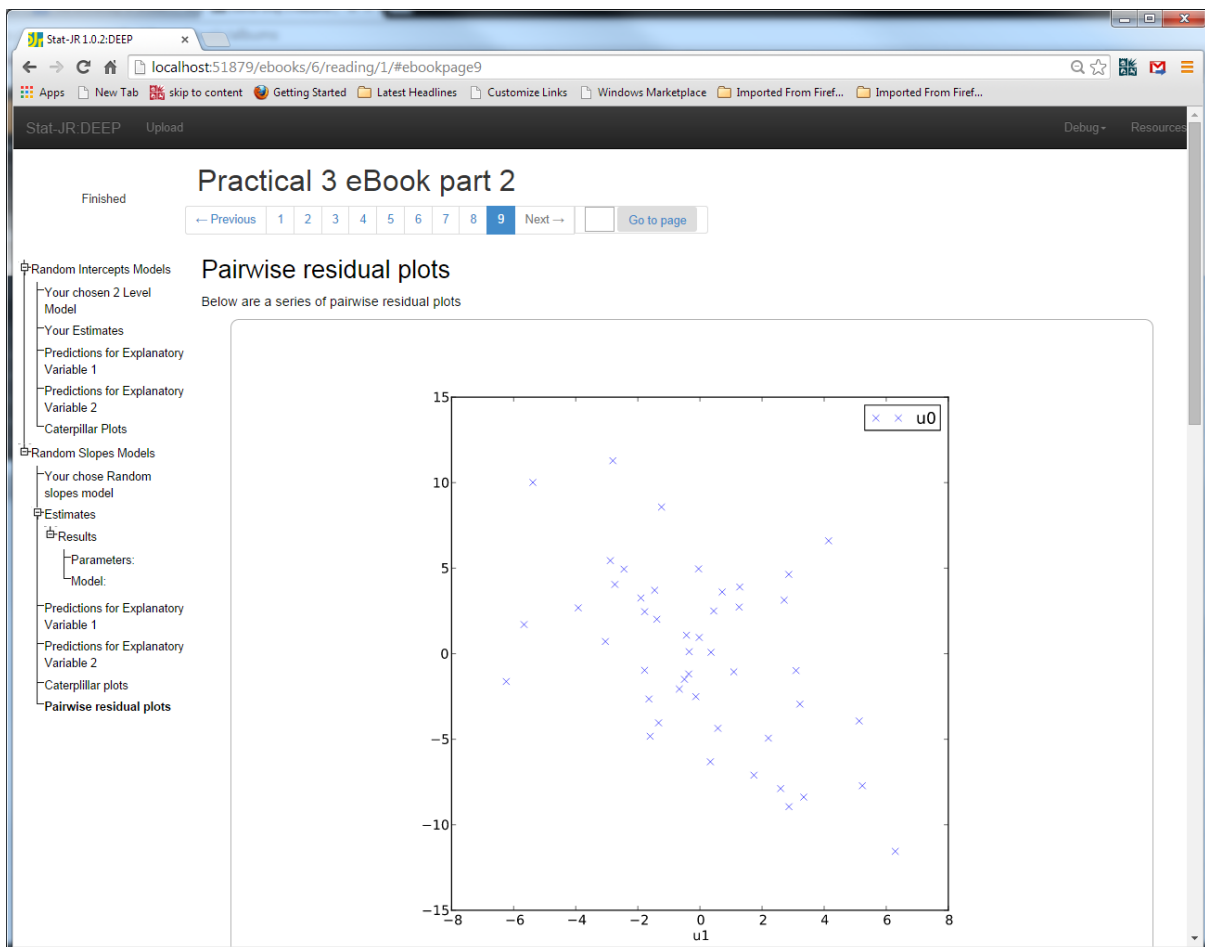
On page 7 we have the prediction lines (note they cross which is now possible as the model is a random slopes model!)



Here are the caterpillar plots (and if you scroll down there are now three sets, for the intercepts, **sex** effects and the **ravens** effects respectively):



Finally here are some pairwise residuals (again three plots) with first the intercepts (u_0) against the **sex** effects (u_1).



This nine page eBook should now be a fairly powerful tool that you can use to investigate as many predictions as you like.

HTML output objects in Templates

One other feature that we might like to incorporate into an eBook is for text that is specific to the user's choice of inputs to appear in an eBook. We will firstly show how this might be achieved in the context of the multilevel modelling examples that we have looked at so far. If we return to the Stat-JR TREE environment you will notice that in the template list is a similarly named template **2LevelPredictallws**. In fact we have written this template specifically for the workshop and if you were to look at the code you would see the following additional lines have been added:

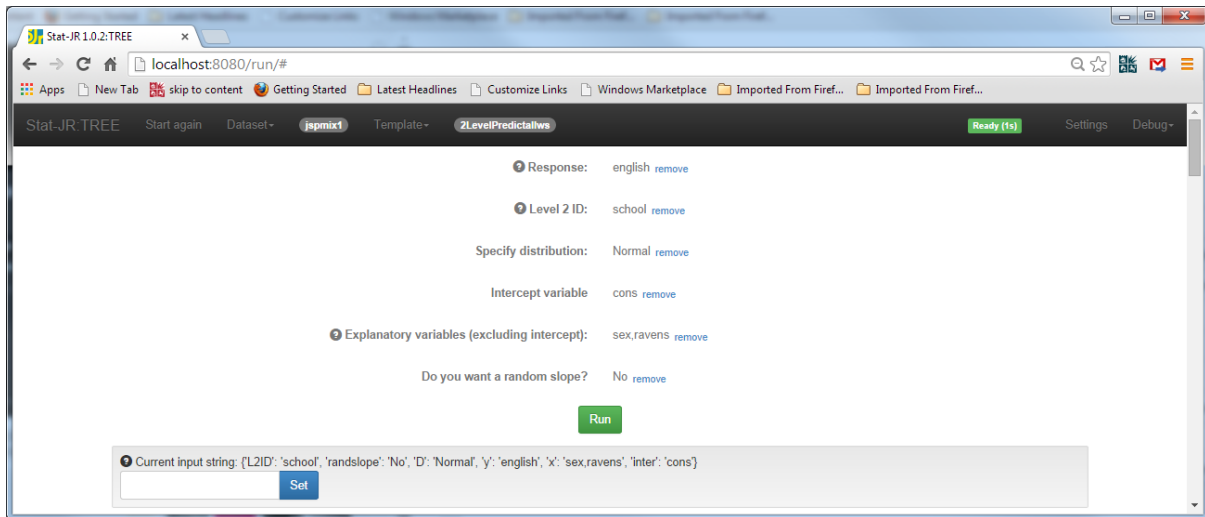
```
if not randslope:
    l2var = m.eng.outputs['ModelResults'].content['paramstats']['mean']['sigma2_u']
    l1var = m.eng.outputs['ModelResults'].content['paramstats']['mean']['sigma2']

    html = '<p>'
    html += 'The Variance partition coefficient (VPC) is the percentage of variability that is
    explained by differences between '
    html += 'level 2 units. In this case '
    html += 'VPC = ' + str(round(l2var,3)) + "/" + str(round(l1var,3)) + "+" +
    str(round(l2var,3)) + ") = " + str(round(l2var/(l1var+l2var),3))
    html += '</p>'
    outputs['vpctext'] = HTMLOutput(html, description = 'Variance partition coefficient')
```

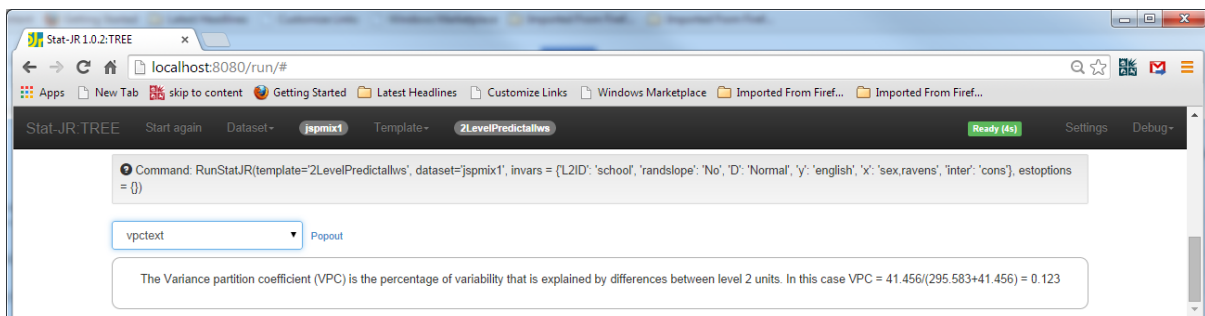
Here the code is run in the super-template after the first model-fitting and will only be run if the model is a random intercepts model. If it is run we first interrogate the **ModelResults** object to retrieve the estimates for the level 1 and level 2 variances, and then construct the string of text that

we wish to be a HTML output. You will see that lines of code build up the string assigned to the object called **html**. If we wish to include actual text then we place it within quotes (single or double) whilst for numerical values or formulas we place them in brackets within a **str** call – note we also use the round function to limit all numbers to 3 decimal places. You'll notice as we are constructing a piece of HTML text that it is all housed with the **<p>** and **</p>** tags to indicate a single paragraph. In the final line we add this object, with name **vpctext**, to the output object list.

If we use this template in TREE then we can set up inputs as earlier:



If we click **Run** we can then hunt the list of objects and find **vpctext**; clicking on it we will see the following:

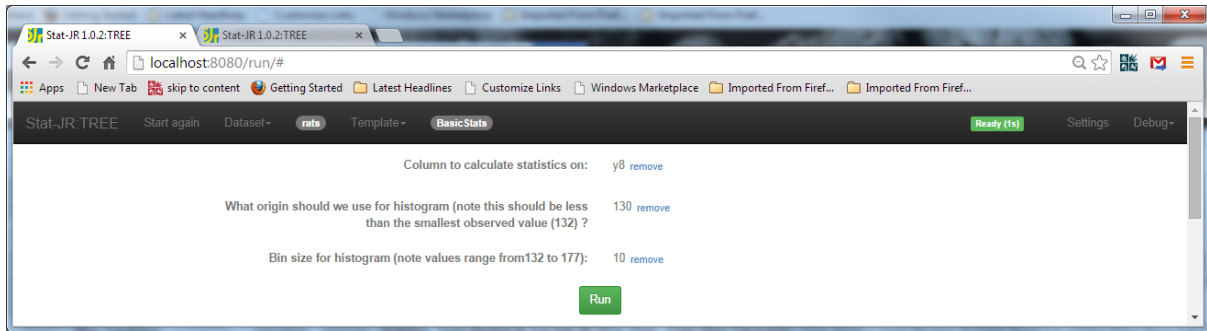


With regard to eBooks this object can be added in the same way as others through the eBook writer. It will appear in the eBook as you see above i.e. as text within a box. We will not here add it to the eBook but instead, to finish up the workshop, demonstrate a couple of more basic statistics templates that might be of interest and which demonstrate somewhat more conditional text.

Basic Statistics Templates

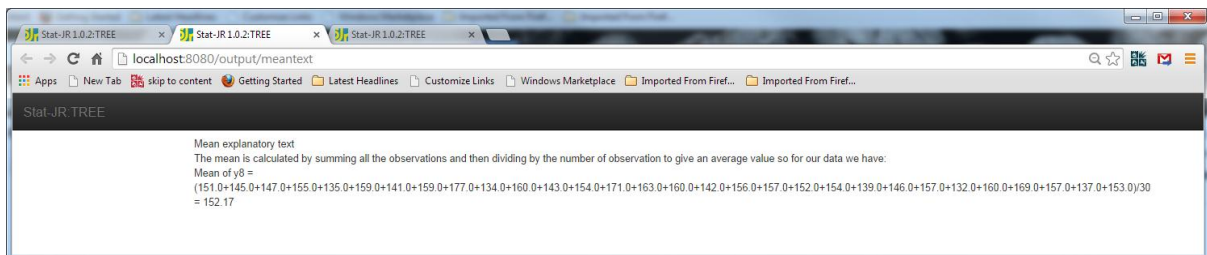
When developing Stat-JR we have spent a lot of time pushing the boundary in terms of statistical methodology development and have created a new estimation engine along with templates that fit models that cannot be fitted in other software. Of course, in terms of a teaching tool, many users will be more interested in the more standard basic statistical functionality. To this end we have written two more basic statistics templates **BasicStats** and **BasicStatsCat**. These templates make much use of the HTML output objects that we saw in the last example. The **BasicStats** template is

designed to be used with a continuous variable and preferably with a small dataset. Here we will use the **rats** dataset (although if you have a small dataset yourself you might like to use it instead). The **rats** dataset consists of the weights of 30 rats measured on 5 weekly occasions from when they were aged 8 days old. We will therefore continue to use Stat-JR TREE but choose **BasicStats** as the template and **rats** as the dataset. I have then chosen the inputs as follows (but feel free to experiment):

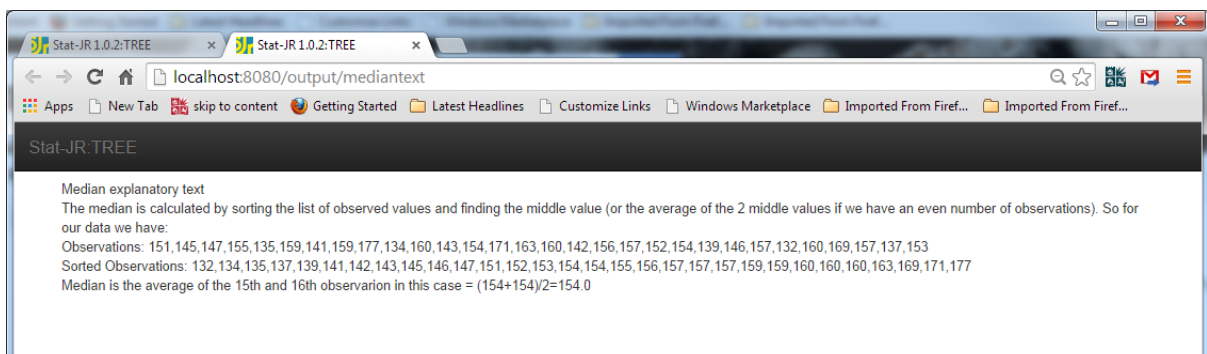


Here we are going to look at the first set of weights, expecting 5 histogram bins (130-140, 140-150, 150-160, 160-170 and 170-180). When we click on **Run** the template executes almost instantaneously and we can view the objects it has created in the object browser.

Firstly if we look at **meantext** we see (after popping out):

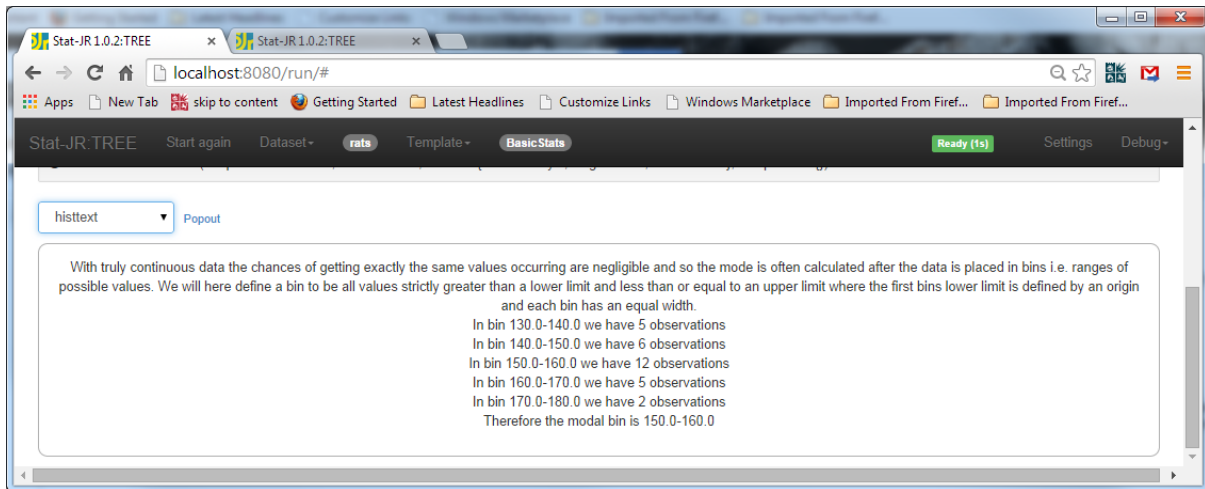


Here you see a textual explanation of how the mean of the variable is calculated along with the mathematics of the calculation. We can see here already that with just 30 observations this is a little cumbersome hence the advantage of small samples. We can next look at the **mediantext** (again after popping out):

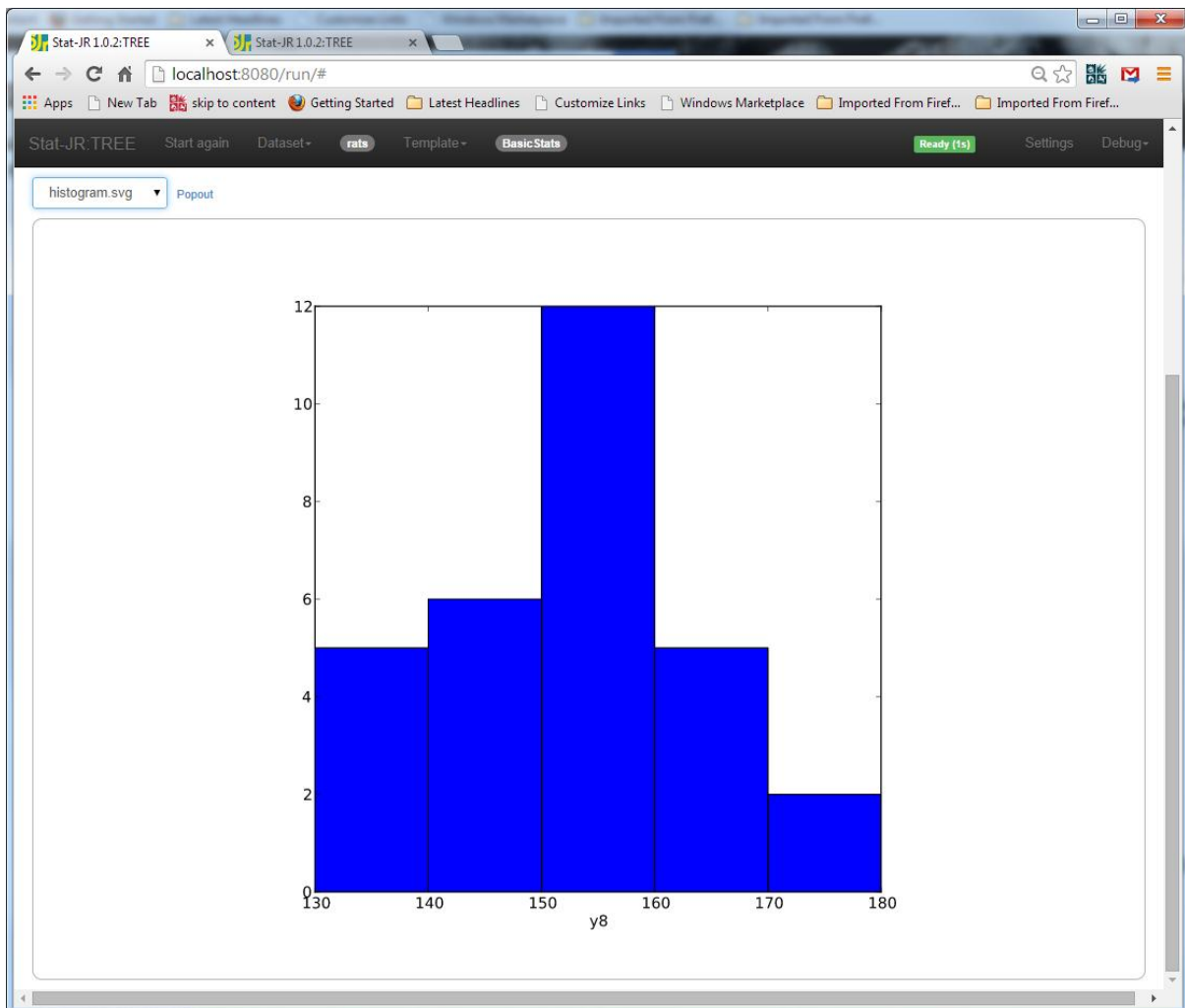


Here we have tried to explain in great detail how the median is calculated by showing how we sort the observations before finding the middle observation. The object **histtext** gives a description of

how one would construct a histogram by hand by identifying the bins and then counting observations in each thus:



We have then tuned the histogram drawing facilities as shown in **histogram.svg** so that the exact same bins are plotted in the graph as we see in text above.

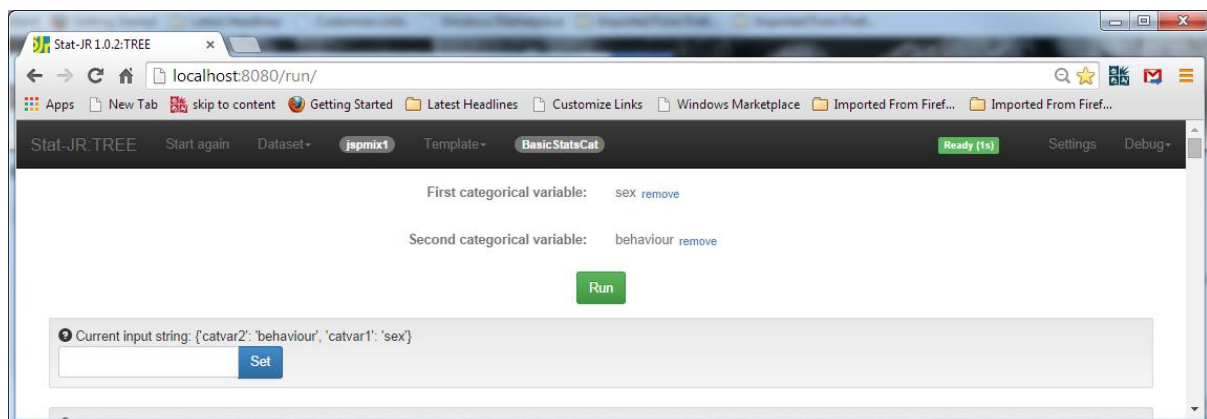


There are also textual outputs for measures of spread (**iqrttext** and **sdtext**) which can be investigated. While we haven't actually constructed an eBook you should feel confident that if you wanted to make a basic statistics eBook then armed with this template and the eBook writer that will be a fairly easy task.

A Chi-squared template

The second basic statistics template is designed for looking at categorical predictors and in fact will perform a chi-squared test for a pair of categorical predictors. We will return to our original **jspmix1** dataset here and choose the template **BasicStatsCat** but if your own dataset has more than one categorical predictor then please try your own dataset. **jspmix1** has a few categorical variables (**sex**, **behaviour**, **fluent** and in theory **school** although it has probably too many categories to really want to be tested via a chi-squared test).

If we choose **sex** and **behaviour** for our two variables then the screen will look as follows:



If we next **Run** the template then it will create a couple of smaller textual outputs but the main output is called **chisq** which we can choose and popout as we see below:

Stat-JR. TREE

Cross-tabulation explanatory text
To do a chi-squared test we start by tabulated observed counts and totals:

Observed	sex=0	sex=1	Total
behaviour=0	97	151	248
behaviour=1	474	397	871
Total	571	548	1119

We can therefore work out the expected counts from the margins of the observed data

And so we expect

$E(\text{sex}=0, \text{behaviour}=0) = \text{Total sex}=0 * \text{Total behaviour}=0 / \text{grand total} = 571 * 248 / 1119 = 126.55$
 $E(\text{sex}=1, \text{behaviour}=0) = \text{Total sex}=1 * \text{Total behaviour}=0 / \text{grand total} = 548 * 248 / 1119 = 121.45$
 $E(\text{sex}=0, \text{behaviour}=1) = \text{Total sex}=0 * \text{Total behaviour}=1 / \text{grand total} = 571 * 871 / 1119 = 444.45$
 $E(\text{sex}=1, \text{behaviour}=1) = \text{Total sex}=1 * \text{Total behaviour}=1 / \text{grand total} = 548 * 871 / 1119 = 426.55$

So the table of expected counts is

Expected	sex=0	sex=1	Total
behaviour=0	126.55	121.45	248.0
behaviour=1	444.45	426.55	871.0
Total	571.0	548.0	1119.0

We next look at differences between what we observe and expect in each cell. We square these values so that every difference is positive and scale by the expected counts so that more frequently expected cells are not overly influential. So for example for sex=0, behaviour=0 $(O-E)^2/E = (97-126.5487042)^2/126.5487042=6.9$. This statistic is shown in tabular form below

$(O-E)^2/E$	sex=0	sex=1
behaviour=0	6.9	7.19
behaviour=1	11.96	2.05

The test statistic for a chi-squared test is found by summing the values of this table so

$\text{Chisq} = 6.9 + 7.19 + 11.96 + 2.05 = 18.1$

This is compared with a chi-squared table with degrees of freedom = (number of columns - 1) x (number of rows - 1) = $(2-1) \times (2-1) = 1$

Looking up the chi-squared table the value for $P=0.05$ is 3.84 and for $P=0.01$ = 6.63

as $18.1 > 6.63$ our P value is less than 0.01 and we have strong evidence to reject the null hypothesis (at the $P=0.01$ level)

The p-value is in fact less than 0.0001

Here the output shows how we tabulate the dataset and then create expected counts, ending up with a test statistic. It then compares the values with a chi-squared table and, depending on the outcome, rejects or fails to reject the null hypothesis. Although this is already quite a comprehensive summary of a chi-squared test it is still to some degree proof of concept and for example might benefit from stating what the hypotheses are for testing. Of course the beauty of the Stat-JR system is that you have access to the code and so can modify the templates if you wish. Whilst we will not inspect the code in detail, if you look at the template file you will nevertheless see that it largely consists of creating the HTML code to produce the text and tables you see in the above table. The tables are produced using tabular tags in html and **for** loops are used within Python to cope with the potential for differing numbers of categories. **If** statements are used to allow conditional output: for example displaying different text if the P value is significant or not. Although the prospect of writing this code from scratch might seem rather daunting you might like to play around with the **BasicStats** template and see how you can change the look and feel of the output. Note here it is sensible to first save the template to a different name.

In a one day workshop we have only really scratched the surface of what is possible with the Stat-JR system. We nevertheless hope we have shown you that writing 'simple' eBooks is fairly easy to pick up and a useful skill to have with potential use for your research and/or teaching. We hope also that you have seen that some aspects of eBook-writing are much less straightforward and we would very much appreciate your input here as we continue to develop our eBook writing system and its successor. In particular if you identify functionality that (i) you would like to be available and (ii) you

would like to be easier to implement yourself perhaps through tools like the eBook writer then we'd like to hear from you.