Practical 3: Making workflows to support the LEMMA training materials

Overview

This practical is somewhat different to the practicals that we have gone through so far in that all the workflows have been written for you and the idea is simply that you follow them and learn some further features of the workflow system as you go. We will be using a very long workflow that aims to use in-house functionality (templates using Python procedures and Stat-JR's in-house eStat engine) to try to produce the equivalent analysis presented in the MLwiN practical for Module 3 of the LEMMA training materials (written by Fiona Steele). It is therefore best to have a copy of the practical with you as you run through the workflow (the website supporting the online LEMMA training materials can be found here: http://www.bristol.ac.uk/cmm/learning/online-course/index.html).

Introducing *procedures*

To begin you will need to start up the Stat-JR workflow system (via the wf.cmd executable), or return to the workflow screen, if it's already open, and press **Clear**. The screen will look as follows:

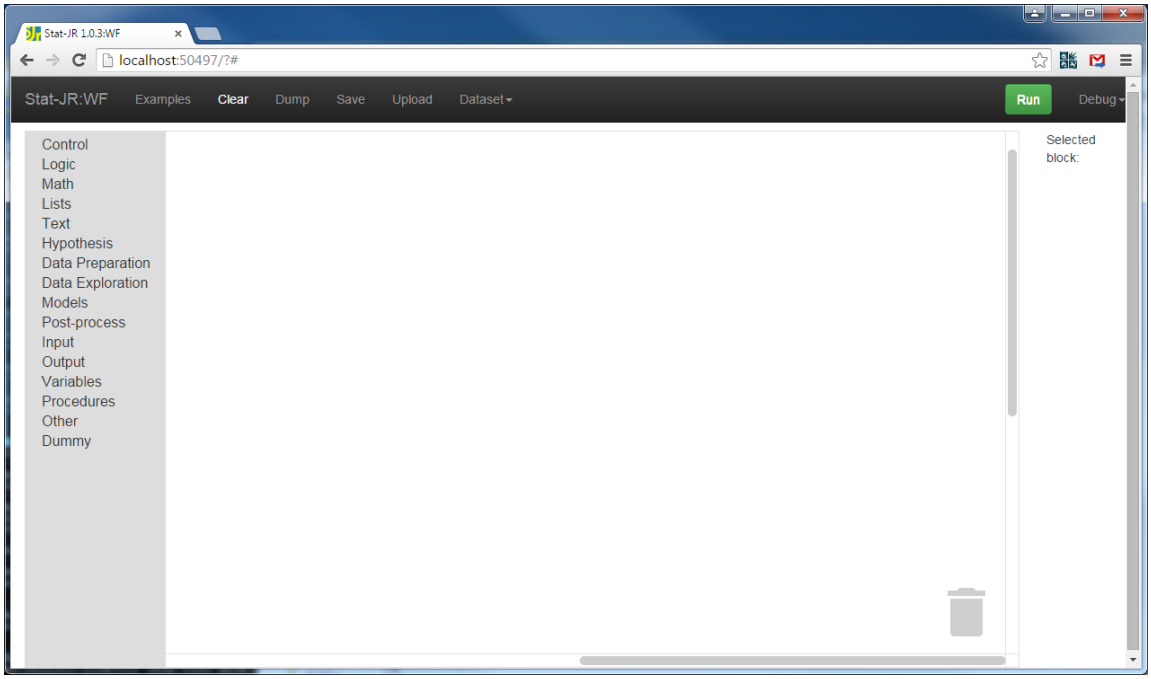

Figure 1

We will be using a workflow with the name *prac3final.xml*, so click on the **Upload** button and select it as shown below:

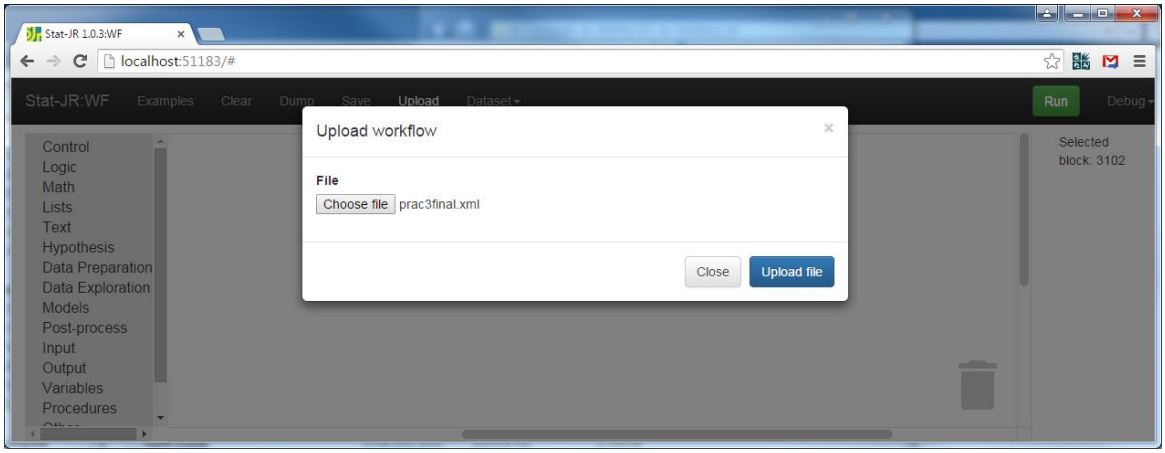

Figure 2

Clicking on **Upload file** displays the relevant workflow:

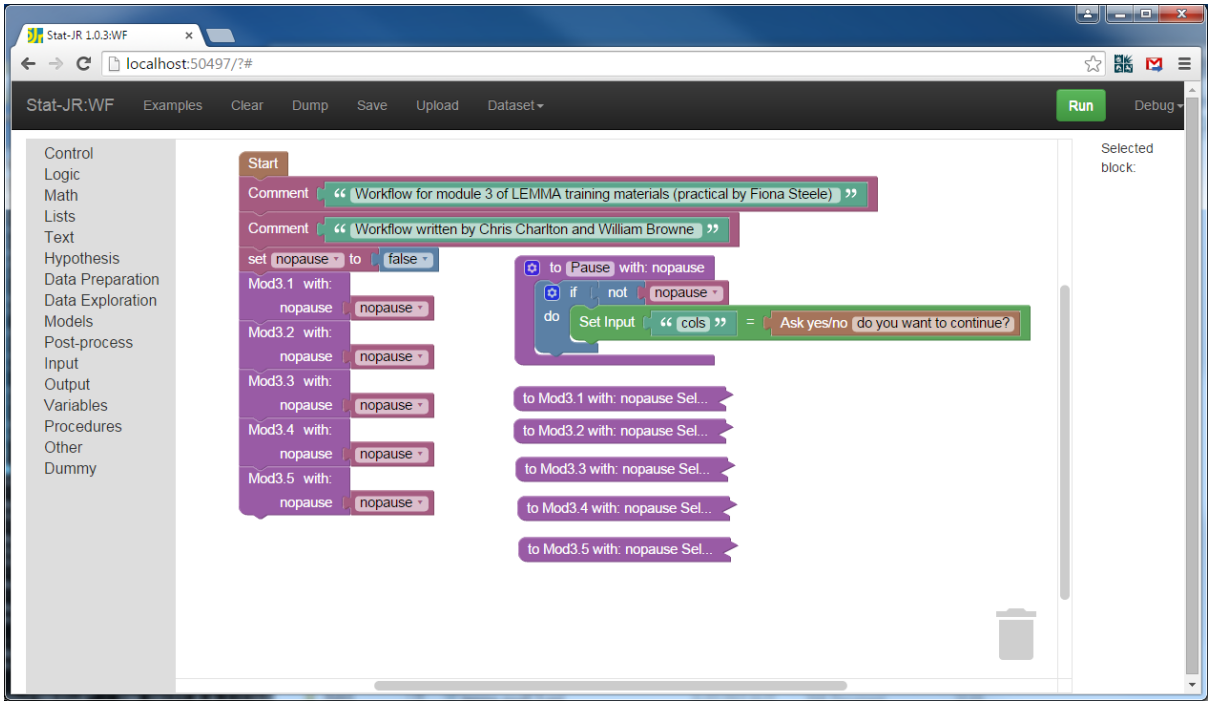

Figure 3

This looks somewhat different to the workflows you have seen previously. There appears to be a rather short workflow contiguous to the *Start* block consisting of two *Comment* blocks, one *set <variable>* block and then five purple blocks that we haven't yet come across. There are then some other blocks over to the right, which do not appear to be connected to the main workflow.

The purple blocks relate to *procedures*; these are available from the **Procedures** menu which, if you click on it, looks as follows:

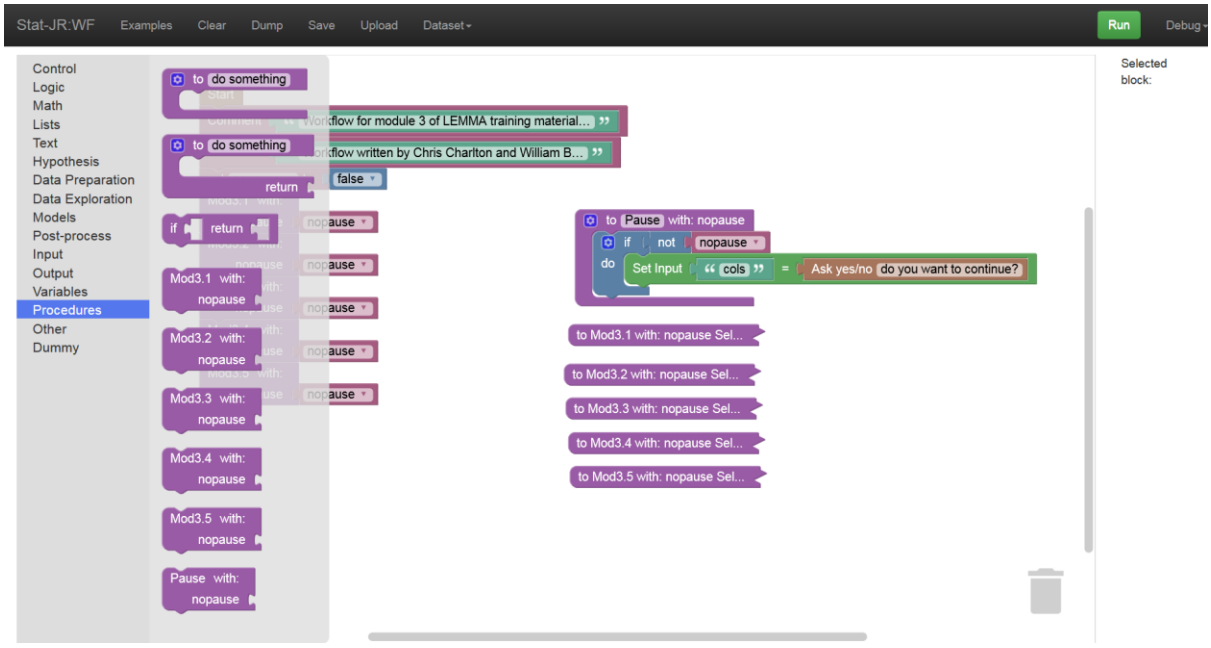

Figure 4

The top three blocks are always found in this list, whilst the others appeared in the list as we added *procedure* blocks when building our workflow – we describe this further below. We used the top block to create the procedures in this workflow. It looks similar to the *grouping* block we used in Practical 2 – we can change the name away from "do something" and place some blocks within the 'mouth' of the *procedure* block which will be executed when the workflow reaches it – but one important difference is that it's call-able: we've been able to place it away from the main workflow (on the right) because we can call it from there.

To demonstrate, if we bring another of these onto the central workflow pane, and give it a name in place of the default *"do something"* (we've chosen *"run a model"* in this example), another purple block (with *"run a model"* written on it) has now appeared at the bottom of the **Procedures** list:

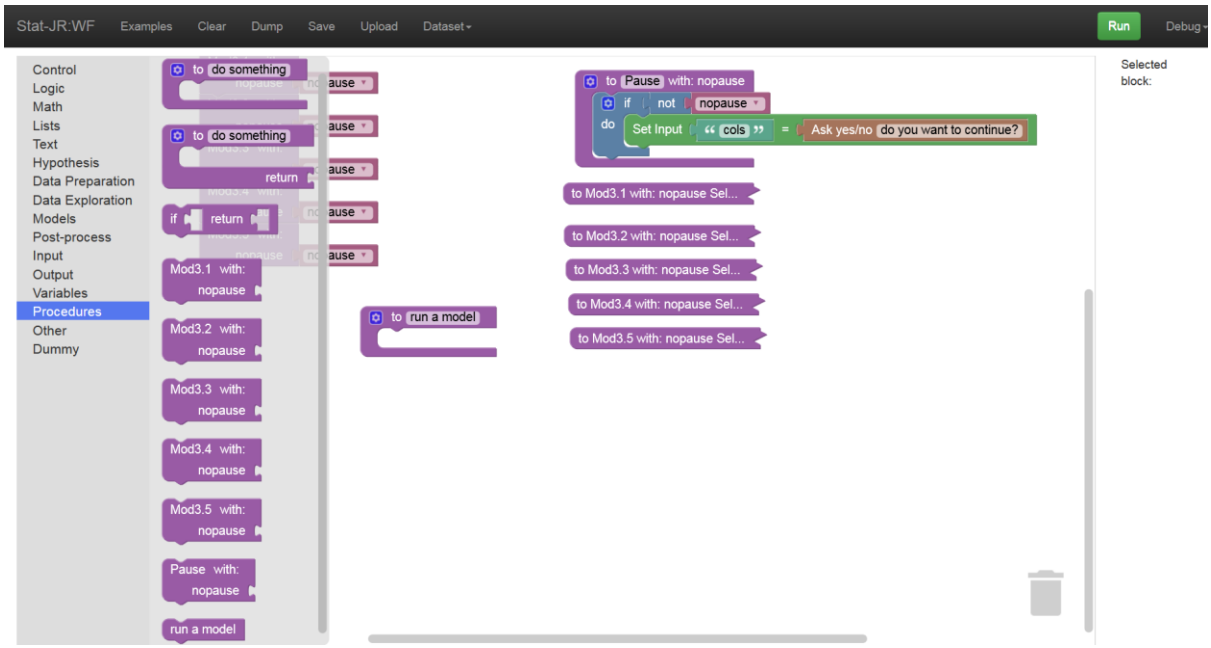

Figure 5

If we next click on the blue button on the procedure block we have just introduced, we can modify the procedure block, requesting that it accept a named input when called – here, in keeping with a number of the other procedures we produced when originally writing this workflow, we've called the named input *"nopause":*

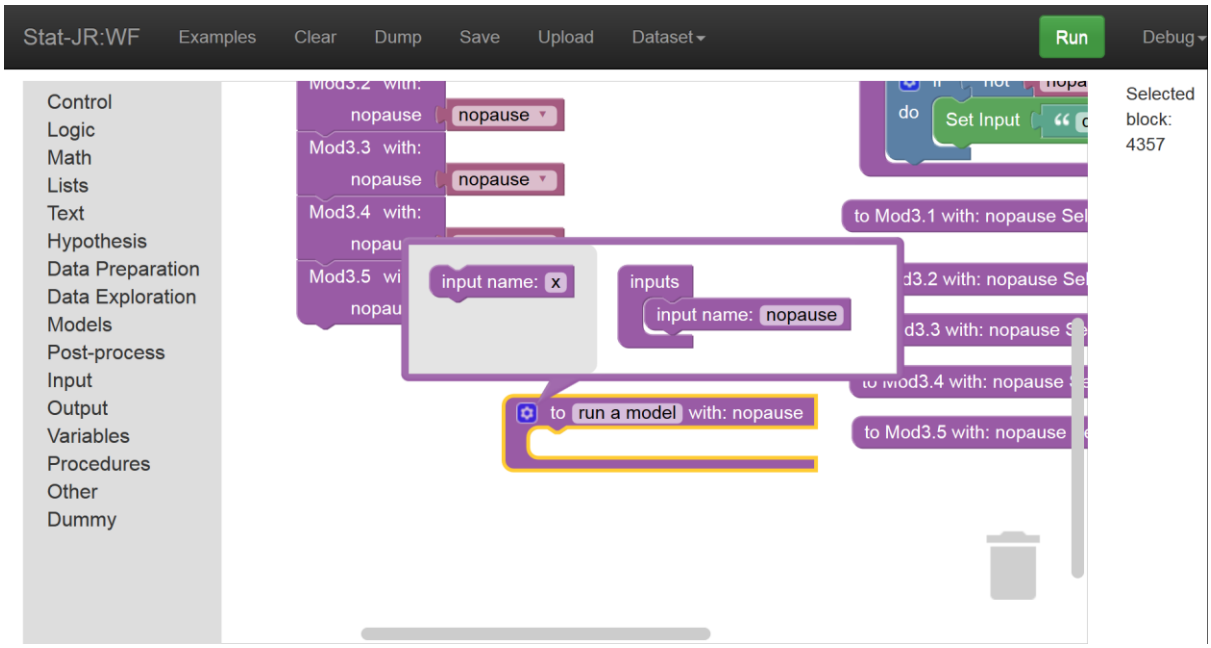

Figure 6

Now if we look again at the list of blocks under the **Procedures** list we see the block at the bottom has been modified, and now looks just like the ones immediately above it:

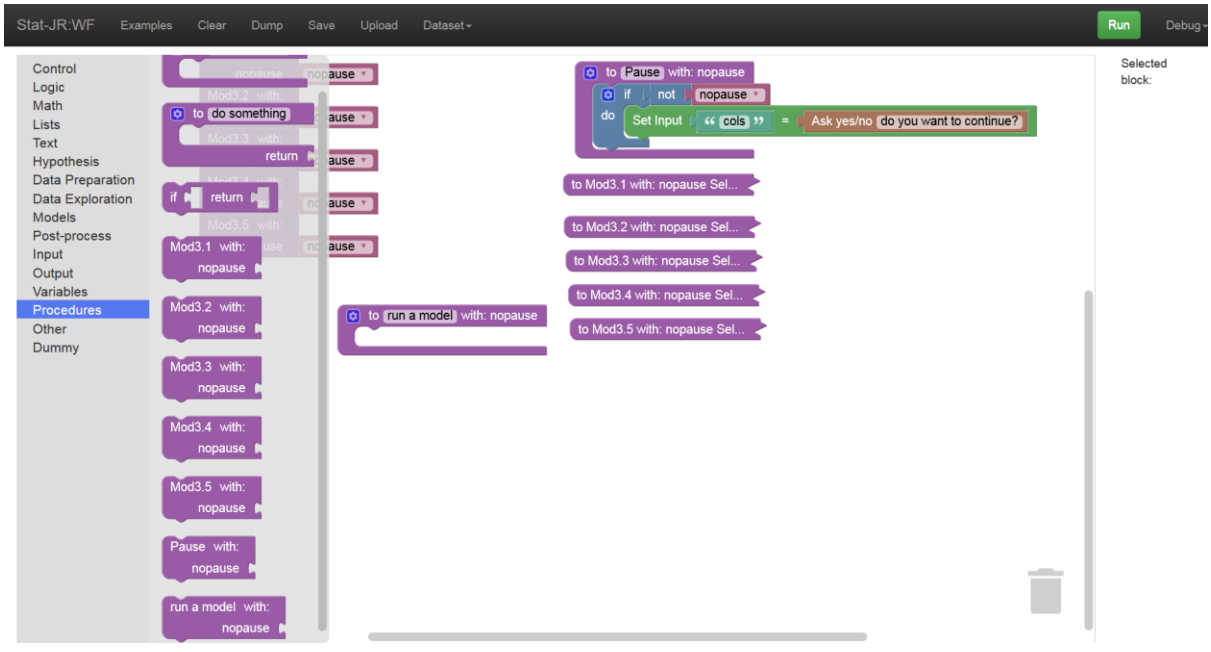

Figure 7

This new block (the one which has appeared at the bottom of the left-hand list) is the block we would use if we wished to call the procedure we've just defined (although not fully: we haven't

added any blocks in the procedure itself in this simple example). Since we've modified it such that it takes an input (called *"nopause"*), we can use this input to control internal aspects of the procedure.

Let's bin the procedure we've just made (you'll notice the **Procedures** list is modified appropriately) and turn our attention back to the original workflow. It has six procedures defined: one named *Pause* and five named *Mod3.1* through to *Mod3.5*: as you might expect, these five procedures carry out all the instructions for each of the five numbered sections within the LEMMA practical module. These five procedures are currently collapsed on the right-hand side of the workflow (if you right-click and **Expand** one you will see it is very long). The top procedure, called *Pause*, is a lot shorter, though, and not collapsed. Like the dummy procedure we produced for illustration a moment ago, it takes an input called *nopause*; looking inside the *Pause* procedure we can see that it evaluates this input via an *if-do* block (as used in Practical 2). The use of *not* when evaluating the *nopause* item means that if *nopause* is 'false' then it will set an input called *"cols"* (the name we've given it here is incidental) to be a Boolean yes/no question asking the user whether they wish to continue or not.

The calls to the *Pause* procedure are within the large *Mod3…* procedures (the ones collapsed when you opened the workflow), whilst the *nopause* input it uses is defined in the main workflow, just three blocks below the *Start* block. It's currently set to 'false', which means that the user would be prompted with the question "do you want to continue?" whenever the *pause* procedure was called. Toggling nopause to 'true', therefore, would mean the whole workflow would run without pausing when you click **Run**. We do not advise doing this here as the whole workflow consists of many model fits and will take a good while to complete (and no output will appear until it ends!)

So here a procedure is being used not for data analytical purposes, but to simply modify the interface for the user; *cf.* the other procedures (beginning *Mod3…*), which are used for data analytical purposes.

Below we run through each section of the workflow in turn. These sections don't pass any objects between each other and so we can interrogate each one separately.

LEMMA P3.1: Regression with a single continuous explanatory variable

Let's look at the first section of LEMMA Module 3. We can actually disconnect all procedure calls from the *Start* block aside from the one representing the section we are interested in, which means we won't execute the whole workflow after pressing **Run**, rather we will just execute the part we are currently focusing on. So let's do this for procedure *Mod3.1*, as follows:

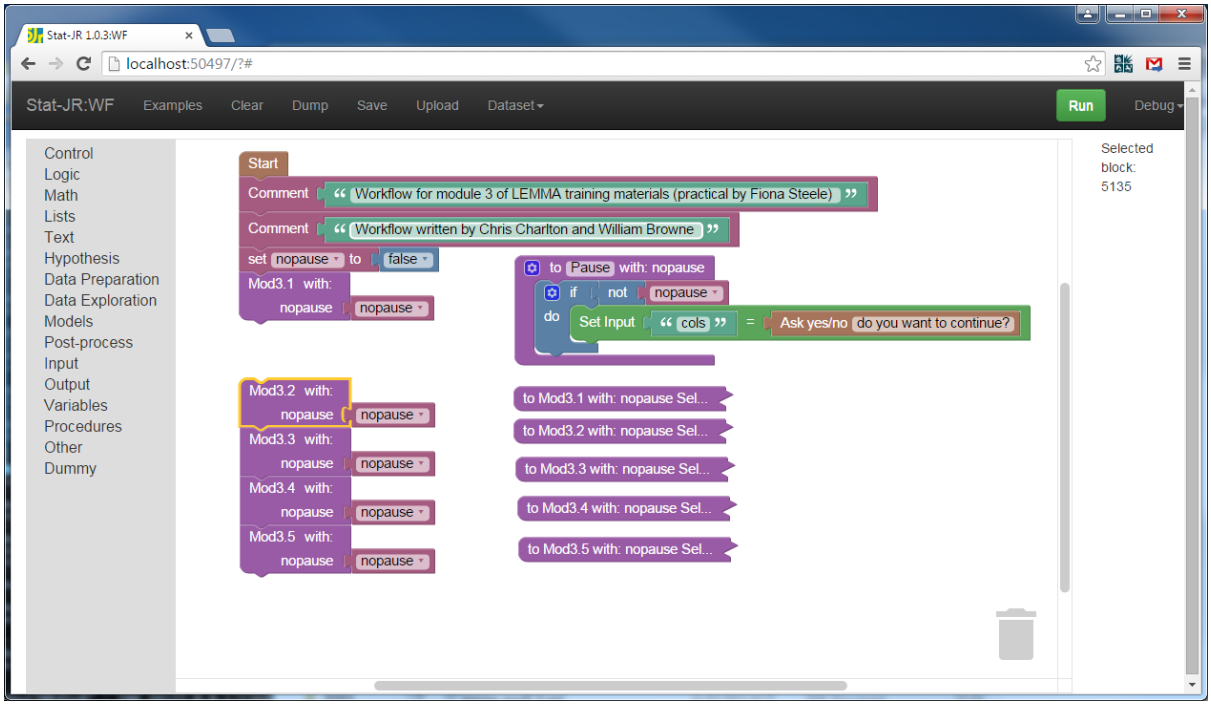

Figure 8

Let's expand the procedure block for *Mod3.1* (here, to better see it in the central workflow pane so that it isn't obscured on the right-hand side, I've moved it to the left and placed it below the other procedure calls I just disconnected):

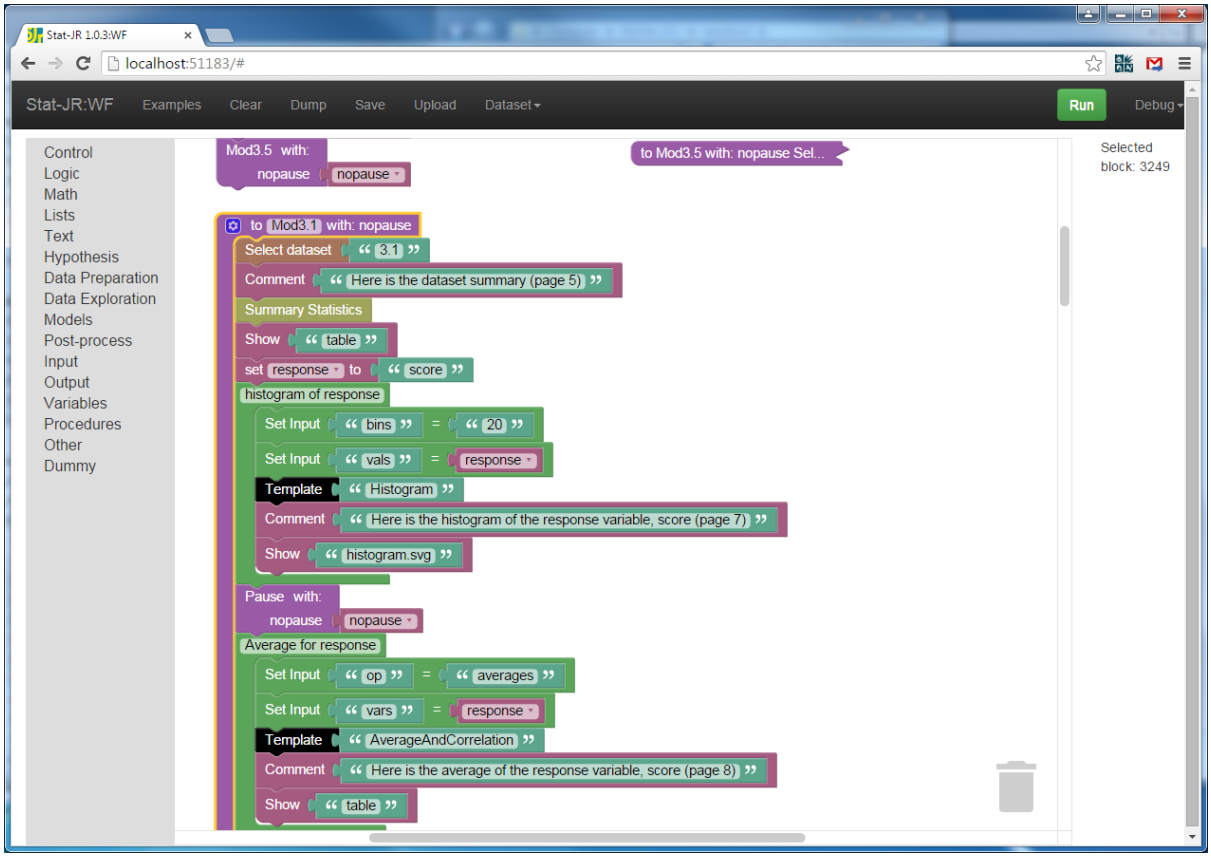

Figure 9

If you click **Run** the workflow will execute the procedure *Mod3.1* and pause each time a *Pause* is reached.

You will see that the procedure consists of green *grouping* blocks identifying what each section of code does, with occasional *Comment* blocks. These *Comment* blocks link the workflow to the page numbers in the LEMMA documentation, and the text in the *Comment* blocks will be printed out in the output. The workflow is also punctuated by calls to the *Pause* procedure which, as described above, will pause the workflow and present the user with a prompt asking whether they would like it to continue or not. Otherwise, the functional structure of the workflow is much like that encountered in the last two practicals: *Set Input* blocks specifying the inputs for subsequent *Template* executions, and *Show* blocks displaying certain outputs from those executions.

Clicking **Run** you will see the following output if you scroll down:

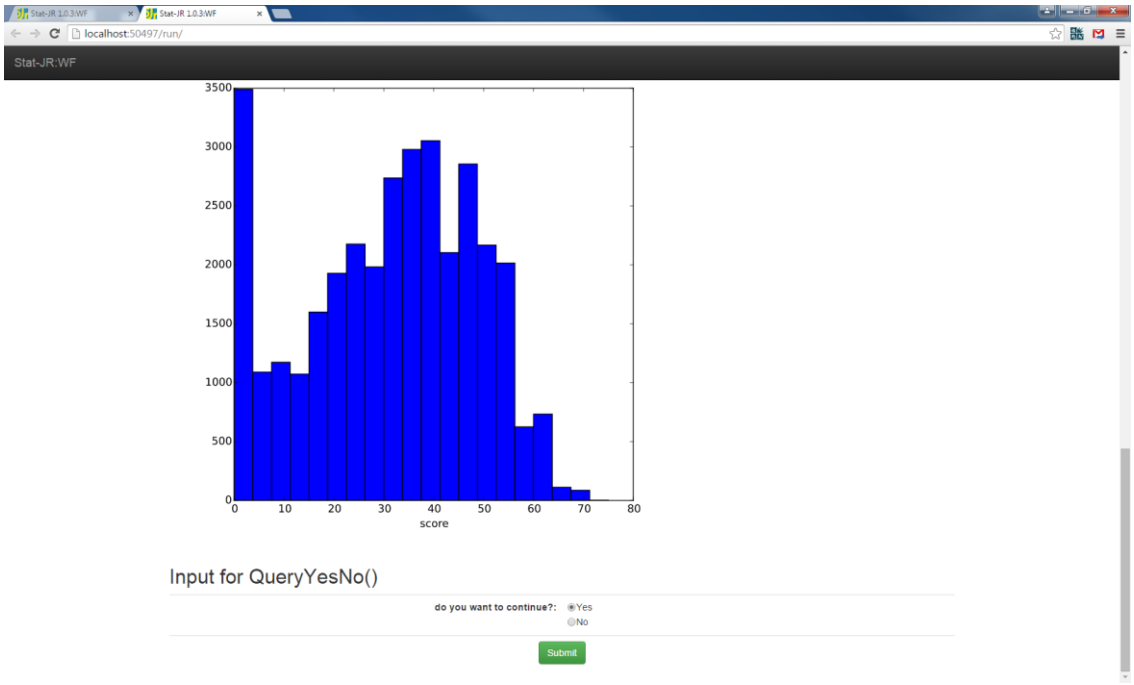

Figure 10

Here is the histogram from page 7 of the MLwiN practical, and our first pause. Clicking **Submit** will take you to the next output and you can do this at your own pace and observe the code in the workflow that does this. If you have two screens (or two windows) you can have the workflow-code window and the workflow-running window up together to see the correspondence. Note that you may need to keep scrolling down the window (this doesn't happen automatically).

If you persevere through all the clicking, the last output will correspond to the histogram on page 24 of the practical.

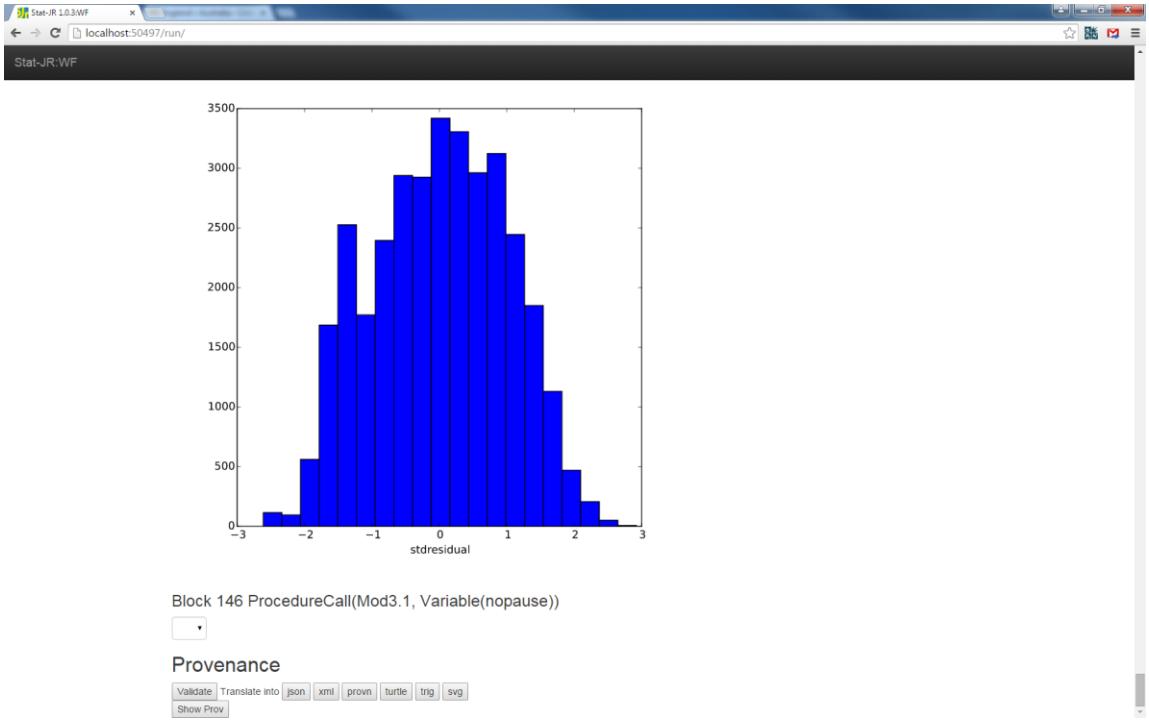

Figure 11

In this first section, whilst a number of the blocks are familiar to us from the first two practicals, some of templates aren't. We make use of the *Tabulate* template that can produce quite a wide range of summary statistics in tabular form and is designed to mimic the MLwiN tabulate window. We also use a *RecodeValues* template for recoding the values of a categorical variable, again mimicking the MLwiN window for recoding values.

Scrolling down the workflow reveals some other new templates we have used:

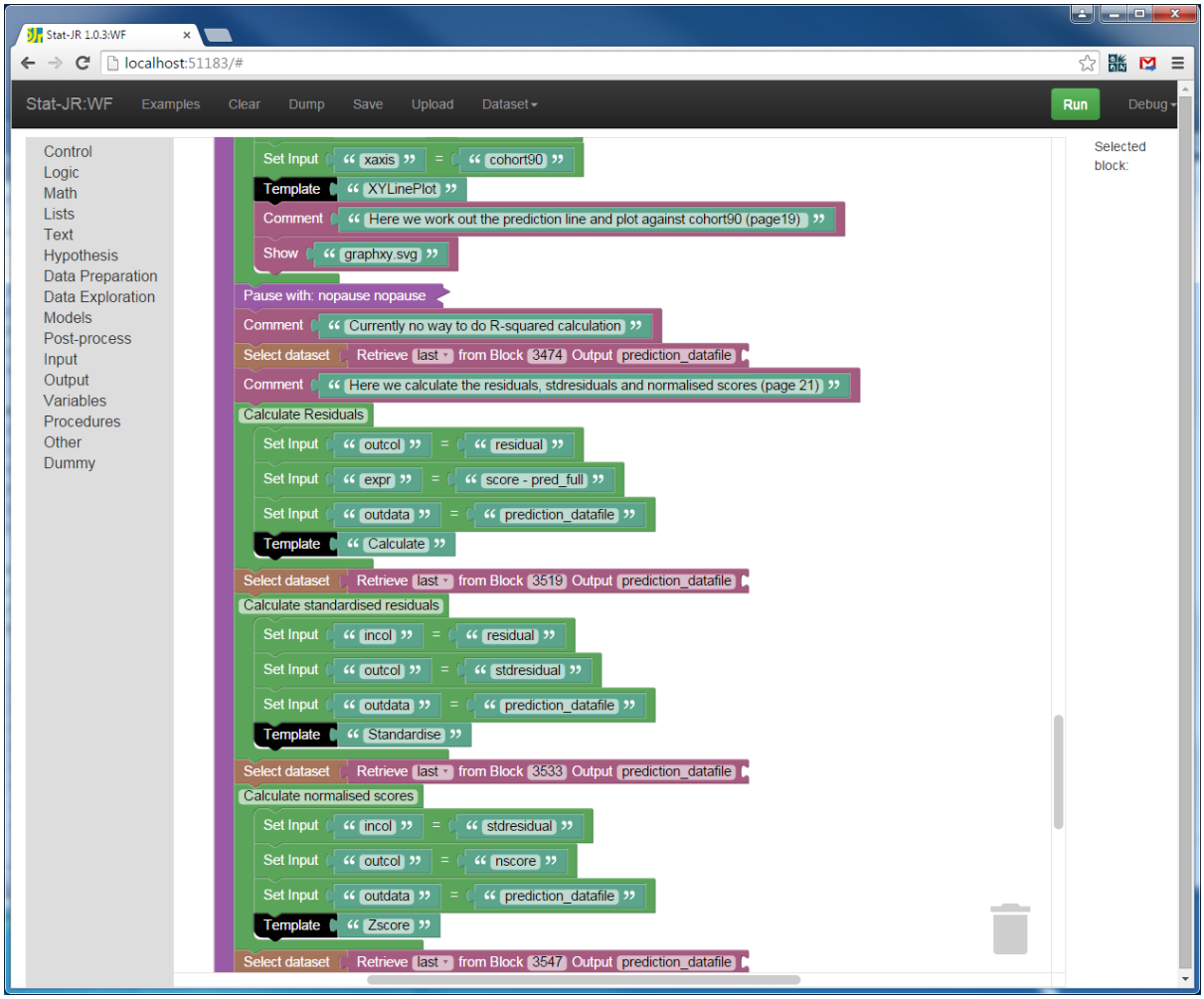

Figure 12

Here we use the *XYLinePlot* which is simply a variant of the *XYPlot* that plots lines rather than points. Then, having switched dataset to the *prediction_datafile* generated from the model fit, we use in quick succession: *Calculate* (which you have encountered before) to create residuals from responses and fitted values; *Standardise*, to create standardised residuals from raw residuals and their standard errors; and *Zscore*, to create normalised scores from the (standardised) residuals. Each of these templates adds a column to a dataset, each of which we save using the same name, ensuring we are using the correct (latest) version by appending the *Retrieve* block onto a *Select dataset* block.

As an exercise you might take your own dataset and see if you can, by choosing one response and one predictor, replicate this exploratory analysis on your dataset: which aspects of the workflow would you need to modify to accommodate your own dataset, and how?

LEMMA P3.2: Comparing groups: regression with a single categorical explanatory variable

We can now look at the next procedure in the workflow (*Mod3.2*). To do this we will detach the *procedure call* to the *Mod3.1 procedure* from the *Start* block and instead attach the *procedure call* to *Mod3.2*. Here we've expanded the *procedure block* for *Mod3.2* and swapped it with *Mod3.1*, which we've collapsed once more:

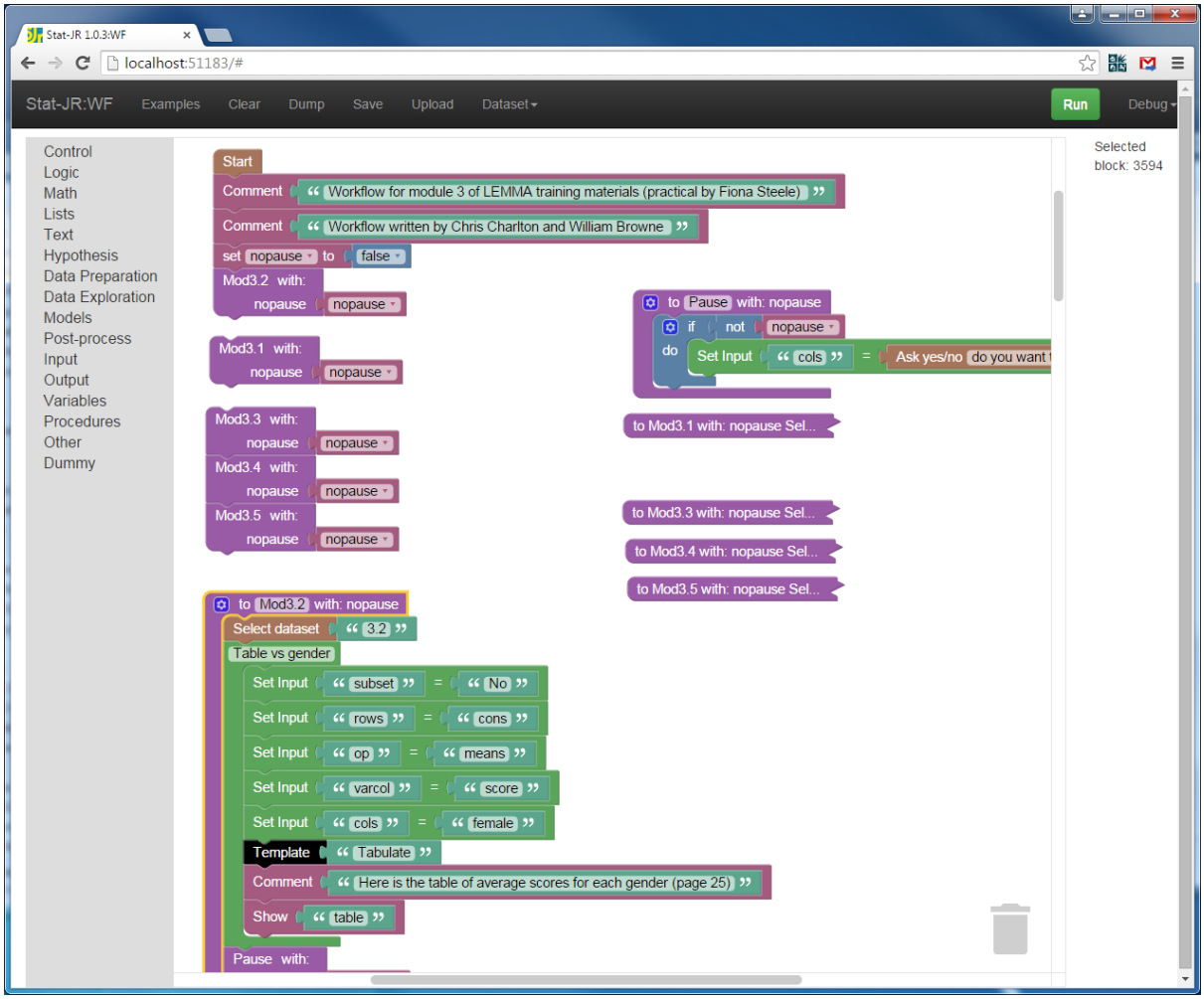

Figure 13

Section 3.2 of the LEMMA MLwiN practical covers some basic modelling of categorical predictor variables. The workflow here consists of some tabulations of the response variable (*score*) for different categorical variables using the *Tabulate* template, several calls to the *Regression1* template for model fitting and much use of the *Calculate* template to create dummy variables for the different categories of social class and the different cohorts, primarily because the models fitted have differing base categories.

There are no really new templates or blocks here so we suggest you simply **Run** the section and cross-reference it with the relevant LEMMA materials. If you have your own data and it contains categorical predictors you might like to adapt the code to your dataset.

LEMMA P3.3: Regression with more than one explanatory variable (multiple regression)

We can now look at the next procedure in the workflow (*Mod3.3*). As before we will detach the call to the previous procedure we looked at (*Mod3.2*) from the blocks contiguous with the *Start* block, and attach the call to the procedure we're interested in now (*Mod3.3*). Again, we will expand / collapse blocks and swap things around so we can better see the blocks of current interest:

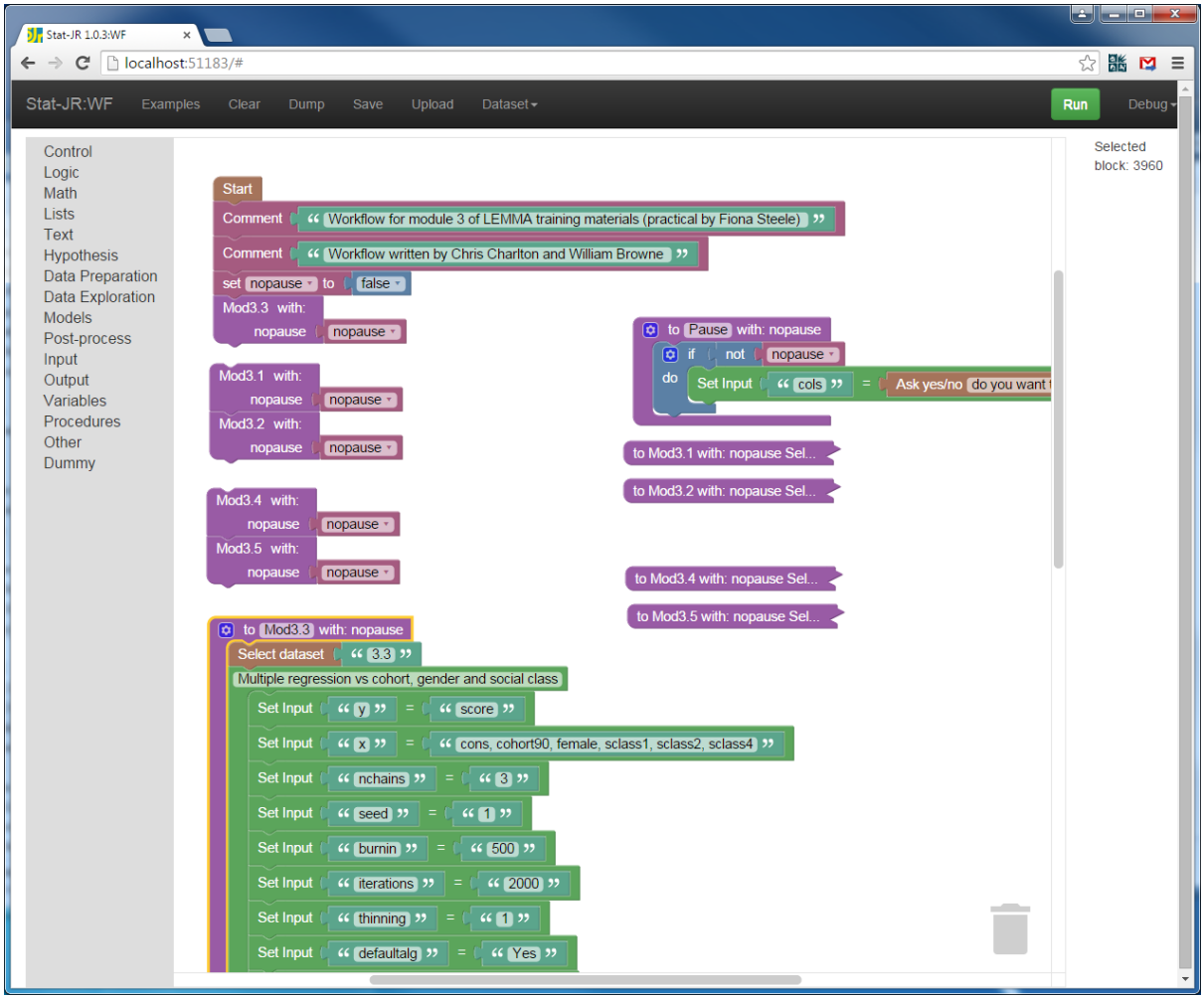

Figure 14

Procedure *Mod3.3* is actually quite short as this section of the LEMMA 3 MLwiN practical simply introduces the concept of multiple regression by placing the three predictor variables, investigated separately up to that point, into the same model. It does this using the *Regression1* template, with which we are familiar. It then also displays a tabulation of two categorical variables to show how social class has changed over time.

Again we suggest you **Run** the code and investigate whether it replicates the LEMMA materials, and if it relates to your own dataset try modifying it accordingly before moving onto the next section.

LEMMA P3.4: Interaction effects

This section is quite a long one in the training materials. We've done our usual moving / expanding / collapsing of the blocks below:

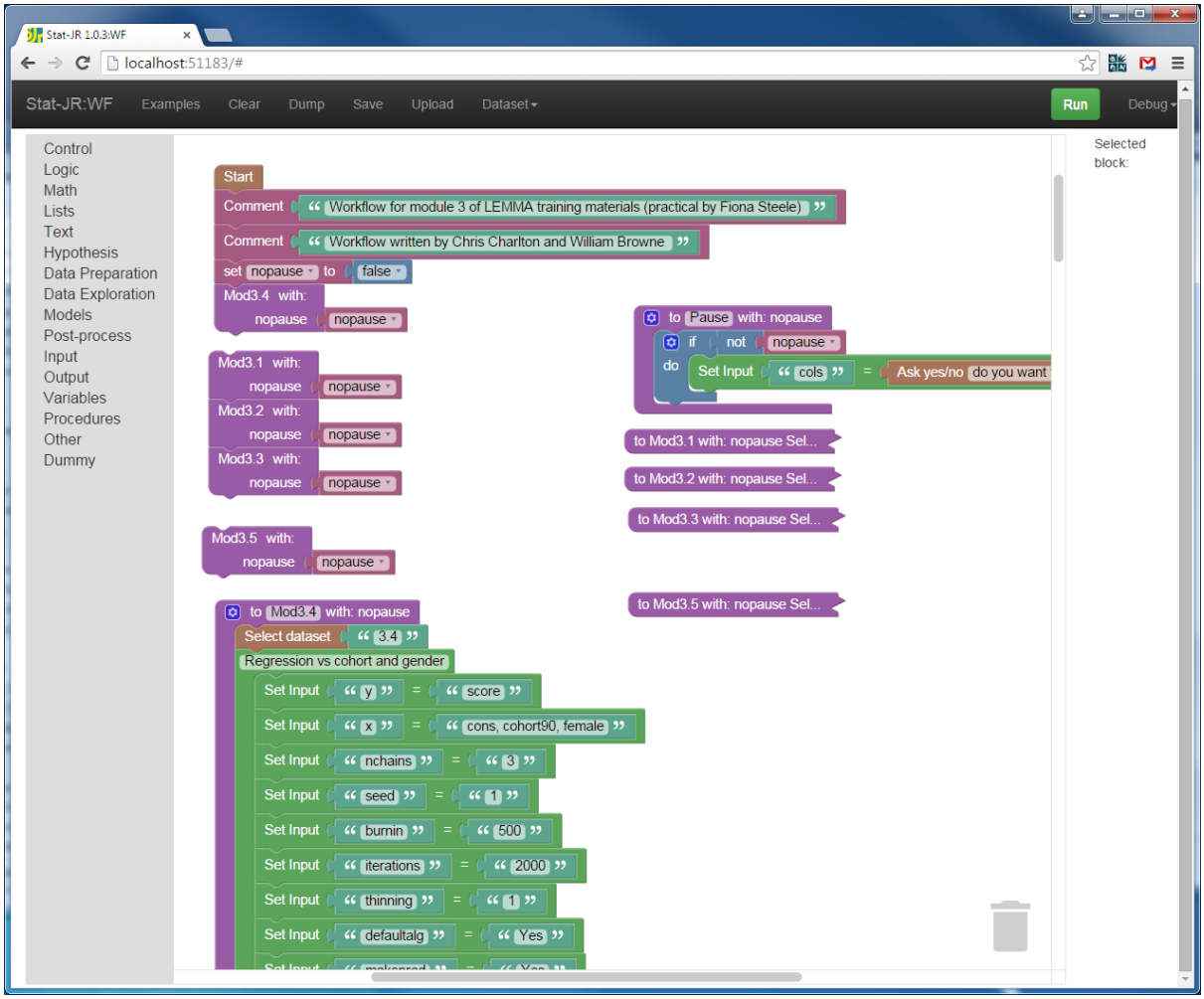

Figure 15

In this section of the LEMMA materials the concept of interactions is introduced, and several regressions are fitted. To start with a multiple regression of cohort and gender on the hedonism score is fitted. The resulting model fit is plotted using a template we haven't come across previously, *XYGroupPlotLine,* which plots separate lines for each group. Another new template, *Choose*, is then used to select subsets of the data, firstly all boys and secondly all girls, and separate regressions of the hedonism score on cohort are performed for each subset. To illustrate interactions, the *Calculate* template is used to create the interaction term and a model including it is fitted. The *XYGroupPlotLine* template is used again, plotting separate lines that are not parallel for the two genders.

Attention then moves from gender to social class, which has more categories. The *Calculate* template is used to create interactions before the *Regression1* template is used to fit a model including these interactions. The fit of the model is illustrated in two ways using the *XYGroupPlotLine* template. Firstly a straightforward predicted line plot with a line for each social class and then a plot of the differences for each social class from a base category. Here the workflow illustrates how to extract values from a table of results thus:

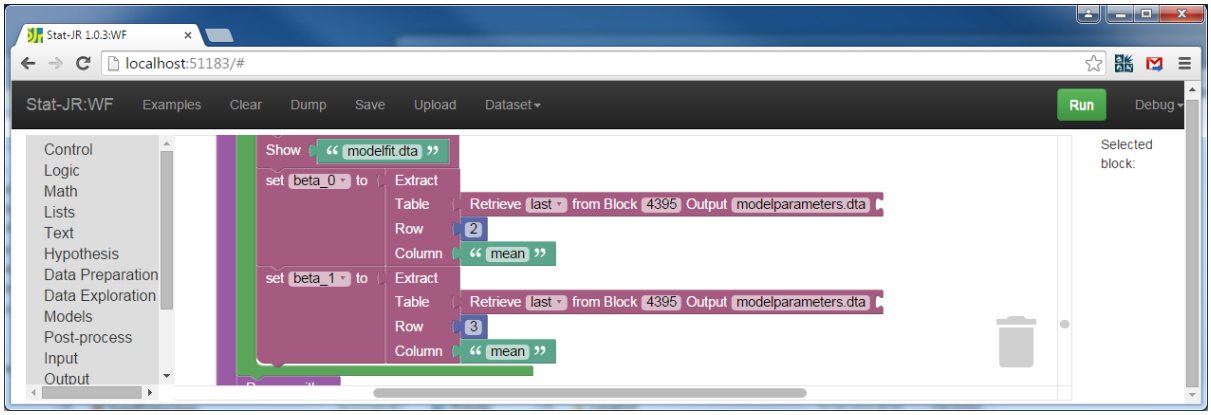

Figure 16

This takes the values of the means of *beta_0* and *beta_1* from the model fit, which are then used to create the differences (stored as *predscore*) from the base category (social class 3) as shown below:

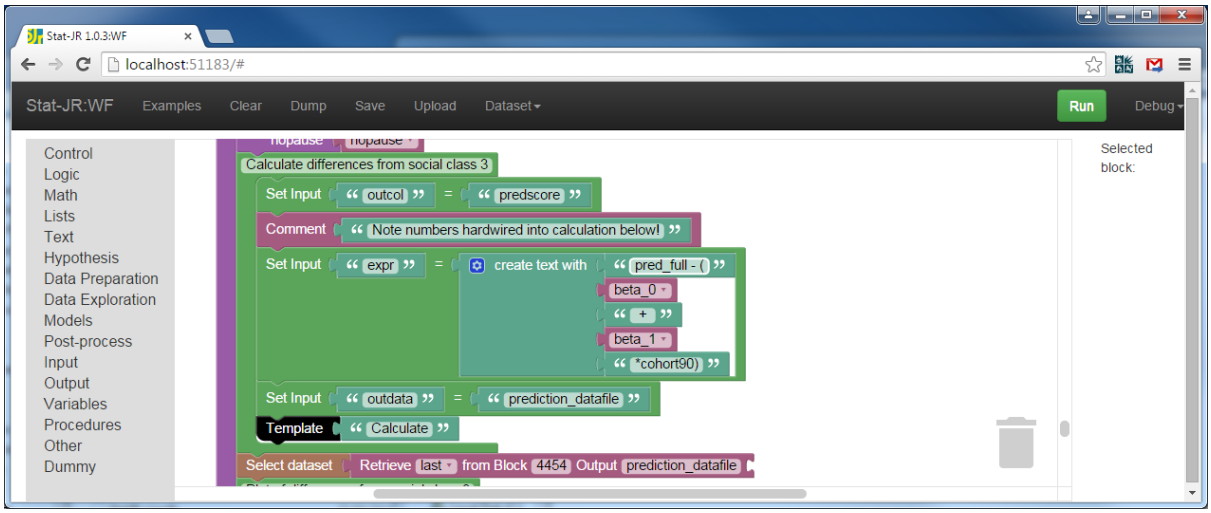

Figure 17

Finally the plot is constructed before a final model without interactions is constructed for comparison purposes.

Again we suggest you try running the section to satisfy yourself you understand the code and see how it replicates the section in the LEMMA materials. If you have suitable data you might consider modifying the code to use your own dataset.

LEMMA P3.5: Checking model assumptions in multiple regression

In this final short section of the workflow, the various predictor variables are brought together in one final model. This model isn't so easy to show graphically so instead this section focusses on checking the fit of the model. If we move the procedures around to run *Mod3.5* we can get the workflow to look as follows:

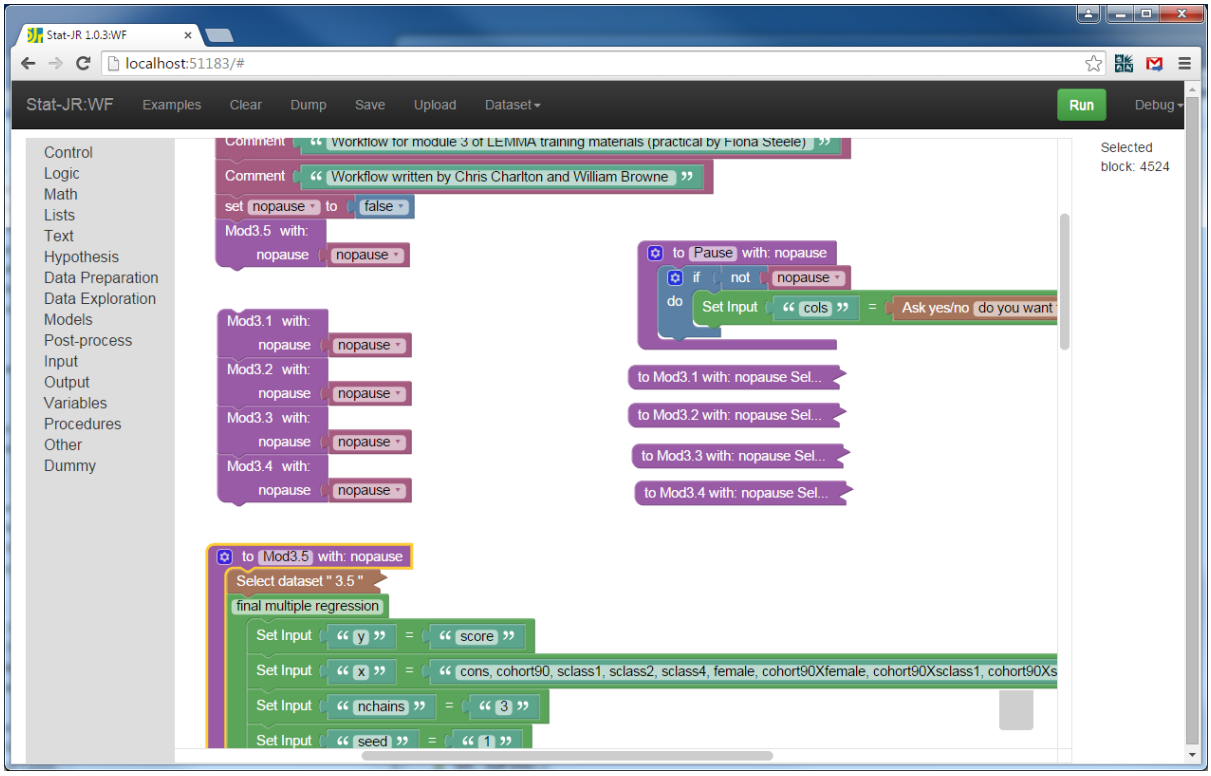

Figure 18

In practice this section very closely resembles the first section only with a more complex model, i.e. all the calculations and plots are ones we have seen before. We suggest you **Run** the section to replicate the LEMMA materials and then consider what you would do with your own dataset.

What have we covered?

In this practical we have demonstrated the use of *procedure* blocks to group sections of workflow together which can then be called from elsewhere in the workflow. We have also encountered a number of new templates, and have more generally demonstrated how we might use the tools of Stat-JR's workflow system to replicate the outputs found in the LEMMA training materials. Given Stat-JR's ability to interoperate with a wide variety of third-party statistical software packages (R, MLwiN, Stata, etc.) this workflow could eventually be modified to allow the user to toggle between packages. In doing so it could expose the scripts (R scripts, *.do* files, etc.) used to run each execution, so that the user can cross-reference the script and outputs from those packages, and gain insight into how the same operation might be achieved by a number of different packages.

At this point you should have the tools to try out other things. For example, you may like to consider fitting other models to your own dataset, perhaps even using other model-fitting templates (e.g. *1LevelMod*, *2LevelMod*) which you can always test first using TREE.