

SOUTHMOD

User manual

Uganda

UGAMOD v1.1

Dorothy Nakyambadde, Ronald Waiswa, Tina Kaidu, Joseph Okello,
Nicholas Musoke and Suzan Kavuma

March 2019



Contents

Contents.....	2
ACKNOWLEDGEMENTS.....	3
1. INTRODUCTION.....	4
1.1 ABOUT UGAMOD	4
1.2 INTRODUCTION TO THIS MANUAL	5
2. GETTING STARTED WITH UGAMOD	7
2.1 A BRIEF DESCRIPTION OF UGAMOD	7
2.2 THE UGAMOD USER INTERFACE	8
3. UGAMOD IN DETAIL.....	10
3.1 INTRODUCTION TO POLICIES	10
3.2 DEFINITIONAL POLICIES	14
3.3 TAX-BENEFIT POLICIES	22
3.4 GENERAL SETTINGS.....	27
3.5 VARIABLES.....	31
4. TASKS IN UGAMOD	34
4.1 RUNNING UGAMOD.....	34
4.2 ADDING A NEW SYSTEM IN UGAMOD.....	36
4.3 IMPLEMENTING A POLICY REFORM IN UGAMOD.....	37
4.4 ADDING NEW VARIABLES TO UGAMOD	43
4.5 USING THE STATISTICS PRESENTER.....	44
4.6 OTHER TASKS	52
References	53

ACKNOWLEDGEMENTS

United Nations University World Institute for Development Economics Research (UNU-WIDER) is thanked for funding the building of the model and the preparation of this manual. The manual was produced as part of SOUTHMOD, a major research project in which tax-benefit microsimulation models for selected developing countries in Africa (Ethiopia, Ghana, Mozambique, Uganda, Zambia) and also elsewhere (Ecuador and Viet Nam) are built in addition to those that already exist for South Africa and Namibia. SOUTHMOD is a collaboration between UNU-WIDER, the EUROMOD team at the Institute for Social and Economic Research (ISER) at the University of Essex, and Southern African Social Policy Research Insights (SASPRI).

1. INTRODUCTION

1.1 ABOUT UGAMOD

Tax-benefit microsimulation models, which combine representative household-level data on incomes and expenditures and detailed coding of tax and benefit legislation, have proven to be an extremely useful tool for policymakers and researchers alike. The models apply user-defined tax and benefit policy rules to micro-data on individuals and households and calculate the effects of these rules on household income. The effects of different policy scenarios on poverty, inequality, and government revenues can be analysed and compared.

Uganda, like other developing countries, is now building up its social protection system and the financing of public spending will need to be increasingly based on domestic tax revenues. In this process, understanding the system-wide impacts of different policy choices is critically important, and tax-benefit microsimulation models are very well suited for this purpose.

Against this backdrop UNU-WIDER, the EUROMOD team at the Institute for Social and Economic Research (ISER) at the University of Essex, and Southern African Social Policy Research Insights (SASPRI) have launched SOUTHMOD, a major research project in which tax-benefit microsimulation models for selected developing countries in Africa (Ethiopia, Ghana, Mozambique, Tanzania, Uganda, and Zambia) and also elsewhere (Ecuador and Viet Nam) are built in addition to those that already exist for South Africa and Namibia.

UGAMOD, the tax-benefit microsimulation model for Uganda, has been developed in cooperation with the Uganda Revenue Authority and Makerere University Kampala. UGAMOD is based on the UGANDA NATIONAL HOUSEHOLD SURVEY 2016/2017 allowing for representative results on the national and sub-national level. Policies are simulated for 2016, 2017 and 2018.

UGAMOD is a highly versatile yet easy to use tool for policymakers and researchers alike. Possible policy reform simulations in UGAMOD include, for example, a universal child benefit, a universal pension payment to the elderly or a youth unemployment benefit. With UGAMOD the number of beneficiaries and the total cost to the state budget can be simulated and characteristics of the prospective recipients analysed (such as: Are prospective recipients more likely to live in urban or rural areas? Do they work in the formal sector or not? etc.). Yet UGAMOD also allows to calculate the effects on the government's budget (How much would such a policy cost?) including, for example, how tax rates could be increased to offset the additional expenditures on social protection.

Microsimulation is a technique that involves taking household survey data and applying a set of policy rules to the data to calculate individual entitlement to benefits and/or liability for taxation. The resulting output at individual and household level can then be analysed to provide national data on, for example, impact of social benefits on poverty and inequality. The base model simulates the existing tax/social benefits arrangements within the country. However, the real strength of the model is that it enables hypothetical changes to benefits and/or the tax system to be simulated and the impact of such changes on the income distribution (including changes in poverty and inequality) to be assessed. Moreover, the expenditure on both the status quo and on any revisions social

benefits can be assessed as can the revenue generated through personal taxation (both direct and indirect).¹

The EUROMOD platform on which UGAMOD is based was built by Professor Holly Sutherland and colleagues at the University of Essex to simulate policies for the European Union countries.² EUROMOD has been built and developed over a 21 year period and now runs simulations for over 25 countries. The main benefits of EUROMOD which make it a particularly suitable basis for the Ugandan model are that: all the calculations are transparent and can be easily modified by the user, and the model is very flexible as it allows policies to be modified and almost any type of new policy to be created.

EUROMOD has a stand-alone user-friendly interface which is stable and compatible with computers running on Windows operating systems. It provides greater control and guidance over user actions, and offers increased functionality and improved user-friendliness.

This manual is prepared with reference to UGAMOD version 1.1. The manual will be equally applicable to future iterations of version 1 (e.g. 1.2, 1.3 etc.) of UGAMOD.

1.2 INTRODUCTION TO THIS MANUAL

This manual is designed as an introductory guide for new users and a reference for those already familiar with the basic operations of UGAMOD. The manual provides comprehensive instructions on using the model for the first time as well as more complex tasks such as building new policies.³ The focus of the manual is on the technicalities of how to use the model UGAMOD, in practice, rather than a manual about the many processes that could be undertaken using the EUROMOD software more generally. A separate Country Report has been produced which describes each of the taxes and benefits that are included in UGAMOD, as well as a discussion of how the simulated results compare with external sources of data for validation purposes (Byaruhanga et al 2019). A Data Requirement Document has also been produced which describes each of the input variables that are used within UGAMOD.

This manual is organized into four main interlinked sections. Section 1 provides an introduction and background to UGAMOD and the manual while section 2 introduces users or readers to the model and its interface. Section 3 presents UGAMOD in detail by describing how to interpret and use the content in the main content file which stores all the information the model needs for its policy simulations. Section 4 explains how to undertake a number of tasks in UGAMOD such as running the

¹ See, for example Mitton et al., 2000 and Zaidi et al., 2009.

² Sutherland and Figari, 2013.

³ Much of the material in this manual has been drawn from documentation prepared for the EUROMOD model and the authors are grateful to the EUROMOD team for granting their permission to use this material.

model, adding in new **SYSTEMS** (the rules necessary to simulate a particular tax-benefit system, e.g. rules for 2016 and 2017 or rules for a reform scenario), and implementing policy reforms.

Throughout the manual special EUROMOD terms are printed in blue capitals, e.g. **INCOMELIST**, **TAXUNIT**, **PARAMETER**. Filenames, tab names, policy names, names of menu items, etc. are printed in italics, e.g. (file) *ug.xml*, (tab) *Display*, (policy) *uprate.ug*, (menu item) *Save Country*.

The following boxes are also used:



Boxes on technical details are marked with a gearwheel. These boxes provide additional information to the main text which is not crucial to understanding the main operations of the model.



Boxes with a warning sign contain information that will help you avoid some of the more common mistakes that can be made when running the model. All users should pay special attention to text in these boxes.



Boxes with a notepad provide a summary of key points and can be used as a quick reminder or reference.

More detail on all aspects of EUROMOD can be found in the EUROMOD help which can be accessed from the *Help & Info* tab within UGAMOD. For general information about the EUROMOD microsimulation model and related research please refer to <https://www.iser.essex.ac.uk/euromod>.

2. GETTING STARTED WITH UGAMOD

2.1 A BRIEF DESCRIPTION OF UGAMOD

UGAMOD consists of a software file and several content files. The software file includes the user interface, the executable and the integrated help menu. The content files include the country xml files, as well as various other tools and applications. The software and content files can be updated separately from each other allowing greater flexibility.

The information the model needs for its calculations is stored in the main content files (*ug.xml* and *ug_DataConfig.xml*). These files contain both the information for the implementation of the framework of the tax-benefit model and for the implementation of the particular policies that make up the tax-benefit system. However, these files are not accessed directly by the user. All user input is via the user interface. Within the user interface information is mainly written into **POLICIES**, and all **POLICIES** – whether relating to the framework of the model or to the tax-benefit policies being modelled – are directly embedded in the **POLICY SPINE** so that the whole system can be displayed all at once in a single workspace.

The **POLICIES** are made up of **FUNCTIONS** which are the building blocks for implementing a country's tax-benefit system. Each **POLICY** is described by one or more such **FUNCTIONS**. Each **FUNCTION** comprises a number of **PARAMETERS** which represent a particular element of the **POLICY** functionality. Usually more than one **FUNCTION** is used to calculate a tax or benefit and **FUNCTIONS** can interact with each other.

UGAMOD is run from the main window of the 'countries' tab of the user interface. UGAMOD draws on data stored in text files and returns the output to a text file.



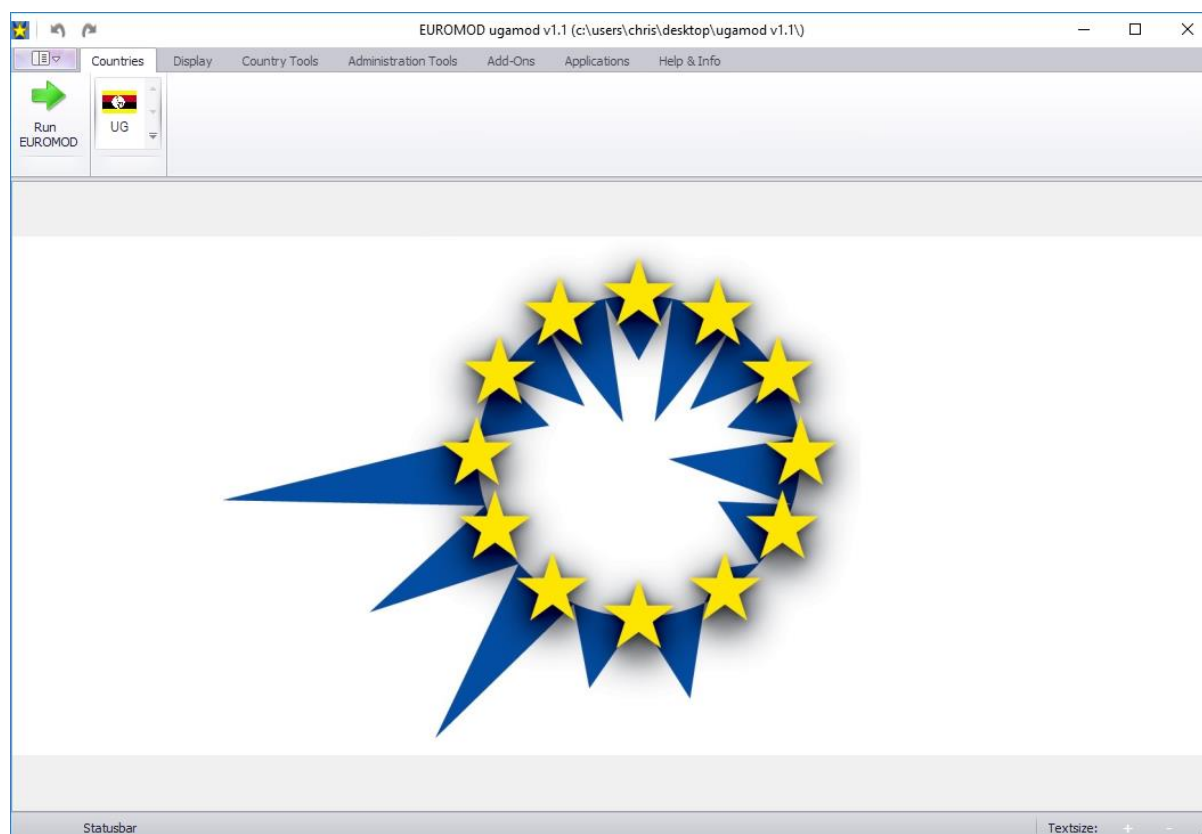
UGAMOD can be summarised as follows:

- DATA – in text format – supplied with the model (but new datasets can be added and existing ones amended)
- MODEL PROGRAM – stores all the model parameters and allows the user to make changes and run simulations
- OUTPUT – in text format – can be analysed using a statistics package

2.2 THE UGAMOD USER INTERFACE

Once the software is launched, the main window of the user interface can be accessed (see Figure 2.1).

Figure 2.1: The UGAMOD user interface

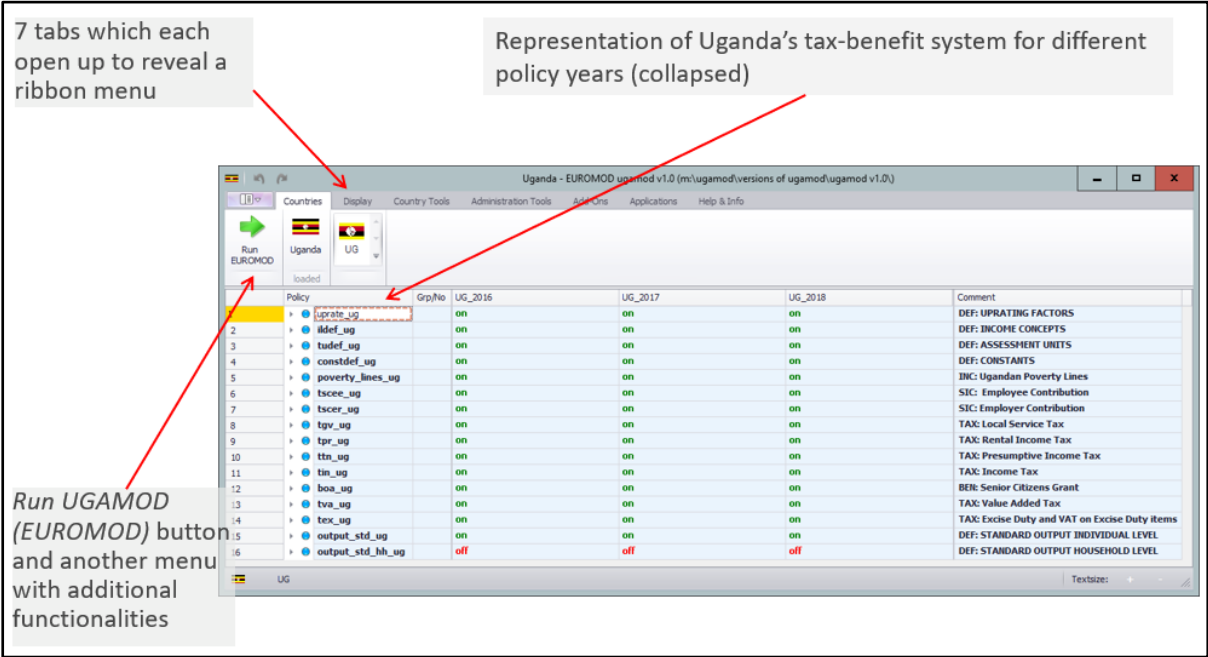


The user interface has seven tabs – *Countries*, *Display*, *Country Tools*, *Administration Tools*, *Add-ons*, *Applications*, *Help & Info* – which each open up to reveal a ribbon menu with a number of functionalities. In addition there is a *Run UGAMOD*⁴ button to the far left of the main window. The main menu directly above the *Run UGAMOD* button contains additional functionalities. Many of these components are described in subsequent sections of this manual.

In order to access the model, click on the Ugandan flag and this opens the main UGAMOD workspace. In EUROMOD there would be a number of country flags visible but in UGAMOD there is just the flag for Uganda. The main part of the window displays the representation of Uganda's tax-benefit system, which when opened is in a 'collapsed' format. In EUROMOD terminology this is frequently referred to as the **POLICY SPINE**, or simply **SPINE**. The UGAMOD user interface showing **POLICIES** in collapsed form and the **SPINE** is shown in Figure 2.2.

⁴ Currently this button is labelled *Run EUROMOD*.

Figure 2.2: The UGAMOD user interface



3. UGAMOD IN DETAIL

3.1 INTRODUCTION TO POLICIES

The first step in microsimulation is to collect data on the incomes and expenditures of individuals in a representative survey of households. The second step is to have a series of policy rules which can be applied to the individuals in the data to determine what benefits they are entitled to and what taxes they should pay. In the model there can be (a) one or more dataset(s) (i.e. survey data collected in different years) and (b) one or more **SYSTEM(S)** (i.e. tax-benefit rules for different policy years).

Ideally, to calculate taxes and benefits for the year 2018, for example, you would use the 2018 policy rules together with data referring to the year 2018 (or the 2017 policy rules together with data referring to the year 2017 etc.). However, corresponding data is not always available and even if so, preparing and integrating new data in the model is a very laborious task. Therefore datasets are used for simulating several policy years, by uprating monetary values to the corresponding policy year. For each **SYSTEM** there is a dataset that is most suitable, normally the one whose collection year is nearest to the policy year (the ‘best match’).



It is EUROMOD good practice to set the best match flag only for **BASELINES**. The EUROMOD Basic Concepts section of the EUROMOD Help (accessed from the *Help & Info* tab) states that **BASELINE** is the term used for a **SYSTEM**-dataset combination which fulfils the best match criterion, and in addition, the **SYSTEM** must refer to an actual policy year and the **SYSTEM**-dataset combination must be the main or default implementation for the respective policy year.

UGAMOD v1.1 is underpinned by a micro-dataset constructed using the UGANDA NATIONAL HOUSEHOLD SURVEY 2016/2017 (which relates to a 2016 time point) and currently contains **SYSTEMS** relating to 2016, 2017, and 2018 (v 1.1). The 2016/17 dataset can be used with any of the **SYSTEMS** because the *uprate_ug* **POLICY** (see below) uprates the monetary values to 2017 and 2018 using the CPI. See Section 3.4 for further information about **SYSTEM**-dataset combinations. For clarity most screenshots in this manual will have only the system for 2018 shown.

There are 16 **POLICIES** in UGAMOD V1.x, which can be grouped into definitional⁵ and tax-benefit **POLICIES**. All 16 **POLICIES** are visible in the **POLICY SPINE** and each row of the **SPINE** represents one **POLICY**. The **POLICIES** are processed by the model in the order they appear in the **SPINE**: first the definitional **POLICIES** *uprate_ug*, *ildef_ug*, *tundef_ug* and *constdef_ug*. This is followed by a **POLICY** which sets the poverty lines *poverty_lines_ug*, then the social insurance contributions **POLICIES** *tscee_ug*, *tscer_ug*. The tax-benefit **POLICIES** *tg_v_ug*, *tpr_ug*, *ttn_ug*, *tin_ug*, *boa_ug*, *tva_ug* and

⁵Referred to as ‘special policies’ in the EUROMOD Basic Concepts section of the EUROMOD Help (accessed from the *Help & Info* tab).

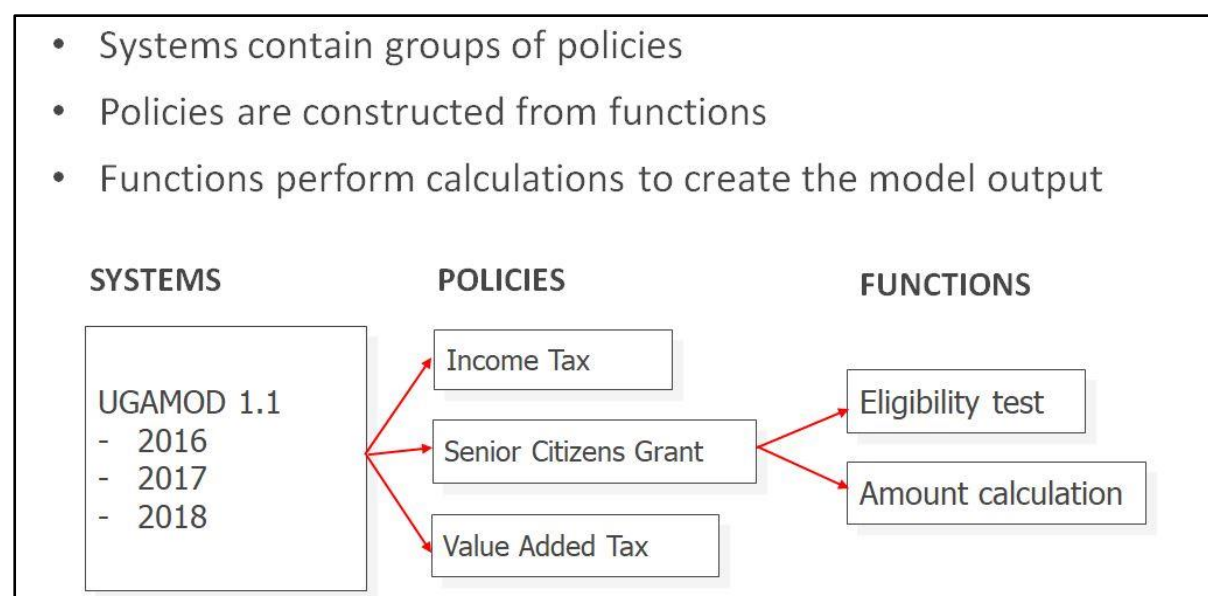
tex_ug come next and these are then followed by the definitional **POLICIES** *output_std_ug* and *output_std_hh_ug*. In other words, first, monetary data variables are uprated to the year to which the tax-benefit system refers; second, expenditure variables for VAT and excise duties are brought into the model. Third, definitions of income concepts and assessment units are specified; fourth, **CONSTANTS** are defined⁶; fifth, there is a policy which specifies the poverty lines appropriate for the year in question; sixth social insurance contributions are modelled; seventh, the taxes and benefits (direct taxes, indirect taxes and social benefits) are computed; and finally, the results are outputted. The order in which the **POLICIES** are simulated (and thus defined in the **POLICY SPINE**) is crucial because some **POLICIES** draw on variables produced in other **POLICIES** and so these need to be created first.

Under the *Display* tab you can choose whether to view the *Full Spine* or a *Single Policy* by checking the appropriate box.

All **POLICIES**, whether definitional or tax-benefit, have the same general structure.

The following figure (Figure 3.1) shows this general structure:

Figure 3.1: The relationship between **SYSTEMS**, **POLICIES** and **FUNCTIONS**



The structure is now described briefly using the Senior Citizens Grant as an example. The **POLICY** *boa_ug* models this grant.

Figure 3.2: The **POLICY** *boa_ug*

⁶**CONSTANTS** are explained in the next section.

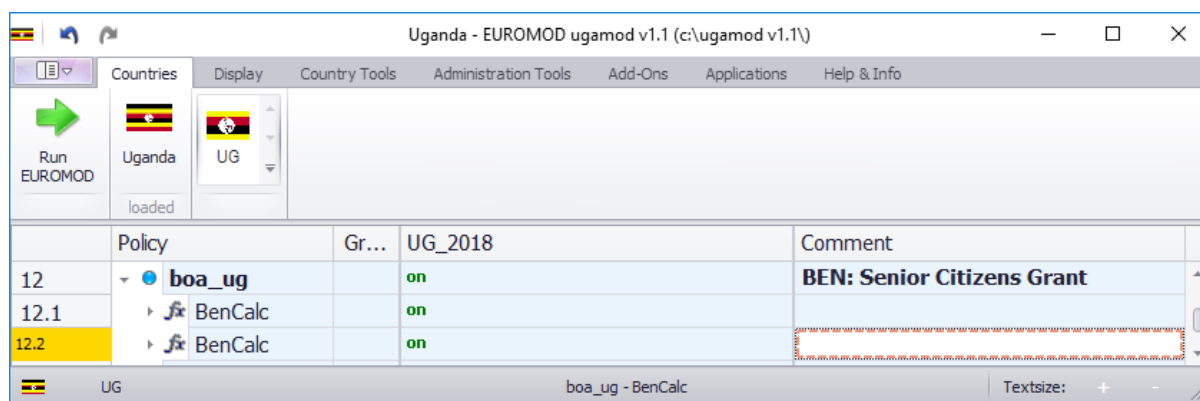


Figure 3.2 shows the **POLICY** for the Senior Citizens Grant (*boa_ug*). Each **POLICY** consists of a header displaying the name of the **POLICY** (*boa_ug* in the example) and a ‘switch’ defining whether the **POLICY** is activated or not in each **SYSTEM** (only **SYSTEM** *UG_2018* is shown in Figure 3.2⁷). This facility to switch off a **POLICY** may, for example, be used if a reform scenario is implemented where the **POLICY** is not required.

The **POLICY** is composed of **FUNCTIONS**. By right-clicking on the blue dot next to the **POLICY** name and selecting *Expand All Functions* you can view all the **FUNCTIONS** of the **POLICY**. You can also do this in a stepwise way by using the grey arrow heads next to the **POLICY** name and the **FUNCTIONS** within. It is also possible to expand all **POLICIES** in the model at once by right-clicking on the word *Policy* and selecting *Expand All Policies*.

Each **FUNCTION** is a self-contained building block that has its own **PARAMETERS** and represents a particular component of the **POLICY** functionality. The purpose of using **FUNCTIONS** as building blocks of the model is to provide a general structure and standardised language to describe policy instruments.

FUNCTIONS can be classified in three categories: policy (for implementing tax-benefit policies), system (for implementing the framework of the model) and special. Nine **FUNCTIONS** are used in UGAMOD, four are policy **FUNCTIONS** (*ArithOp*, *BenCalc*, *SchedCalc*) and six are system **FUNCTIONS** (*Uprate*, *DefVar*, *DefIL*, *DefTU*, *DefConst*, and *DefOutput*). There are no special **FUNCTIONS** in UGAMOD. A summary of all the **FUNCTIONS** and their **PARAMETERS** available for use can be found in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab).

The Senior Citizens Grant example is a fairly typical benefit **POLICY** that can be implemented. In the *boa_ug* **POLICY** there are two **FUNCTIONS** – both *BenCalc* – each of which determines eligibility for the benefit and assigns the relevant amount to each eligible person. The *BenCalc* **FUNCTION** is one of the most frequently used **FUNCTIONS** in UGAMOD. More complex **POLICIES** may have a number of **FUNCTIONS**. For example, currently the most complex benefit modelled in UGAMOD is the Presumptive Tax which is modelled in the **POLICY** *ttn_ug* has six **FUNCTIONS**: a *DefVar* **FUNCTION** which defines intermediate variables used within the **POLICY**, one *Elig* **FUNCTION** and seven *BenCalc* **FUNCTIONS**.

⁷ The other **SYSTEMS** *UG_2016*, and *UG_2017* are hidden

Each **FUNCTION** consists of a header displaying the name of the **FUNCTION** (*BenCalc* in the Senior Citizen's grant example) and a 'switch' defining whether the **FUNCTION** is activated or not (the **FUNCTION** is switched on in the example). This facility to switch off a **FUNCTION** may, for example, be used if a reform scenario is implemented where, in a policy with more than one **FUNCTION**, one of the **FUNCTIONS** is not required.

UGAMOD is told how to calculate a particular **POLICY** by setting the **PARAMETERS** of a **FUNCTION** to appropriate values. Many **PARAMETERS** appear within multiple **FUNCTIONS**, while some are specific to particular **FUNCTIONS**. Most of the policy **FUNCTIONS** and some of the system and special **FUNCTIONS** provide **COMMON PARAMETERS**. There are compulsory and optional **PARAMETERS**, for example the **PARAMETER** *TAX_UNIT* must be included in all policy **FUNCTIONS** otherwise UGAMOD will issue an error message. **PARAMETER VALUES** can take a number of different forms, for example yes/no values, amounts, variables and formulae. These will be introduced as necessary in subsequent sections. The **PARAMETER VALUES** may change for each **SYSTEM**.

Figure 3.3 shows *boa_ug* with the first *BenCalc* **FUNCTION** expanded. In the final column, *Comment*, where each **POLICY** is named (the name starts with either *DEF* for definitional, *INC* for income⁸, *TAX* for tax, *BEN* for benefit, or *SIC* for social insurance contributions), each **FUNCTION** can also be described, and each of the **PARAMETERS** can be explained if necessary.

Figure 3.3 The **POLICY** *boa_ug* with the first *BenCalc* **FUNCTION** expanded

	Policy	Gr...	UG_2018	Comment
12	boa_ug		on	BEN: Senior Citizens Grant
12.1	BenCalc		on	
12.1.1	Comp_C...	1	{drgn2=308} & {dag>=60} {drgn2=311} & {dag>=60}	Eligible if in Moroto & Nakapiripit districts which are part of Karamoja region
12.1.2	Comp_p...	1	\$senior_grant_amount	
12.1.3	Output_Var		boa_s	
12.1.4	TAX_UNIT		tu_individual_ug	
12.2	BenCalc		on	

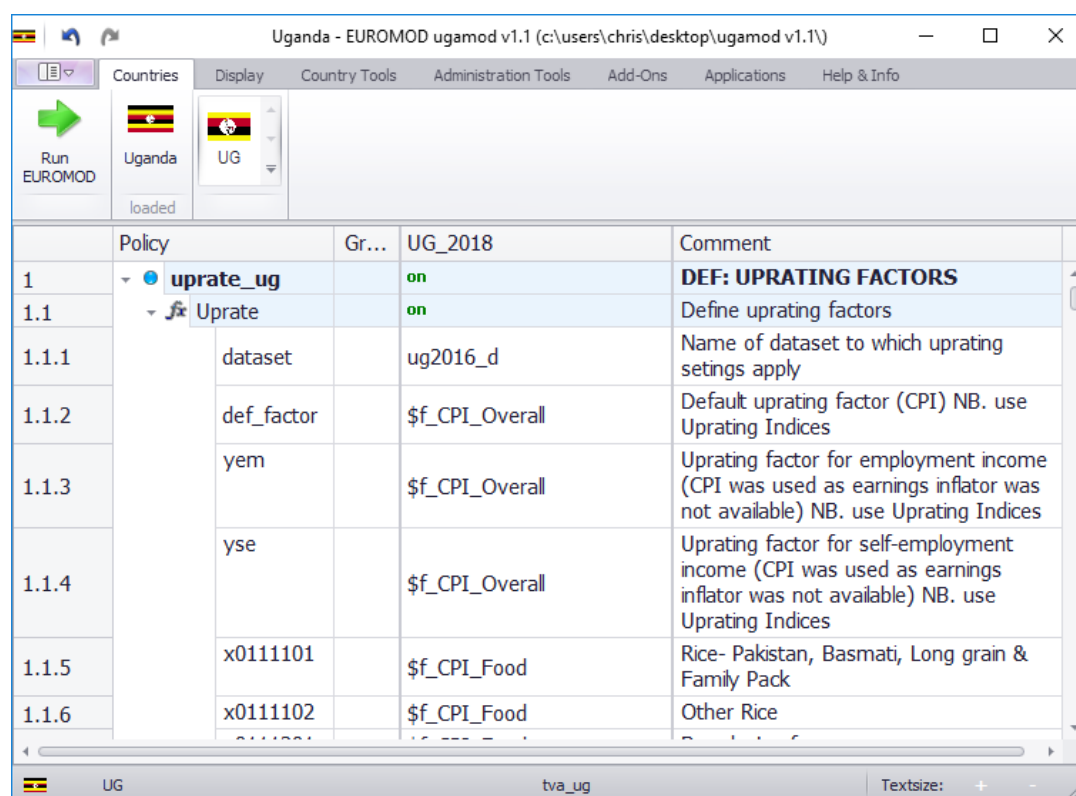
⁸ The only *INC* policy in UGAMOD is the policy which sets the poverty lines for the Statistics Presenter

3.2 DEFINITIONAL POLICIES

Uprate_ug

Overview: Datasets are usually used for simulating several policy years, by uprating monetary values to the corresponding year. The **POLICY** *uprate_ug* contains information for uprating monetary variables in the dataset.

Figure 3.4: The **POLICY** *uprate_ug*



	Policy	Gr...	UG_2018	Comment
1	• uprate_ug		on	DEF: UPDATING FACTORS
1.1	• Uprate		on	Define uprating factors
1.1.1	dataset		ug2016_d	Name of dataset to which uprating settings apply
1.1.2	def_factor		\$f_CPI_Overall	Default uprating factor (CPI) NB. use Uprating Indices
1.1.3	yem		\$f_CPI_Overall	Uprating factor for employment income (CPI was used as earnings inflator was not available) NB. use Uprating Indices
1.1.4	yse		\$f_CPI_Overall	Uprating factor for self-employment income (CPI was used as earnings inflator was not available) NB. use Uprating Indices
1.1.5	x0111101		\$f_CPI_Food	Rice- Pakistan, Basmati, Long grain & Family Pack
1.1.6	x0111102		\$f_CPI_Food	Other Rice

Figure 3.4 shows the **POLICY** *uprate_ug*. The **FUNCTION** *Uprate* allows for the uprating of monetary dataset variables to the price level of a policy year.

The **PARAMETER** *dataset* specifies the name of a dataset for which the uprating settings apply.

The default uprating factor is specified in *def_factor*. This is the CPI (UBOS, 2015). It is expressed as *\$f_CPI_Overall*. The numerical values of the uprating factors are specified in the 'uprating indices' dialogue which is called from the tab 'Country Tools' and pressing the 'uprating indices' button. This is shown in figure 3.5.

Figure 3.5 Updating Indices

Index	Reference	2015	2016	2017	2018	Comment
1 Overall CPI (Base October 2009=100)	\$f_CPI_Overall	150.3	158.1	167.1	172.2	
2 Food CPI (Base Date=1000)	\$f_CPI_Food	159.2	166.8	184.9	180	Source: UBS; Food
3 Non Food CPI (Base Date=100)	\$f_CPI_Non_Food	146.9	154.7	160.2	169.6	Source: UBS; Non food
4 Alcohol CPI (Base Date=100=100)	\$f_CPI_Alcohol	144.9	153.1	153.9	158.5	Source: UBS; Alcohol and Tobacco
5 Tobacco CPI (Base Base Date=100=100)	\$f_CPI_Tobacco	144.9	153.1	153.9	158.5	Source: UBS; Alcohol and Tobacco
6 Fuel (Base Base Date=100=100)	\$f_CPI_Fuel	162.8	167.6	180.7	209.6	Source: UBS; Energy and Fuels - combining electricity and other fuels for use at home with petrol and diesel
* 7						

This dialogue defines the uprating factors, gives a reference name and describes the source. Additional indices, for example wage inflator, can be added as additional rows. New years of data can be added by pressing the 'add year' button which will result in a new column being added to the table.

Other uprating factors can be specified for specific variables. Thus, for example, food expenditure is uprated by the food CPI \$f_CPI_Food. The first few food items are shown in Figure 3.4 (however, all food and non-food expenditure, together with expenditure on alcohol tobacco and fuel are dealt with individually within this policy).

ldef Ug

Overview: The **POLICY ldef Ug** contains definitions of **INCOMELISTS**. Technically an **INCOMELIST** is the aggregate of several variables, which are added or subtracted to build the aggregate. The most common applications of this concept are income definitions, for example disposable income or taxable income. **INCOMELISTS** are used in tax-benefit **POLICIES** for the implementation of the respective tax or benefit.

INCOMELISTS are an important concept. In a single country tax-benefit model such as UGAMOD, definitions of income may simply be hard-wired with a single variable. However, in a multi-country tax-benefit model such as EUROMOD it is important to have a standardised output format for all countries, which still allows for country specific definitions of the single variables. **INCOMELISTS** achieve this and are therefore retained within UGAMOD.

Figure 3.6: The `POLICY` *ildef_ug*

Uganda - EUROMOD ugamod v1.1 (c:\users\chris\Desktop\ugamod v1.1)

Countries	Display	Country Tools	Administration Tools	Add-Ons	Applications	Help & Info
Run EUROMOD Uganda loaded	 UG					

Policy	Gr...	UG_2018	Comment
1 ▶ ● uprate Ug		on	DEF: UPRATING FACTORS
2 ▼ ● ildef Ug		on	DEF: INCOME CONCEPTS
2.1 ▶ ✖ DefIl		on	
2.2 ▶ ✖ DefIl		on	Define income list - Taxable income (used in income tax policy)
2.3 ▶ ✖ DefIl		on	Define income list - Taxable income (used in income tax policy)
2.4 ▼ ✖ DefIl		on	Define income list - Original income NB This is fairly Standard but may need some alteration for Uganda
2.4.1 name		ils_origy	
2.4.2 yem		+	Income from employment
2.4.3 yot		+	Income other
2.4.4 ypr		+	Income from property
2.4.5 yprld		+	Income from property - land
2.4.6 yse		+	Income from self employment
2.4.7 yag		+	Income from agriculture
2.4.8 ypt		+	Income from private transfers
2.5 ▶ ✖ DefIl		on	Define income list - Simulated Taxes NB This is fairly standard but may need some alteration for Uganda
2.6 ▶ ✖ DefIl		on	Taxes ("Direct taxes" in Statistics Presenter)"
2.7 ▶ ✖ DefIl		on	Define income list - Employee social insurance contributions
2.8 ▶ ✖ DefIl		on	Define income list - Self-employed social insurance contributions (not relevant but needed for Summary Stats Plugin)
2.9 ▶ ✖ DefIl		on	Define income list - Employer social insurance contributions (not relevant but needed for Summary Stats Plugin)
2.10 ▶ ✖ DefIl		on	Define income list - Simulated benefits
2.11 ▶ ✖ DefIl		on	Define income list - Benefits
2.12 ▼ ✖ DefIl		on	Define income list - Disposable income.
2.12.1 name		ils_dispy	
2.12.2 ils_origy		+	Original income
2.12.3 ils_ben		+	Simulated Benefits plus non sim Benefits in Original Data Set
2.12.4 ils_tax		-	Taxes
2.12.5 ils_sicce		-	Employee SICs
2.12.6 ils_sicse		n/a	Self-employed SICs

UG ildef Ug - DefIl Textsize: + -

Figure 3.6 shows the **POLICY** *ildef Ug*. The **FUNCTION** *DefIL* defines each of the **INCOMELISTS**. There are a number of such **FUNCTIONS** in *ildef Ug* but only two have been expanded in Figure 3.6: the **FUNCTION** for defining the **INCOMELIST** for original income (*ils_origy*) and disposable income (*ils_dispy*). Names of **INCOMELISTS** start either with *ils_* or *il_*, mainly to distinguish them from ordinary variables. **INCOMELISTS** starting with *il_* are 'normal' **INCOMELISTS** (which are model specific), while **INCOMELISTS** starting with *ils_* are 'standard' **INCOMELISTS** which exist in all countries' models and have a particular definition.

The income list *ils_dispy* describes one of the most important EUROMOD concepts: standard disposable income. In general the following components make up disposable income in EUROMOD:

- original income (essentially employment and self-employment income; capital, property and investment income; private pensions and transfers)
- plus benefits (cash transfers, that is unemployment benefits, public pensions, family benefits, social transfers, other (country specific) cash transfers)
- minus direct taxes (income tax, turnover tax, and other (country specific) direct taxes)
- minus social insurance contributions (paid by employees and the self-employed)

This is a fairly standard definition of disposable income but it could easily be modified if a slightly different definition was required, by either amending the existing **INCOMELIST** or adding a new **INCOMELIST**.

It can be seen in Figure 3.6 that different income components, listed by their variable name (e.g. *ils_origy*, *ils_benism*, *ils_tax.*), are either added (+) or subtracted (-) to create *ils_dispy*. In UGAMOD *ils_dispy* is not used within any **POLICY**, but rather is an important concept for the analysis stage. Three **INCOMELISTS** that are used within **POLICIES** in UGAMOD are *il_taxabley*, *il_taxabley2*, and *il_exp_vat01* (standard rated items for VAT). The **INCOMELISTS** *il_taxabley* and *il_taxabley2* define taxable income and are used in the income tax policy (*tin Ug*). The **INCOMELIST** *il_exp_vat01* lists all expenditure items subject to standard rate VAT. In fact, it lists all expenditure items and puts a + sign against items subject to standard rate VAT and an 'n/a' against items that are exempt from VAT or zero rated. N.B. items that are VAT exempt or zero rated should **not** have a '-' sign alongside them as this would mean they are subtracted from the total **INCOMELIST** and this would be incorrect.

There are also a group of **INCOMELISTS** that are required for the correct functioning of the **STATISTICS PRESENTER** application. These are described in detail in the section relating to the **STATISTICS PRESENTER** (Section 4.5).



In principle any **POLICY** other than *uprate Ug* may contain **INCOMELIST** definitions. However, for reasons of transparency, most **INCOMELISTS** are defined centrally in the **POLICY** *ildef Ug*. This rule may be disregarded if a particular **INCOMELIST** is just used temporarily in one special policy. In any case, an **INCOMELIST**, once defined, is available for all subsequent **FUNCTIONS** and **POLICIES**.

Tudef_ug

Overview: The **POLICY** *tudef_ug* contains definitions of **TAXUNITS** or assessment units. Many taxes and especially benefits do not concern single individuals, but refer to larger units, most commonly households. **TAXUNITS** specify which household member belongs to the different assessment units and who is a child, etc. **TAXUNITS** are used in tax-benefit **POLICIES** for the implementation of the respective tax or benefit.

As with **INCOMELISTS**, in a single country tax-benefit model, the definition of, for example, a dependent child may simply be hard-coded. In contrast, a multi-country tax-benefit model has to deal with numerous and possibly very different assessment unit definitions and therefore must allow for flexible specification. **TAXUNITS** allow this and are therefore retained within UGAMOD.

Figure 3.7: The **POLICY** *tudef_ug*

Policy	Grp/No	UG_2018	Comment
3 tudef_ug		on	DEF: ASSESSMENT UNITS
3.1 DefTu		on	Define tax unit - Household
3.1.1 Name		tu_household_ug	
3.1.2 Type		HH	All members of the household belong to one unit
3.1.3 DepChildCond		{dag<=17} & {dag>=0}	
3.1.4 AssignDepChOfDepend...		yes	
3.1.5 AssignPartnerOfDepend...		yes	
3.2 DefTu		off	Define tax unit - Family NOT Currently used
3.3 DefTu		on	Define tax unit - Individual
3.3.1 Name		tu_individual_ug	
3.3.2 Type		IND	Each member of the household forms own unit
3.4 DefTu		on	Define tax unit - Couple
3.4.1 Name		tu_couple_ug	
3.4.2 Type		SUBGROUP	Unit members are defined by parameter Members
3.4.3 Members		Partner	
3.4.4 PartnerCond		{Default} & {IsMarried}	Default = {head:idperson=idpartner} i.e. one's own partner id is the same as the head's id

Figure 3.7 shows the **POLICY** *tudef_ug*. The **FUNCTION** *DefTu* defines the assessment units or **TAXUNITS**.

There are three assessment unit definitions in UGAMOD, and they are called *tu_individual_ug*, *tu_household_ug*, and *tu_couple_ug* (as indicated by the **PARAMETER** *Name*). The assessment unit *tu_family_ug* is defined but is switched off as no current policy in UGAMOD uses it. The **PARAMETER** *Type* has three possible values: *HH*, *IND* and *SUBGROUP*. *HH* refers to household type units, which means that all members of the household belong to the same unit. *IND* denotes individual type

units, which means that each member of the household forms its own unit. *SUBGROUP* means that the household may be split into several units of different size.

The simplest form of definition is for the **TAXUNIT** *tu_individual_ug*. This definition is used, for example, for personal income tax, where tax is calculated for each person and child/adult definitions are not relevant.

The expression *IsMarried* is used in the **PARAMETER** *PartnerCond* in the definition of the **TAXUNITs** *tu_family_ug*, and *tu_couple_ug*. Such an expression is referred to in EUROMOD terminology as a **QUERY**. **QUERIES** are used extensively in EUROMOD, effectively as shorthand for determining the answers to particular questions (e.g. *IsMarried* answers the question ‘Is the person married?’ and saves each time having to specify all the variables and codes needed to provide the answer). The full list of **QUERIES** can be found in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab). The result of a **QUERY** is either 1 (i.e. true) or 0 (i.e. false) (e.g. for the **QUERY** *IsMarried*), or some numeric (monetary or non-monetary) value (e.g. for the **QUERIES** *GetPartnerIncome* and *nDepChildrenInTu* respectively). There are no **PARAMETERS** that only take **QUERIES** as their values; rather **QUERIES** are usually used within formulae (discussed above) and conditions (e.g. the **PARAMETER** *DepChildCond* used in the *tu_household_ug* **TAXUNIT** definition).



Negation: The operator **!** before an expression indicates negation. The expression **!{IsParent}** specifies that the person should not be a parent. Had the expression been **{IsParent}** without the **!** operator, this would have specified that the person should be a parent. Operators will be discussed in more detail in Section 3.3.

The most complex form of definition is for the **TAXUNIT** *tu_family_ug*, where *Type* is set to *SUBGROUP*, meaning that the household is split into several units of different size. Which household member belongs to which unit primarily depends on the **PARAMETER** *Members*. This is set to *Partner & OwnDepChild*. The **PARAMETER** *Members* usually defines relations with respect to the head of the unit, which is the richest person in the unit, or if there are several equally rich persons, the oldest, and if there are several equally rich and equally old persons, the one with the lowest number for the variable *idperson*. Knowing this, the **PARAMETER** *Members* can be interpreted as follows: a unit consists of the head, her/his partner and their own dependent children. Although the **TAXUNIT** *tu_family_ug* is not currently used in UGAMOD, its description is retained here as it is illustrative of how a more complex assessment unit is described and is a fairly common type of **TAXUNIT** within the broader EUROMOD family of models.

The **PARAMETERS** *DepChildCond*, *AssignDepChOfDependents* and *AssignPartnerOfDependents* further specify the **TAXUNIT** (and are the same as for the **TAXUNIT** *tu_household_ug*). The **PARAMETER** *DepChildCond* determines who is treated as a dependent child: **!{IsParent}** specifies that they should not themselves be a parent and **{dag<=17}** specifies that they should be aged under 18. The **PARAMETERS** *AssignDepChOfDependents* and *AssignPartnerOfDependents* mean that dependent children or partners respectively of dependent unit members (i.e. persons who are not the head or partner of the unit) are assigned to the unit).

In order to build the family unit, first the head of the household is determined and her/his (potential) partner and their (potential) own dependent children are assigned to her/his unit. If there are any persons left in the household who have not yet been assigned, a head is determined amongst them and her/his (potential) partner and their (potential) own dependent children are assigned to her/his unit. This process is continued until all persons in the household are assigned to a unit.



In principle any **POLICY** other than *uprate_ug* may contain assessment unit definitions. However, for reasons of transparency, assessment units are usually defined centrally in the **POLICY** *tundef_ug*. This rule may be disregarded if a particular assessment unit is just used temporarily in one special **POLICY**. Anyway, an assessment unit, once defined, is available for all subsequent **FUNCTIONS** and **POLICIES**.

Constdef_ug

Overview: The **POLICY** *constdef_ug* contains definitions of **CONSTANTS**. A **CONSTANT** is simply a way of storing a number (usually a monetary amount) for use in one or more tax-benefit **POLICIES**. Because the **CONSTANTS** defined reside in one place, this **POLICY** is a particularly good place to define the social benefit amounts that are updated each year.

Figure 3.8: The **POLICY** *constdef_ug*

Policy	Grp/No	UG_2018	Comment
4		on	DEF: CONSTANTS
4.1		on	Define constants
4.1.1		2820000#y	
4.1.2		25000#m	
4.1.3		0.18	Vat rate
4.1.4		42322.54	Basic needs poverty line
4.1.5	24	38221.82	After deducting mean indirect taxes of those around pov line
4.1.6	10	100	Current Excise Duty Rate for sugar(per kg)
4.1.7	2	0.1	Current Excise Duty Rate for mineral water (ad valorem)
4.1.8	18	.13	Current Excise Duty for sodas(ad valorem)
4.1.9	21	300	
4.1.10	4	0.15	Current Excise Duty Rate for fruit juices (ad valorem)

Figure 3.8 shows the **POLICY** *constdef_ug*. The **FUNCTION** *DefConst* defines the **CONSTANTS**. Note that only the first 10 constants defined are shown in the figure.

The name of the **CONSTANT** is defined in the *Policy* column (the first character of a **CONSTANT**'s name should always be \$) and the value of the **CONSTANT** is defined in the respective **SYSTEM**

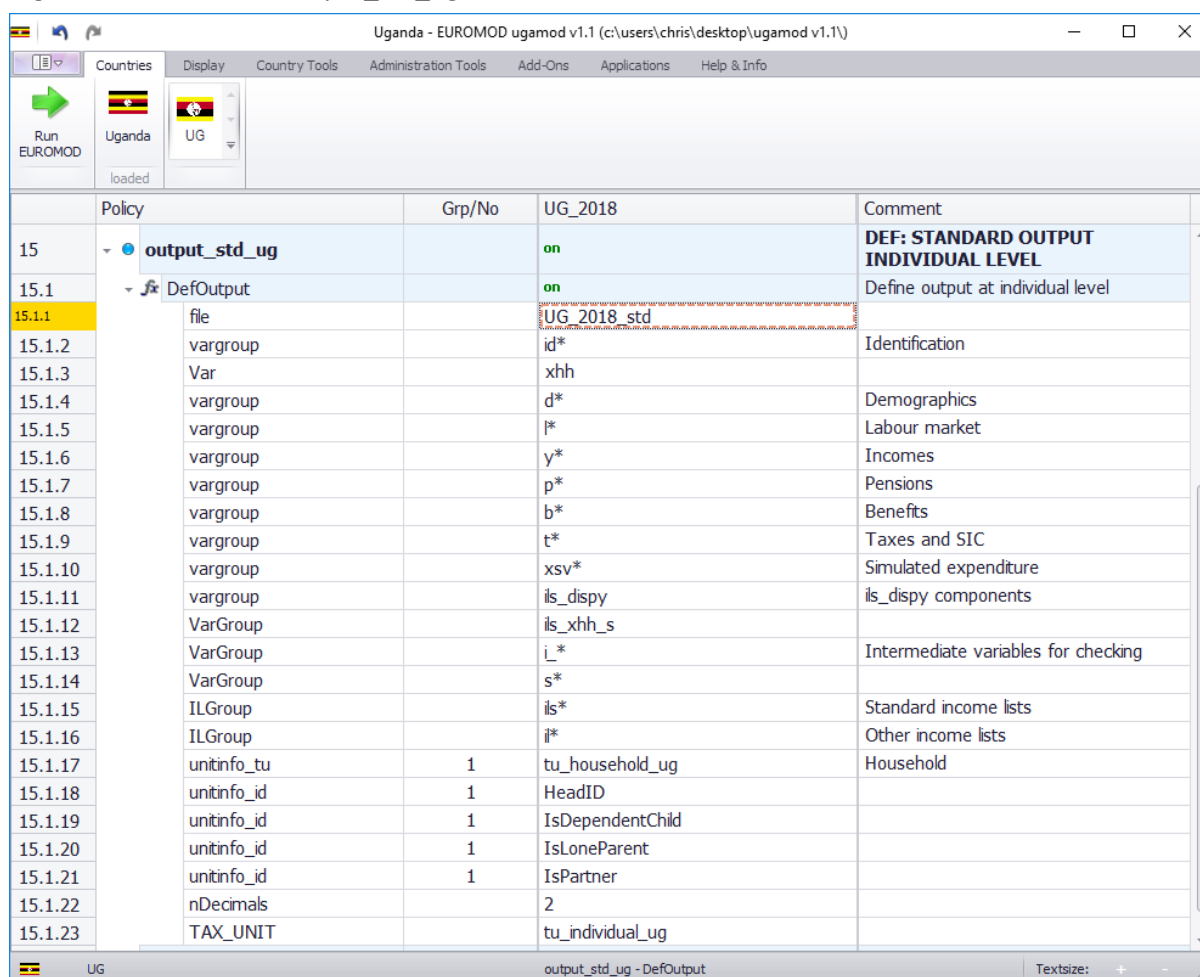
column. The *#m* after the value indicates that it is a monthly amount and the *#y* indicates that it is an annual amount. By default, a **CONSTANT** is created as a monetary variable. In UGAMOD v1.x **CONSTANTS** are defined and relate to benefit amounts, tax rates etc. For example the amount of excise duty per kilo of sugar for the excise duty **POLICY** is assigned to the **CONSTANT** *\$sugar_ed_per_kilo* and given a value of 100 UGX per kilo in 2018. Having all the benefit thresholds and amounts in one place makes the task of updating the model for a subsequent year much easier.

The **CONSTANTS** are referred to in the benefit **POLICIES**, as will be seen in Section 3.3.

Output_std_ug

Overview: The **POLICY** *output_std_ug* contains the specification of the output file.

Figure 3.9: The **POLICY** *output_std_ug*



Policy	Grp/No	UG_2018	Comment
15		on	DEF: STANDARD OUTPUT INDIVIDUAL LEVEL
15.1		on	Define output at individual level
15.1.1		UG_2018_std	
15.1.2		id*	Identification
15.1.3		xhh	
15.1.4		d*	Demographics
15.1.5		l*	Labour market
15.1.6		y*	Incomes
15.1.7		p*	Pensions
15.1.8		b*	Benefits
15.1.9		t*	Taxes and SIC
15.1.10		xsv*	Simulated expenditure
15.1.11		ils_dispy	ils_dispy components
15.1.12		ils_xhh_s	
15.1.13		i_*	Intermediate variables for checking
15.1.14		s*	
15.1.15		ils*	Standard income lists
15.1.16		il*	Other income lists
15.1.17	1	tu_household_ug	Household
15.1.18	1	HeadID	
15.1.19	1	IsDependentChild	
15.1.20	1	IsLoneParent	
15.1.21	1	IsPartner	
15.1.22		2	
15.1.23		tu_individual_ug	

Figure 3.9 shows the **POLICY** *output_std_ug*. The **FUNCTION** *DefOutput* specifies the output.

The name of the output file is determined by the **PARAMETER** *file*. **STANDARD OUTPUT** for the 2018 **SYSTEM** is written to a text file named *UG_2018_std.txt*, and for the 2017 **SYSTEM** it would be named *UG_2017_std.txt* etc. The **PARAMETER** *vargroup* allows for a group of variables to be output indicated by the initial letter (S) followed by the *symbol. The asterisk is a 'wild card' and thus for

example, *vargroup b** will output all variables in the input data set as well as any simulated variables that begin with the letter b (i.e. the benefit variables both original and simulated). However, it is also possible to output individual variables by using the [PARAMETER](#) *var* and specifying the name of the variable, though this is not used currently in UGAMOD. Just as wild cards can be used to output groups of variables this also is the case with income lists so, for example, *ILgroup ils** will output the value of all the standard [INCOMELISTS](#). The [PARAMETER](#) *nDecimals* determines the number of decimal places for monetary variables (any amounts with more decimal places are rounded). The [PARAMETER](#) *TAX_UNIT* defines the level of output. In UGAMOD it is set to an individual assessment unit (*tu_individual_ug*), which means that the output contains one row for each individual. If the [PARAMETER](#) is set to some larger unit (e.g. a household), there would be one row for each unit, where monetary variables refer to the sum over all unit members and non-monetary values refer to the head of the unit.

There is also the option to use the [PARAMETER](#) *defIL*, which stands for ‘definition [INCOMELIST](#)’, or in other words, the set of variables that are included within a particular [INCOMELIST](#) will be outputted. So, for example, specifying the [PARAMETER](#) *DefIL* as *ils_dispy* would mean that the output file would contain all variables included in standard disposable income. Using the [PARAMETER](#) *ILGroup* and specifying as *ils_dispy* instead results in only the overall value for standard disposable income being outputted.

3.3 TAX-BENEFIT POLICIES

Tax-benefit [POLICIES](#) usually contain the description of one tax or benefit, where this description is made up of [FUNCTIONS](#). Examples of some [FUNCTIONS](#) are given below.

- *Elig* determines eligibility/liability for benefits/taxes.
- *BenCalc* calculates the benefit/tax amount for all eligible units.
- *ArithOp* is a simple calculator, allowing for the most common arithmetical operations.
- *SchedCalc* allows for the implementation of the most common (tax) schedules.
- *Allocate* allows (re)allocating amounts (incomes, benefits, taxes) between members of assessment units.⁹
- *Min and Max* are simple minimum and maximum calculators.¹⁰

[FUNCTIONS](#) are described in detail in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab) and they are not elaborated in great detail here.

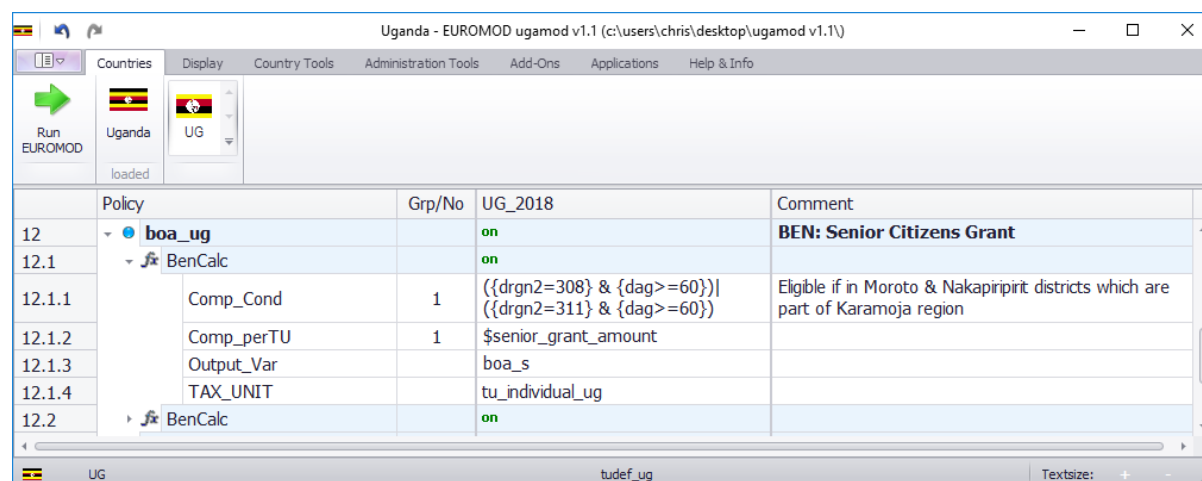
Tax-benefit [POLICIES](#) are described in general terms below using the Senior Citizen’s Grant as an example. The Senior Citizen’s Grant [POLICY](#) was given as an example in Section 3.1 and is explained in greater detail here. It is shown in Figure 3.10. The policy rules for the year 2018 are shown here as an example.

⁹This [FUNCTION](#) is not currently used in UGAMOD; further details can be found in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab).

¹⁰ This [FUNCTION](#) is not currently used in UGAMOD; further details can be found in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab).

Many of the social benefit **POLICIES** follow the same general structure, using the **FUNCTION** *BenCalc* to calculate eligibility for and amount of the grant. Alternatively, they use the **FUNCTION** *Elig* to calculate eligibility for the grant and the **FUNCTION** *ArithOp* to calculate the amount of grant for eligible individuals.

Figure 3.10: The **POLICY** *boa_ug*



	Policy	Grp/No	UG_2018	Comment
12	boa_ug		on	BEN: Senior Citizens Grant
12.1	BenCalc		on	
12.1.1	Comp_Cond	1	{{drgn2=308} & {dag>=60}} {{drgn2=311} & {dag>=60}}	Eligible if in Moroto & Nakapiririt districts which are part of Karamoja region
12.1.2	Comp_perTU	1	\$senior_grant_amount	
12.1.3	Output_Var		boa_s	
12.1.4	TAX_UNIT		tu_individual_ug	
12.2	BenCalc		on	

The **POLICY** *boa_ug* comprises two **FUNCTIONs** – both *BenCalc* **FUNCTIONs**. The two *BenCalc* **FUNCTIONs** are similar (they simply relate to different conditions prevalent in different areas so only the first **FUNCTION** is described here.

The *BenCalc* **FUNCTION** calculations are at the **individual** level. This is specified in the final parameter **TAX_UNIT** which has the value *tu_individual_ug*. The **PARAMETER** *Comp_Cond* (an abbreviation for ‘component condition’ is the eligibility condition that has to be fulfilled before an amount specified in *Comp_perTU* can be allocated. In this particular *BenCalc* there is one *Comp_Cond* and one *Comp_perTU*. However, the **FUNCTION** *BenCalc* can have a number of *Comp_Conds* and *Comp_perTUs* (or *Comp_perEligs*). The **PARAMETER** *Comp_Cond* specifies conditions of eligibility as regards both the age of the individual and the district in which s/he resides. **PARAMETER** *Comp_perTU* specifies that if *Comp_Cond* is fulfilled then the amount contained within the **CONSTANT** *\$senior_grant_amount* is allocated to the TU i.e. the individual.



Certain syntax rules have to be followed when writing the **PARAMETER** *Comp_Cond* or *elig_cond*:

First and foremost each condition must be included within curly brackets {...}.

A single condition {...} has either one component, e.g. a yes/no QUERY (e.g. {IsParent}) or two components separated by a comparison operator > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), = (equal to) or != (not equal to) (e.g. {ddi=1}).

There can be conditions which must be fulfilled e.g. {dag<16} and conditions which must not be fulfilled e.g. !{IsMarried}.

Conditions are combined by the logical operators & (and) and | (or) and parentheses 23 can be used to group conditions.



The **PARAMETER** formula in the **FUNCTION** *ArithOp* allows for the following operations:

- addition: operator +
- subtraction: operator –
- multiplication: operator *
- division: operator /
- raising to a power: operator ^, e.g. 2^3 (result: 8)
- percentage: operator %, e.g. *yem*3%* (result: *yem*(3/100)*)
- remainder of division: operator \, e.g. $22 \backslash 5$ (result: 2)
- minimum and maximum: operators <min> and <max>, e.g. $10 <min> 15$ (result: 10)
- absolute value: operator <abs>(), e.g. <abs>(-22) (result: 22), <abs>(50-70) (result: 20)
- negation: operator !(), e.g. *!(IsMarried)*, *!(17)* (result: 0), *!(0)* (result: 1)

to be used with the following operands:

- numeric values, e.g. 10, 0.3, -25
- numeric values with a period, e.g. $12000\#m$, $1000\#y \#i$ as place holders for numeric values specified by footnote parameters variables, e.g. *yem*
- INCOMELISTS, e.g. *ils_dispy*
- queries, e.g. *IsUnemployed*
- random numbers, *rand*

Order of operation rules:

- ^, <min>, <max>, <abs>, (), !(), %
- before multiplicative operations */ \
- before additive operations + –

Parentheses can be used to group operations, e.g. $(2+3)*4$.

The remainder of this section comprises points of particular relevance to UGAMOD, but again it is recommended that the user refers to the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab) for a more comprehensive introduction to the **FUNCTIONS**.

Common parameters

COMMON PARAMETERS are provided by many of the **FUNCTIONS** and can be classified into four categories:

- **COMMON PARAMETERS** affecting output. An example already encountered is *output_var* which is provided by all tax-benefit **POLICY FUNCTIONS**. Other options are *output_add_var* (with *output_var* any existing value is overwritten, but with *output_add_var* the result is added to any existing value of the output variable) and *result_var* (allows a second output variable, generally

used in combination with *output_add_var*). These are provided by all **FUNCTIONS** that have the **PARAMETER** *output_var* (except *Elig* where it is not meaningful to add to a yes/no variable).

- **COMMON PARAMETERS** affecting eligibility. Examples already encountered are *who_must_be_elig* and *elig_var*. These are provided by all tax-benefit **POLICY FUNCTIONS**.
- **COMMON PARAMETERS** limiting results. There are three **PARAMETERS** in this category: *lowlim* (for setting a lower limit), *uplim* (for setting an upper limit) and *threshold* (to define a threshold). These are provided by all tax-benefit **POLICY FUNCTIONS**.
- The **COMMON PARAMETER** *TAX_UNIT*. As we have seen, this allows for the definition of an assessment unit to which a **FUNCTION** refers and is compulsory for all tax-benefit **POLICY FUNCTIONS**.

There are also features that control whether a **FUNCTION** is processed, including the switch (for turning off **FUNCTIONS** - not a **PARAMETER** per se) and the **PARAMETER** *run_cond* which allows for conditional processing of the **FUNCTION** (in other words the **FUNCTION** is only carried out if the respective condition is fulfilled). These are provided by all **POLICY FUNCTIONS**.

Further detail on **COMMON PARAMETERS** can be found in the EUROMOD Functions section of the EUROMOD Help (accessed from the *Help & Info* tab).

Interpreting conditions with respect to assessment units

There is an issue of interpretation if **PARAMETERS** taking variables, **INCOMELISTS** and queries as their values are used with assessment units comprising more than one person.

The following rules of interpretation apply:

- **PARAMETERS** relating to monetary variables and **INCOMELISTS** are interpreted at the level of the assessment unit (defined by the **PARAMETER** *TAX_UNIT*). Values are therefore added up over all members of the unit.
- **PARAMETERS** relating to non-monetary variables and individual level **QUERIES** are interpreted at the level of the individual, if a condition, or at the level of the head of the assessment unit for all other **PARAMETERS**.
- **PARAMETERS** relating to non-individual level **QUERIES** are interpreted in varying ways.

It is possible to change the assessment unit generally used by the **FUNCTION** for single variables, **INCOMELISTS** or **QUERIES**.

Footnotes

There are **PARAMETERS** which serve to further specify other **PARAMETERS**. These are referred to as footnote **PARAMETERS** (or **FOOTNOTES**). They can be easily identified by names starting with the character # with an integer number, *i*, given in the *Grp/No* column, e.g. *#_amount1* and *#_level 1*. **FOOTNOTES** are applicable with several **FUNCTIONS**, specifically all **FUNCTIONS** providing formula and/or condition **PARAMETERS**.

The four most commonly used types of **FOOTNOTE** are:

1. Limits – it is sometimes necessary to set limits (lower, upper, threshold) to **FUNCTION** results and single operands.
2. Amounts – it is sometimes more transparent to indicate amounts outside the formula (particularly in a complex formula and/or implementation of several policy years).
3. Assessment units – it is possible to change the **FUNCTION**'s assessment unit (indicated by the **PARAMETER TAX_UNIT**) for a single operand.
4. **QUERIES** – a few queries need further specification (e.g. *nDepChildrenInTaxunit* counts the number of dependent children in the assessment unit and has two optional **PARAMETERS**, *#_AgeMini* and *#_AgeMaxi*, which allow you to specify the age of the dependent children).

Amounts

Amount **PARAMETER** values are usually followed by their 'period'. It is good practice to always indicate a period, although *#m* (monthly) has no real effect as EUROMOD internally converts all amounts to monthly and would assume a monthly amount if no period was specified. Only *#m* (monthly – no conversion) and *#y* (yearly – divided by 12) are used in UGAMOD, but other options are possible, for example quarterly (*#q*), weekly (*#w*) and daily (*#d*). In general it makes sense to specify amounts as the time period in which they are commonly discussed (e.g. benefit/grant amounts in monthly terms, income tax thresholds in annual terms).

Interactions between functions

Usually more than one **FUNCTION** is used to calculate a benefit or tax, which means that the **FUNCTIONS** interact in some way. These interactions can be classified in four categories:

- Condition: one **FUNCTION** (usually *Elig*) evaluates a condition and a subsequent **FUNCTION** operates on the basis of the result of this evaluation.
- Input: one **FUNCTION** calculates some result, which is used as an input by a subsequent **FUNCTION**.
- Addition: one **FUNCTION** calculates a part of a **POLICY** and a subsequent **FUNCTION** calculates another part of the **POLICY** and therefore needs to add to the first part.
- Replacement (actually not a real interaction): a subsequent **FUNCTION** replaces the result of a precedent **FUNCTION**, which of course only makes sense if the result of the first **FUNCTION** is used in between.

3.4 GENERAL SETTINGS

The following is a list of some of the key actions which can be undertaken from the *Country Tools* tab:

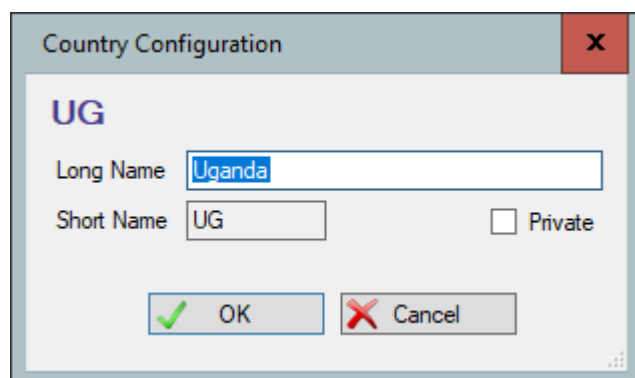
- configuring country settings (e.g. name, short name), [SYSTEM](#) settings (e.g. currency) and the input datasets which can be used for simulating the country's tax benefit system;
- adding or deleting [SYSTEMS](#);
- searching – standard search and replace and help with finding errors and assessing if and where certain EUROMOD components (e.g. variables) are used in the respective country's [PARAMETERS](#) (see Working with EUROMOD - Searching section of the EUROMOD Help, accessed from the *Help & Info* tab);
- formatting tools, e.g. highlighting with colours and setting bookmarks (see Working with EUROMOD –Formatting section of the EUROMOD Help, accessed from the *Help & Info* tab).

The first of these actions is described in this section. Adding or deleting [SYSTEMS](#) is dealt with in Section 4.2.

There are three buttons to the left hand side of the *Country Tools* ribbon: *Country*, *Systems* and *Databases*.

Country

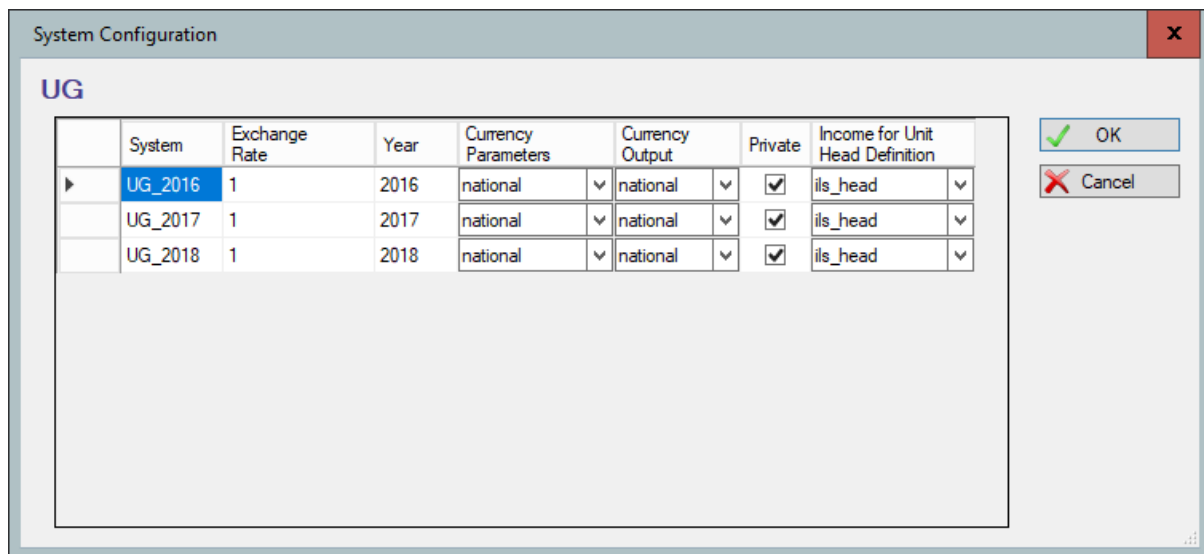
Figure 3.13: The *Country Configuration* dialog



In the *Country Configuration* dialog the *Long Name* of the country can be changed, however the *Short Name* of the country cannot be changed by the user as it needs to correspond to the filenames of the country XML files.

Systems

Figure 3.14: The *System Configuration* dialog



All systems and their settings are listed and can be changed in the *System Configuration* dialog (see Figure 3.14). The settings include:

Exchange Rate: Indicates the rate used by the model to convert national currency into Euro or vice versa (i.e. Euro amounts are multiplied by the rate to get amounts in national currency). This setting is not relevant in UGAMOD and so is set to 1.

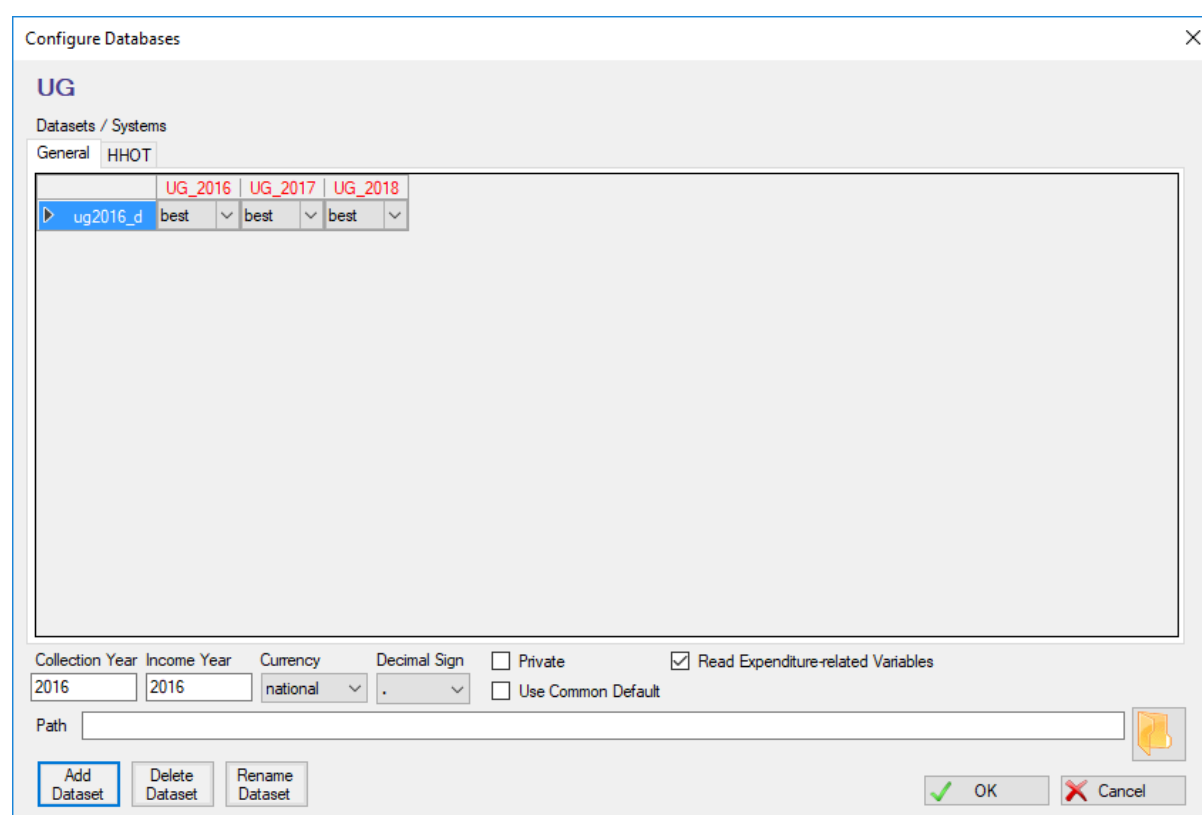
Currency Parameters: Indicates the currency used for monetary **PARAMETER** values. It is set to *national* in UGAMOD, which means the national currency is used (the alternative in EUROMOD would be *euro* but this is not relevant for Uganda).

Currency Output: Indicates the currency used within the output file (e.g. *UG_2017_std.txt*), which again is set to *national* (the alternative in EUROMOD would be *euro*).

Income for Unit Head Definition: Indicates which **INCOMELIST** (from the dropdown list) is to be used as a default for determining who the head of the assessment unit is. The default setting is *ils_origy*. But for SOUTHMOD countries this is set to *ils_head*.

Databases

Figure 3.15: The *Configure Databases* dialog



The concept of **SYSTEM**-dataset combinations was introduced in Section 3.1. There are various options relating to databases:

Assigning datasets to systems

The upper part of the *Configure Databases* dialog (see Figure 3.12) shows a table where the row headers list all datasets available for the country, while the column headers list all available **SYSTEMS**. The intersection of a dataset (row) and system (column) indicates whether the **SYSTEM** can be run with the dataset, in other words whether they form a so-called **SYSTEM**-dataset combination (as described in Section 3.1). There are three possible settings: a cross (x) indicates that the dataset and the **SYSTEM** form a **SYSTEM**-dataset combination; *best* denotes a **SYSTEM**-dataset combination that is a 'best match' (as described in Section 3.1), and *n/a* means that the **SYSTEM** cannot be run with the dataset.¹¹ In UGAMOD v1.x there is only one dataset and so this is the 'best match' for the **SYSTEMS** *UG_2016*, *UG_2017* and *UG_2018*.¹²

¹¹ Note that, if a system is copied (see Section 4.2), the datasets assigned to the original system are automatically also assigned to the copied system. Use the *Configure Databases* dialog to change this, if necessary.

¹² Though as mentioned above it is good practice to set the best match flag only for **BASELINES**.

Adding, removing or renaming a dataset

To add a dataset click the *Add Dataset* button, which opens a file search dialog allowing for the search of a text file containing data suitable to run (one or more of) the [SYSTEMS](#).

To delete a dataset, select it and click the *Delete Dataset* button. Note that the dataset is removed without any further warning, but you can still undo this action by closing the dialog with the *Cancel* button or by using the undo functionality. The dataset is not deleted physically of course; the removal concerns only the ability to use the dataset within UGAMOD.

To rename a dataset, select it and click the *Rename Dataset* button, which opens a textbox where the new name can be entered.

Settings of the selected dataset

Below the *Datasets/Systems* table the dialog shows the settings of the selected dataset:

Collection Year: Indicates the year the data were collected.

Income Year: Indicates the year to which the monetary values within the data refer.

Currency: Indicates in which currency data are stored. This is set to *national* in UGAMOD (*euro* would be an option in EUROMOD).

Decimal Sign: Indicates whether data uses point (.) or comma (,) as the decimal sign.

Path: Indicates a specific path to locate the dataset. Usually it is left empty, to instruct the model to locate the dataset at the default path.

Use Common Default: Checking this option means that any variable not in the data, but used by the [SYSTEM](#), is set to zero (i.e. no error message is issued).

With all of these buttons, click *OK* to confirm any changes or *Cancel* to close the dialog without any consequences. Note that changes are only definite once the project is saved. Before that you can still use the undo functionality (see the Working with EUROMOD - Undo and redo section of the EUROMOD Help, accessed from the *Help & Info* tab) or close the project without saving.

For further information see the Working with EUROMOD - Changing Countries' Settings section of the EUROMOD Help (accessed from the *Help & Info* tab).

3.5 VARIABLES

All countries implemented in EUROMOD and based on the EUROMOD framework (e.g. UGAMOD) use the same set of variables to store information taken from the input data and information generated by the model.

Variables follow specific naming conventions. All variables generated by the model end with *_s* for simulated, whereas variables in the input data do not have such an ending (e.g. *bch* is a child benefit taken from the data i.e. reported receipt, whereas *bch_s* is a simulated child benefit). The first character of a variable's name indicates its type and the rest of the name is composed of two-character acronyms:

b = benefit (e.g. *bsa*: b=benefit, sa=social assistance)
t = tax (e.g. *tin*: t=tax, in=income)
p = pension (e.g. *poa*: p=pension, oa=old age)
d = demographic (e.g. *dag*: d=demographic, ag=age)
l = labour market (e.g. *les*: l=labour market, es=economic status)
y = income (e.g. *yem*: y=income, em=employment)
a = assets (e.g. *afc*: a=assets, fc=financial capital)
x = expenditure (e.g. *xhh*: x=expenditure, hh=household)
k = in kind (e.g. *ked*: k=in kind, ed=education)

Variables are stored in the [VARIABLE DESCRIPTION FILE](#) (*VarConfig.xml* file). The file contains the names and properties of all variables available in the model. Some of the properties help the model to distinguish if certain routines should be applied to the variable (e.g. only monetary variables are uprated). Other properties are descriptions of the variables, including an automatically generated description of each variable and columns which allow for a special description for the country in question. The file also stores the acronyms of which the variable names are composed.

The UGAMOD user interface provides a tool for administering the information stored in [VARIABLE DESCRIPTION FILE](#) (*VarConfig.xml* file). To access this tool click on the button *Variables* in the *Administration Tools* tab. To the left hand side is a list of all available variables. For each variable there is a checkbox indicating whether the variable is monetary (checked) or not (not checked). There is a verbal description (*Automatic Label*) which is automatically generated out of the acronyms building the variable's name and the user cannot edit it.

To the right hand side is a list of all available acronyms, organised into three levels. The first level is the type as described above (i.e. benefit, tax, income, ...). The second level subdivides types into different categories and the third level shows the acronyms themselves together with a verbal description, which is used to generate the *Automatic Labels* of the variables. If an acronym is categorical (e.g. gender has two categories: male and female), selecting the acronym lists the respective categories below the list of acronyms.

A number of operations can be performed within the tool, all of which are described in the Working with EUROMOD –Administrating variables section of the EUROMOD Help (accessed from the *Help & Info* tab):

- Adding variables (see also Section 4.4)
- Changing the name of a variable
- Changing the monetary state of a variable
- Changing the country specific descriptions of a variable
- Deleting variables
- Filtering variables
- Sorting variables
- Searching variables
- Adding acronyms

As has been shown, it is also possible to add variables using the DefVar FUNCTION. Such variables fall outside the standard EUROMOD variable naming conventions. They are used sparingly as their use impedes harmonisation of variables across different SOUTHMOD models.

In fact, there are just two general uses of such variables within UGAMOD. The first is for temporary or intermediate variables within policies which are generally not output to the final output file (except for the purpose of debugging). These all begin with i_ (i for intermediate). The rest of the name is as meaningful as possible.

The other exception is VAT/Excise expenditure variables which all begin with an x and quantity variables for excise duties which begin with q. These are then followed by the COICOP code for the item in question. Because there are a very large number of these items, the underpinning software has been amended to enable all x and q variables to be automatically read into the model provided that the box [Read expenditure related variables] has been checked in the Configure Databases dialogue box (activated by clicking 'databases' in the in the Country Tools tab (see figure 3.15 above). For these cases the Variables Tool is not required.



UGAMOD can be summarised as follows:

1. UGAMOD's central access point is the main user interface. From here the content files can be accessed. These store the information the model needs for its calculations and provide other tools and applications.
2. UGAMOD's content files contain information required for both the implementation of the framework of the tax-benefit model, and for the implementation of the particular policies which comprise the tax-benefit system. This information is mainly contained in **POLICIES** displayed in the **POLICY SPINE**.
3. **FUNCTIONS** are used as building blocks of both definitional and tax-benefit **POLICIES**. **FUNCTIONS** are comprised of **PARAMETERS**.
4. The definitional **POLICIES** include *uprate Ug*, *ildef Ug*, *tundef Ug* and *constdef Ug*.
5. *Uprate Ug* contains the uprating factors for monetary variables.
6. *Ildef Ug* contains definitions of **INCOMELISTS** (i.e. aggregates of variables defining for example disposable income, taxable income, etc.). **INCOMELISTS** are used in tax/benefit **POLICIES** for the implementation of the respective tax or benefit.
7. *Tundef Ug* contains definitions of assessment units, which are also used in tax-benefit **POLICIES** for the implementation of the respective tax or benefit.
8. *Constdef Ug* contains definitions of **CONSTANTS** used in tax-benefit **POLICIES**.
9. There is also a definitional **POLICY** *output_std Ug* which contains the specification of the output from the model.
10. Information relating to country settings (e.g. name, short name), **SYSTEM** settings (e.g. currency) and the input datasets is contained and modified within *Country Tools*.
11. The **VARIABLE DESCRIPTION FILE** (*VarConfig.xml* file) contains descriptions of all variables available in the model.

4. TASKS IN UGAMOD

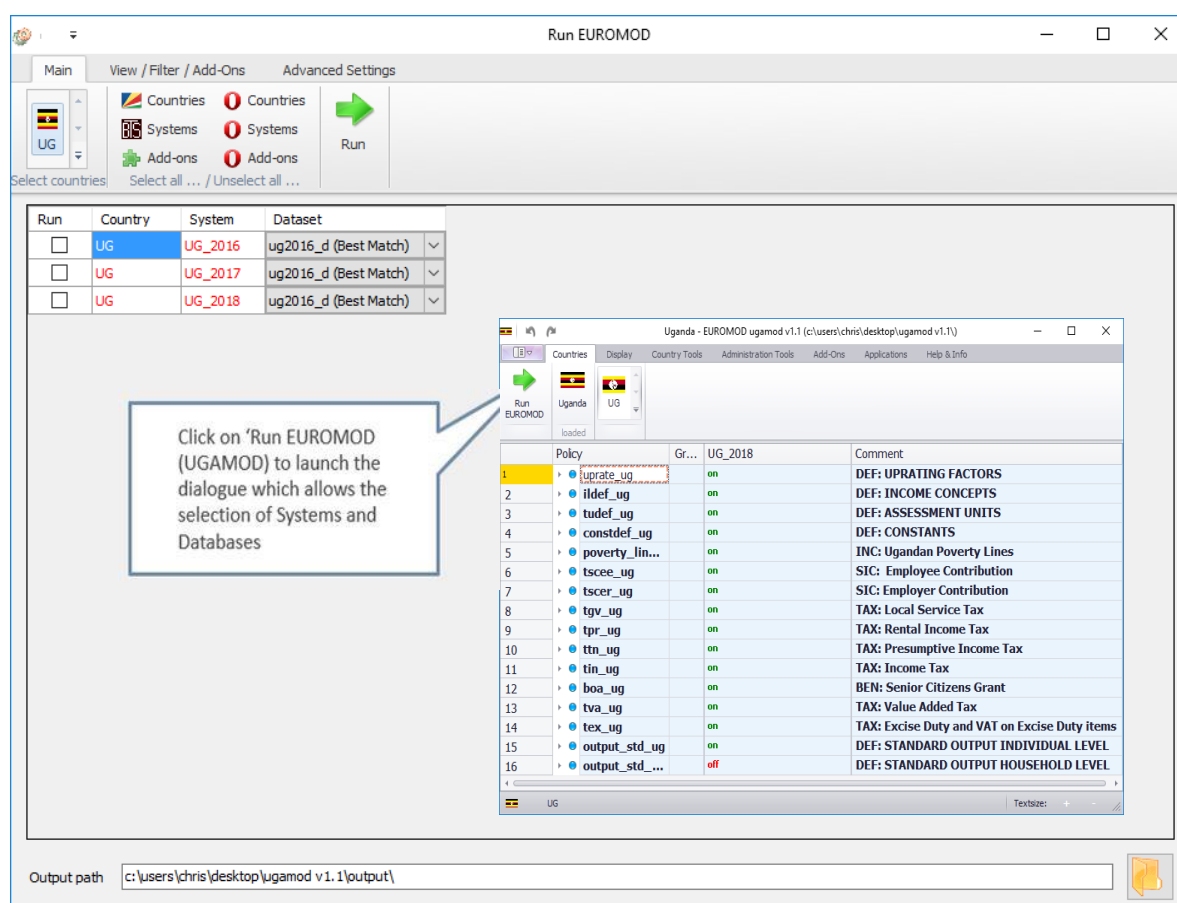
4.1 RUNNING UGAMOD

The next step, having implemented each of the [POLICIES](#) and configured the settings in UGAMOD, is to run the model to simulate the tax and benefit policies.

As we have seen, UGAMOD output is based on two inputs: (a) household micro-data and (b) rules on how to calculate taxes and benefits stored in the content file. Using these two information sources, the model calculates all taxes and benefits that have been implemented. The calculations are carried out for each individual or household in the dataset and the result is written to a micro-output file. The output is at the individual level (unless specified otherwise).

To activate the *Run UGAMOD* dialog (see Figure 4.1) click on the *Run UGAMOD* button in the top left corner of the user interface (see inset of Figure 4.1).¹³

Figure 4.1: The *Run UGAMOD* dialog



¹³ Currently these both say *Run EUROMOD*.

The main part of the dialog is a list of **SYSTEMS** which are ready to run. UGAMOD only covers Uganda and so the *Country* column only shows **SYSTEMS** for Uganda (*UG*) but it has the capability (though not the content) to cover other countries as well. To select a **SYSTEM** for running check the box to the right of the **SYSTEM**. The list provides a drop down box for each **SYSTEM** which contains all available datasets. As a default these boxes show the best matching datasets. It is however possible to choose another dataset by selecting it from the list. In UGAMOD v 1.1 there are three **SYSTEMS** (*UG_2016*, *UG_2017* and *UG_2018*) and just one dataset (*ug2016_d data*).

The buttons to the left of the *Run* button facilitate the selection of more than one country and/or **SYSTEM**.

The field *Output path* at the bottom of the dialog defines the folder where the model writes its output. By default this is the output folder defined when installing UGAMOD (or when opening a project – see Section 4.5). The folder can be changed by clicking the folder button right of the field to browse and select the appropriate folder, or by typing. Please note that the output folder must exist, otherwise UGAMOD issues an error message.

Once the required selections have been made, click on the *Run* button to start the simulation process. A window appears providing information about the progress of the run and allowing for some manipulation. All **SYSTEM**-dataset combinations selected for running are listed and their status is shown: *running*, *queued*, *finished* or *aborted* (either by the user or due to an error). Once a run is started, its starting time is displayed, and once it is finished (or aborted), the finishing time is indicated as well, together with the time taken. The total time needed by the simulation depends on the processing speed of the computer, on the size of the dataset (i.e. how many households must be processed), and on the extensiveness of the **SYSTEM** (i.e. the number and complexity of implemented taxes and benefits).

There are three buttons for each run. The *Stop* button enables the user to abort the run. Once a run is started, the *Run Log* button is activated. If it is clicked, the field below the list of runs shows progress information. If a run produces an error (stopping the run) or a warning (allowing the run to continue) the *Error Log* button is activated. Clicking the button shows the run's warnings and/or errors in the field below the list of runs. Note that the content of this field is determined by the most recently clicked button - its heading indicates what is currently displayed. If any warnings or errors are issued, an error log file is generated, named *yyyymmddhhmm_errlog.txt* (e.g. *201709081530_errlog.txt*).

The information window will stay open until it is closed by the user, even if all runs are finalised, to allow possible error logs to be checked and to inform about the times taken. If the user closes the window before all runs are finished (after a warning), the still active runs are aborted and the queued runs are taken from the queue. To hide the window, use the minimise button.

When UGAMOD has finished its calculations, the output is stored as one or more text files at the storage place defined in the field *Output path*. An output text file is produced: the **STANDARD OUTPUT**. Output files can be viewed in a text editor program such as notepad or imported into any statistical analysis package, for example Stata, for more detailed analysis. The output can be viewed

in Excel by using the in-built tool *Open Output File* (accessed from the *Applications* tab). The output data can also be analysed using the [STATISTICS PRESENTER](#) (See Section 4.5).

The model produces a header file relating to the output file with the following information: system, database, EUROMOD version number, user interface version number, executable version number, start date and time, end date and time, name and location of the output file, currency and exchange rate. The name of this file is *yyyymmddhhmm_EMHeader.txt* (e.g. *201709081530_EMHeader.txt*).



How to produce output:

1. Enter the user interface and click on the button *Run UGAMOD* in the top left corner.
2. Check the box for the [SYSTEM](#)-dataset combination of choice. Also ensure that the path to the output file is correct.
3. Click on *Run*.
4. When the simulations have finished running use your favourite text or statistical package to explore the output files or utilise the [STATISTICS PRESENTER](#).

[SYSTEMS](#) will need to be added for future years, for example the policy rules for 2018 will need to be incorporated as a new [SYSTEM](#) in due course. In addition, there may be a need to test the impact of reforms to the current tax and benefit rules. These two related tasks are discussed next.

4.2 ADDING A NEW SYSTEM IN UGAMOD

There is a tool in UGAMOD that allows you to add a new [SYSTEM](#). There are two ways to access this tool: either click on the *Add System* button in the *Country Tools* tab or right click on the header of an existing [SYSTEM](#) and select *Copy/Paste System*. Either option opens a dialog which asks for the new [SYSTEM NAME](#) (e.g. a [SYSTEM](#) for 2019 could - and should! - be named *UG_2019*) and the [SYSTEM YEAR](#) (in the example this would be 2019). Note that the [SYSTEM](#)'s name must not contain any other characters than letters, numbers and underscores. If any other character is used or if the chosen name is equal to an existing [SYSTEM](#)'s name, an error message is issued, and you are asked to change the name. Clicking *OK* adds the new [SYSTEM](#).

In the first instance the new [SYSTEM](#) is almost an exact copy of the base [SYSTEM](#) and is automatically configured to run with the same datasets as in the base [SYSTEM](#). There is just one small difference between the base [SYSTEMS](#) and their copies: the *Add System* tool changes the names of the output files for the new [SYSTEM](#) to reflect the new [SYSTEM](#)'s name (e.g. the output file for *UG_2018* is called *UG_2018_std.txt*, but in a new [SYSTEM](#), say *UG_2019*, the name of the output file would be *UG_2019_std.txt*).

The user interface allows the differences between a base and a derived [SYSTEM](#) to be highlighted, using background and/or text colour (Tab *Display – Automatic Conditional Format* button). This is

very useful to highlight the changes introduced by a reform (see below). If the base **SYSTEM** highlights differences to its own base **SYSTEM**, the new **SYSTEM** will inherit this feature.

Changes to **PARAMETERS** (e.g. inputting the 2019 values) can now be made in the new **SYSTEM**.

Deleting a **SYSTEM** can similarly be undertaken in two ways: either click the *Delete System(s)* button in the *Country Tools* tab, which opens a dialog (*Select Systems*) where the **SYSTEM** to be deleted can be selected, or right click the **SYSTEM**'s header and select *Delete System*.



Note the warning that **SYSTEM(S)** will be deleted permanently. Take great care when using the *Delete System* tool in order to avoid inadvertently deleting a **SYSTEM** that you did not intend to delete.



Conditional formatting can be used to highlight the differences between **SYSTEMS**. Under the *Display* tab you can choose *Automatic Conditional Formatting* to highlight the differences between the newest **SYSTEM** and the previous **SYSTEM**. Alternatively you can specify your own *Conditional Formatting*.

4.3 IMPLEMENTING A POLICY REFORM IN UGAMOD

In order to implement a policy reform, it is necessary to remember that a **SYSTEM** is a collection of tax and benefit policies that apply to a particular point in time, for example the *UG_2018 SYSTEM* records the tax and benefit rules in existence in 2018.

Understanding how to correctly implement a policy reform is a crucial part of using UGAMOD. There are two ways of doing this but only one way which is regarded as good modelling practice. The process of implementing a reform should begin with adding a new **SYSTEM** to UGAMOD (as described above). This could be called *UG_2018_reform*, for example, and can be made to copy an existing **SYSTEM**. Having done this all the changes can be made in the reform **SYSTEM** and the **PARAMETERS** in both the existing **SYSTEM** and the reform **SYSTEM** will be preserved so that it is easy to see the differences between them and simulations can easily be run for both **SYSTEMS**.

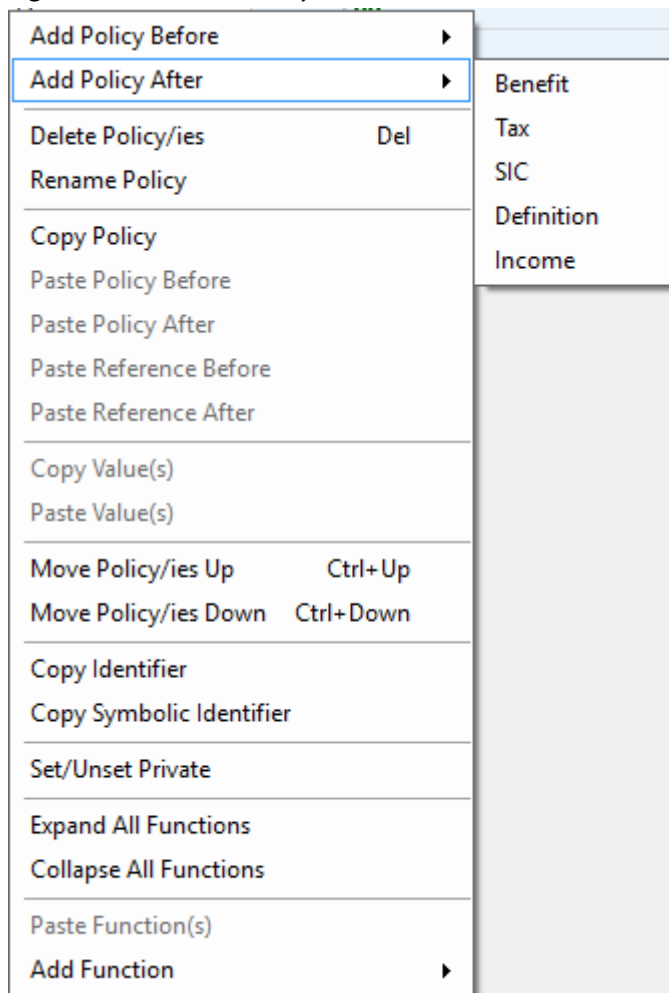
This is preferable to making changes to particular policies in an existing **SYSTEM**, which although possible is definitely not recommended. The existing **SYSTEM** is a record of the **POLICY PARAMETERS** at a particular time point. If it is amended then this record is lost and it would not be possible to run simulations on the existing **SYSTEM** without undoing the changes made in the reform scenario.

The exact process of implementing a policy reform will then depend on whether the reform is a change to an existing policy (e.g. amending a means test threshold or grant amount, removing a means test, or varying the means test amount by criteria) or the introduction of a new policy. The possible options are too many and varied to explain in detail here, so instead the main operations that will be required when implementing a policy reform are described next.

Adding a policy

To add a **POLICY** right click on the name of the **POLICY** either before or after which you want to insert the new **POLICY**. This opens the **POLICY**'s context menu, where you select the menu item *Add Policy Before* or *Add Policy After* (see Figure 4.2). A sub menu opens where the type of the new **POLICY** can be chosen (*Benefit, Tax, ...*). Click the respective **POLICY** type to open a dialog where you are asked to indicate the new **POLICY**'s name. Clicking *OK* adds the new **POLICY**. Note that the **POLICY**'s name must not contain any characters other than letters, numbers and underscores. If any other character is used or if the chosen name is equal to an existing **POLICY**'s name, an error message is issued, and you are asked to change the name. Moreover, the **POLICY** name is by convention expected to end with *_ug*. If this is not the case the user interface asks whether it should add this ending for you. You may answer this question with *No* to use a non-standard policy name, it is however recommended to answer with *Yes*.

Figure 4.2: The *Add Policy* tool



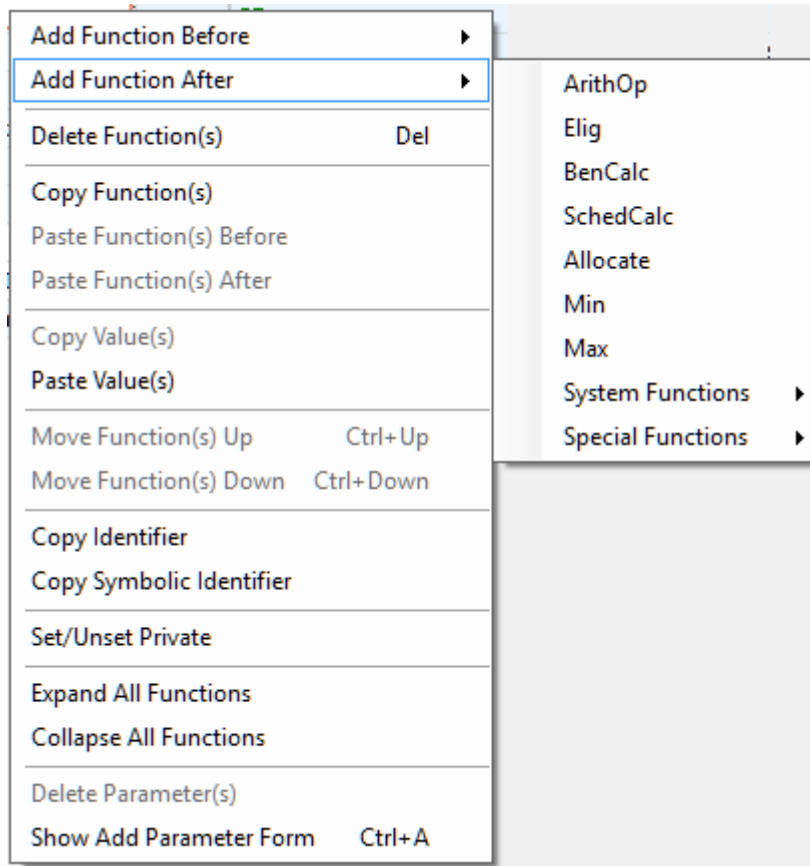
Adding a function

To add a **FUNCTION** to a **POLICY** right click on the name of the **FUNCTION** either before or after which you want to insert the new **FUNCTION**. This opens the **FUNCTION**'s context menu, where you

select the menu item *Add Function Before* or *Add Function After*. A sub menu opens where the **FUNCTION** to be added can be chosen (see Figure 4.3).

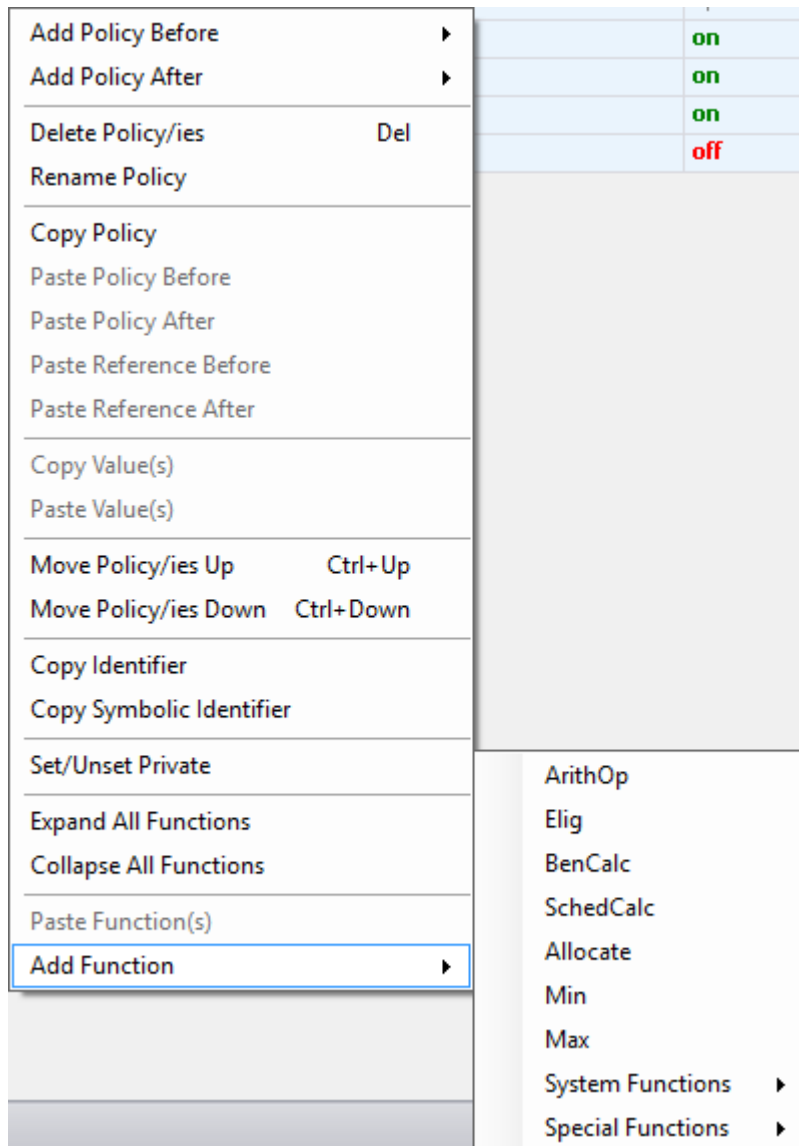
This sub menu is slightly different depending on the **POLICY** that the new **FUNCTION** will be part of. Click the respective **FUNCTION** to add the **FUNCTION** itself as well as the compulsory **PARAMETERS** of this **FUNCTION** (e.g. for most **FUNCTIONS**, *TAX_UNIT* and *Output_var*).

Figure 4.3: The *Add Function* tool accessed from **FUNCTION**'s context menu (right click on preceding or succeeding **FUNCTION**)



Alternatively, the **POLICY** context menu offers the menu item *Add Function* (see Figure 4.4). This adds the **FUNCTION** as the very last **FUNCTION** of the **POLICY**.

Figure 4.4: The *Add Function* tool accessed from **POLICY**'s context menu (right click on **POLICY** name)



Adding a parameter

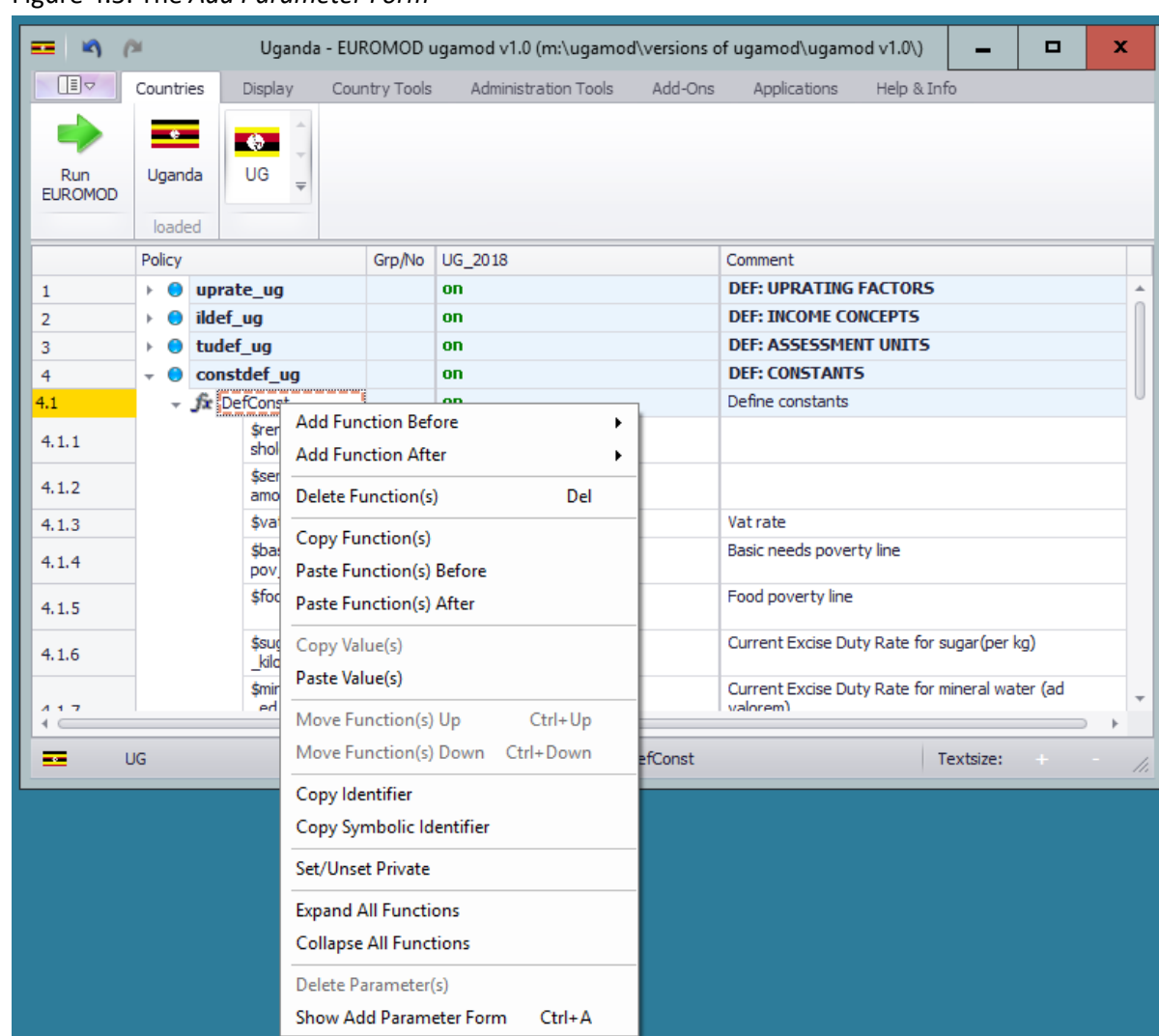
To add a **PARAMETER** to a **FUNCTION** right click the respective **FUNCTION** to open its context menu and select the menu item *Show Add Parameter Form* (see Figure 4.5).

The form shows all **PARAMETERS** that can be added to the **FUNCTION** (in the column *Parameter*) with a description (in the column *Description*). Compulsory **PARAMETERS** of the **FUNCTION** are not listed (e.g. the **PARAMETER** *TAX_UNIT*). Exceptions to this rule are 'special' **PARAMETERS** such as those which can be added more than once (e.g. the **PARAMETER** *Var* of the **FUNCTION** *DefOutput* or the **PARAMETER** *Comp_Cond* of the **FUNCTION** *BenCalc*) and those which have 'aliases' (e.g. the **PARAMETER** *Output_Var* with its alias *Output_Add_Var*). The form observes what the user is doing and adapts its content respectively.

In brief, to add one or more **PARAMETERS** to the **FUNCTION** select them by ticking the corresponding check boxes. Then click the *Add* button (the button with the green plus). It is possible to add **PARAMETERS** in bulk (where more than one incidence is allowed e.g. the **PARAMETER** *Var* of the **FUNCTION** *DefOutput*). To understand the full functionality of the *Add Parameter Form* see the Working with EUROMOD - Changing countries' tax-benefit systems - Adding parameters section of the EUROMOD Help (accessed from the *Help & Info* tab).

Note that if in the main view a **PARAMETER** is selected, new **PARAMETERS** are added after this **PARAMETER**, whereas if a **FUNCTION** is selected (as in Figure 4.3), new **PARAMETERS** are added at the end of the **FUNCTION**.

Figure 4.5: The *Add Parameter Form*



Add Parameters

DefConst (order: 1) in policy constdef Ug

Add	Parameter	Replaces	Grp/No	Count	Description
<input type="checkbox"/>	Const_Dataset		24	1	If set, constant is only defined if the respective dataset is used for the run.
<input type="checkbox"/>	Const_SystemYear		24	1	If set, constant is only defined if the run concerns the respective system year.
<input type="checkbox"/>	[Placeholder]		24	1	[Placeholder] stands for the name of the constant, which is defined in the policy column. The val...
<input type="checkbox"/>	Run_Cond				Function is only carried out if the condition is fulfilled. The parameter is intended to be a condition...
<input type="checkbox"/>	#_LimPriority		1	1	Footnote parameter for the further specification of an operand: Possible values: If upper limit (#_U...
<input type="checkbox"/>	#_LowLim		1	1	Footnote parameter for the further specification of an operand: replaces operand if operand is sm...
<input type="checkbox"/>	#_UpLim		1	1	Footnote parameter for the further specification of an operand: replaces operand if operand is hig...
<input type="checkbox"/>	#_Amount		1	1	Footnote parameter for the further specification of an operand: indicates the numeric value of an ...
<input type="checkbox"/>	#_DataBasename		1	1	Parameter of query IsUsedDatabase.

☒ Show Common Parameters
☒ Show Footnote Parameters

Description (F5) Summary (F6)

+ Add - Close

Having introduced a new policy, it is necessary to amend certain INCOMELISTS. If it is a new benefit, the INCOMELIST `ils_bensim` will need to be amended to include the new benefit output variable. If it is a new direct tax policy, the new output variable must be added to the INCOMELIST `ils_taxsim`. In addition certain INCOMELISTS will need amendment to ensure that STATISTICS PRESENTER takes the new policy into account for the summary statistics (see 4.5 below).

For further information on adding [POLICIES](#), [FUNCTIONS](#) and [PARAMETERS](#) see the Working with EUROMOD - Changing countries' tax-benefit systems section of the EUROMOD Help (accessed from the *Help & Info* tab).



To make the process of implementing a reform more manageable, sometimes it is helpful to concentrate only on this [SYSTEM](#) and maybe its base [SYSTEM](#). To facilitate this, [SYSTEMS](#) can be hidden. Right click on a [SYSTEM](#)'s header and then move the mouse over the menu item *Move to Hidden Systems Box ...* to reveal various sub menu items where there are options for hiding and un-hiding [SYSTEMS](#) from the main view. [SYSTEMS](#) hidden from the main view are listed in the *Hidden Systems Box*. This is a small window, which is displayed by choosing the sub menu item *Show Hidden Systems Box* or by any of the other sub menu items except *Unhide All Systems*. The sub menu item *Unhide all Systems* redisplay all hidden [SYSTEMS](#) (i.e. the [SYSTEMS](#) listed in the *Hidden System Box*). To redisplay a single [SYSTEM](#), double click on the [SYSTEM](#) in the *Hidden System Box*. It is also possible to hide and unhide a [SYSTEM](#) by dragging it into the *Hidden System Box* or by dragging it from the *Hidden System Box* to its old or any other position.

Note that the user interface draws attention to any changes which could affect hidden [SYSTEMS](#).

4.4 ADDING NEW VARIABLES TO UGAMOD

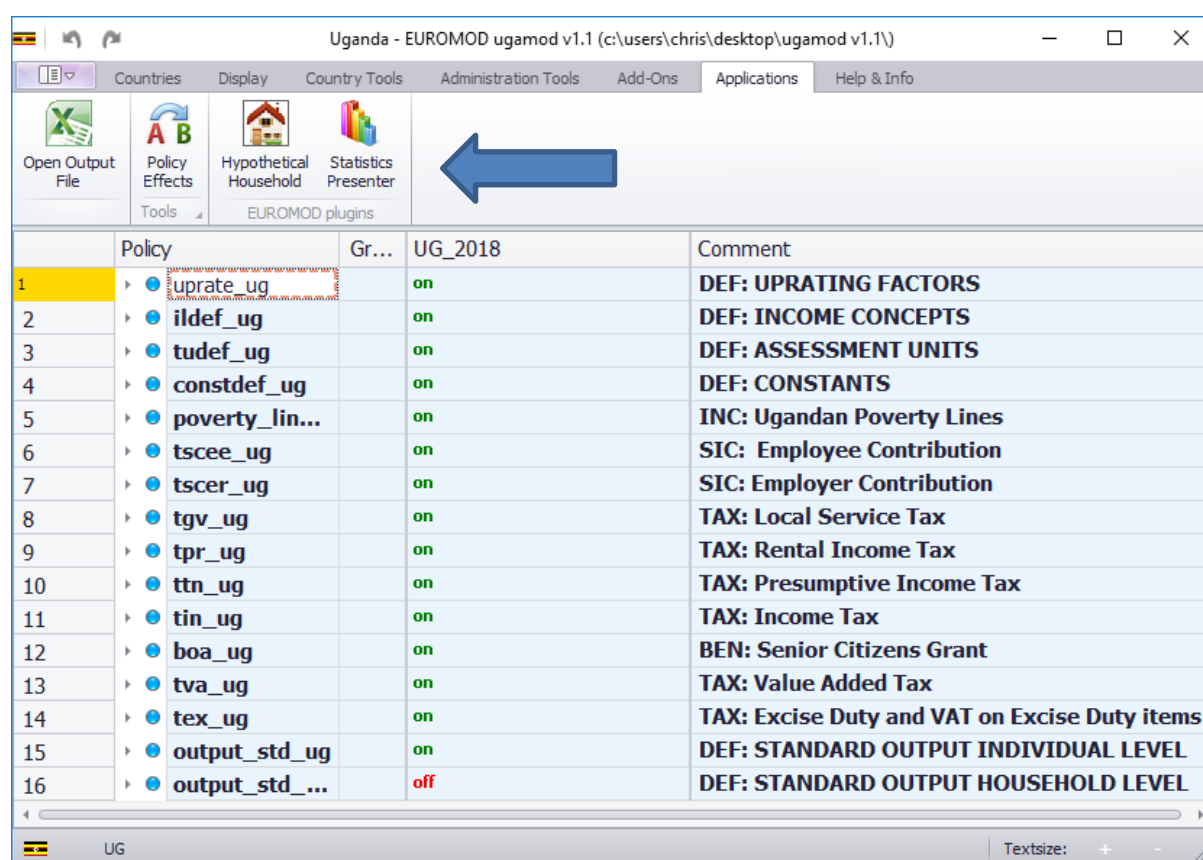
As part of implementing a reform or a new [SYSTEM](#) where there has been a change to a policy, it may be necessary to incorporate new variables. As described above, the [VARIABLE DESCRIPTION FILE](#) (*VarConfig.xml* file) is accessed via the button *Variables* in the *Administration Tools* tab. In order to add a variable click on the button *Add Variable* in the top left hand corner of the *Variables* tab or alternatively press the keys *Alt* and *V* simultaneously. This adds an empty row to the list of variables. Initially the row is added below the selected row. Re-sorting the list (manually or by an automatic update due to another change) moves empty rows to the beginning (ascended sorting) or end (descended sorting) of the list of variables.

The variable name can be entered by typing directly into the cell of the *Name* column, using the listed acronyms. If the required acronym is not in the list it can be added using the buttons in the *Acronyms* tab (see the Working with EUROMOD – Administration of EUROMOD variables section of the EUROMOD Help, accessed from the *Help & Info* tab). The *Monetary* box is checked by default but can be unchecked. As discussed above, the *Automatic Label* is automatically generated from the acronyms used in the name of the variable, and cannot be edited. A Uganda specific description of the variable can be added by typing directly into the *Description* cell. Remember to save the file before closing.

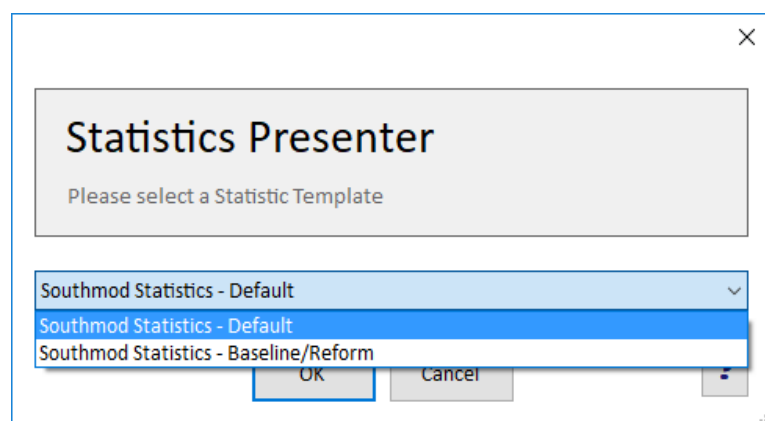
4.5 USING THE STATISTICS PRESENTER

The **STATISTICS PRESENTER** is an extremely powerful (and flexible) tool within UGAMOD. It allows immediate analysis of the output data. At the present time the **STATISTICS PRESENTER** calculates the costs of benefits and amount of taxes simulated. It produces estimates of both consumption poverty and inequality and income poverty and inequality taking into account simulated taxes and social transfers. It also allows comparisons between two systems.

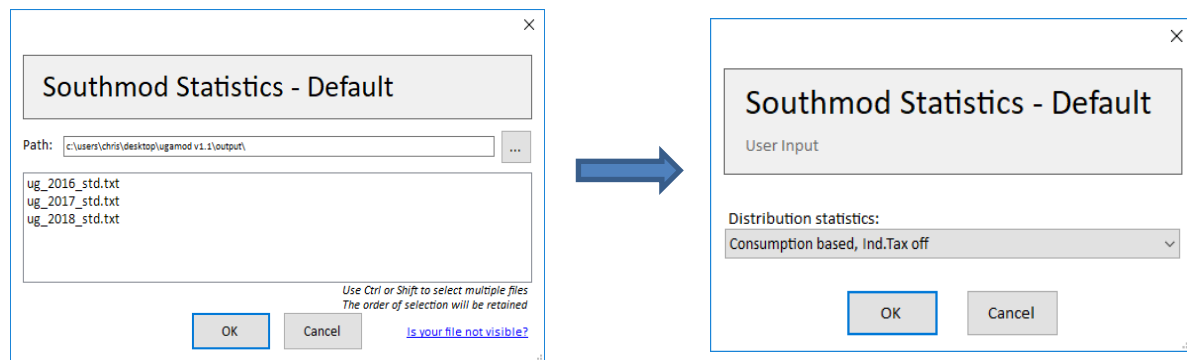
The **STATISTICS PRESENTER** is accessed from the Applications menu by clicking on the Statistics Presenter button displayed:



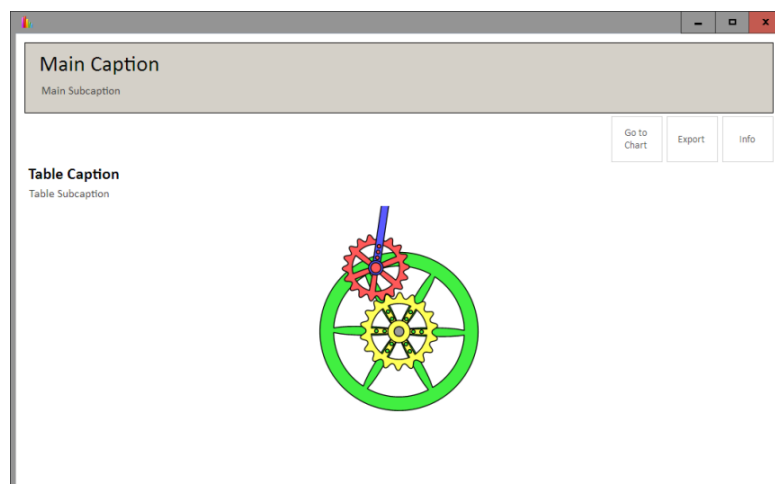
This then reveals the following dialogue:



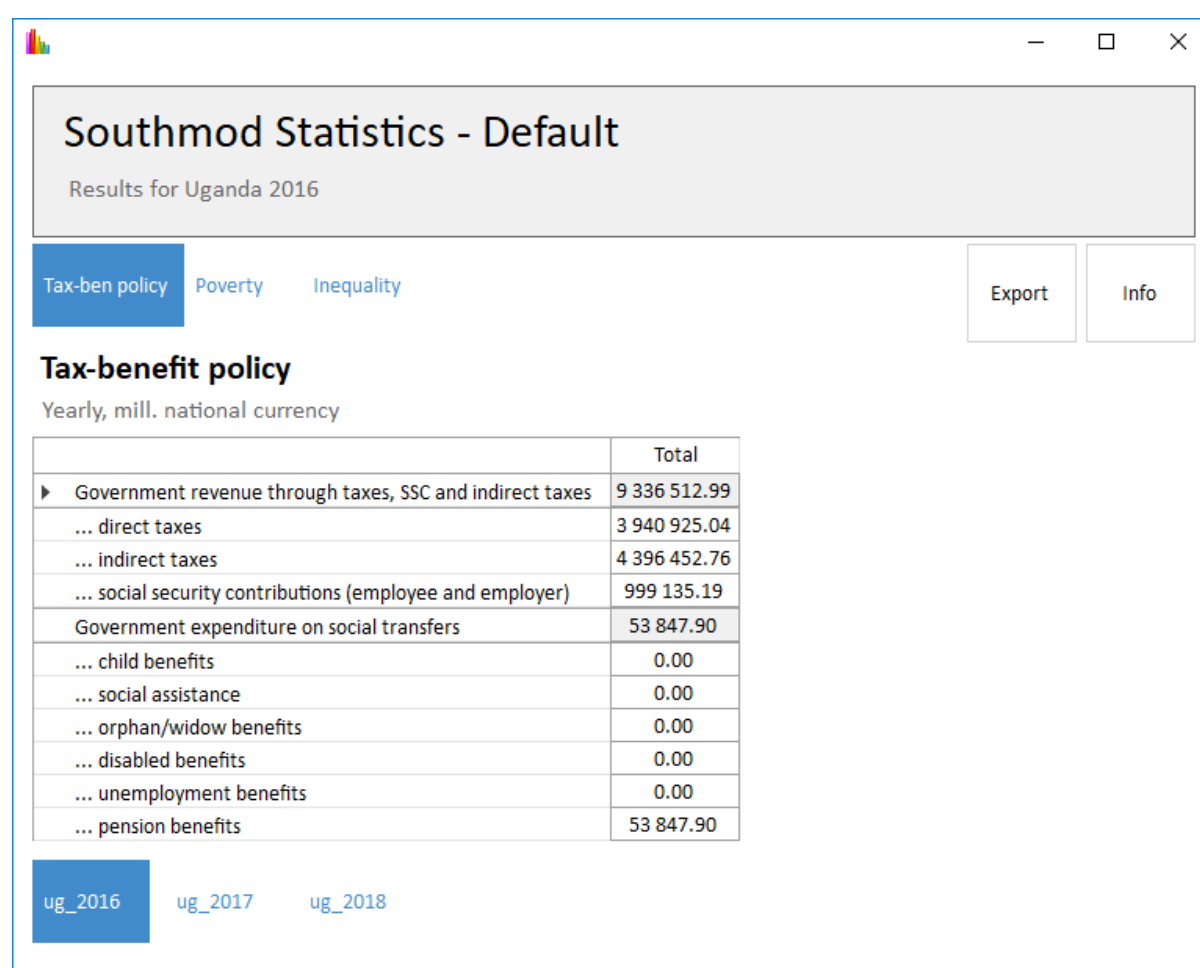
At this point the user is asked to choose between two templates: **SOUTHMOD STATISTICS** or **SOUTHMOD STATISTICS COMPARISON** (NB Other Templates may be added in due course). Choosing **SOUTHMOD STATISTICS** will reveal a list of possible output files to be analysed. One or more may be selected. To select more than one use the control key and mouse click to select the desired output files. Pressing OK will give another dialogue which enables the user to select whether poverty and inequality statistics are consumption based or income based.



After the user has selected either consumption or income-based poverty/inequality the **STATISTICS PRESENTER** will begin to process the results and will display the following gear wheel whilst it is doing so:



When the calculations are complete the following screen is showed first:

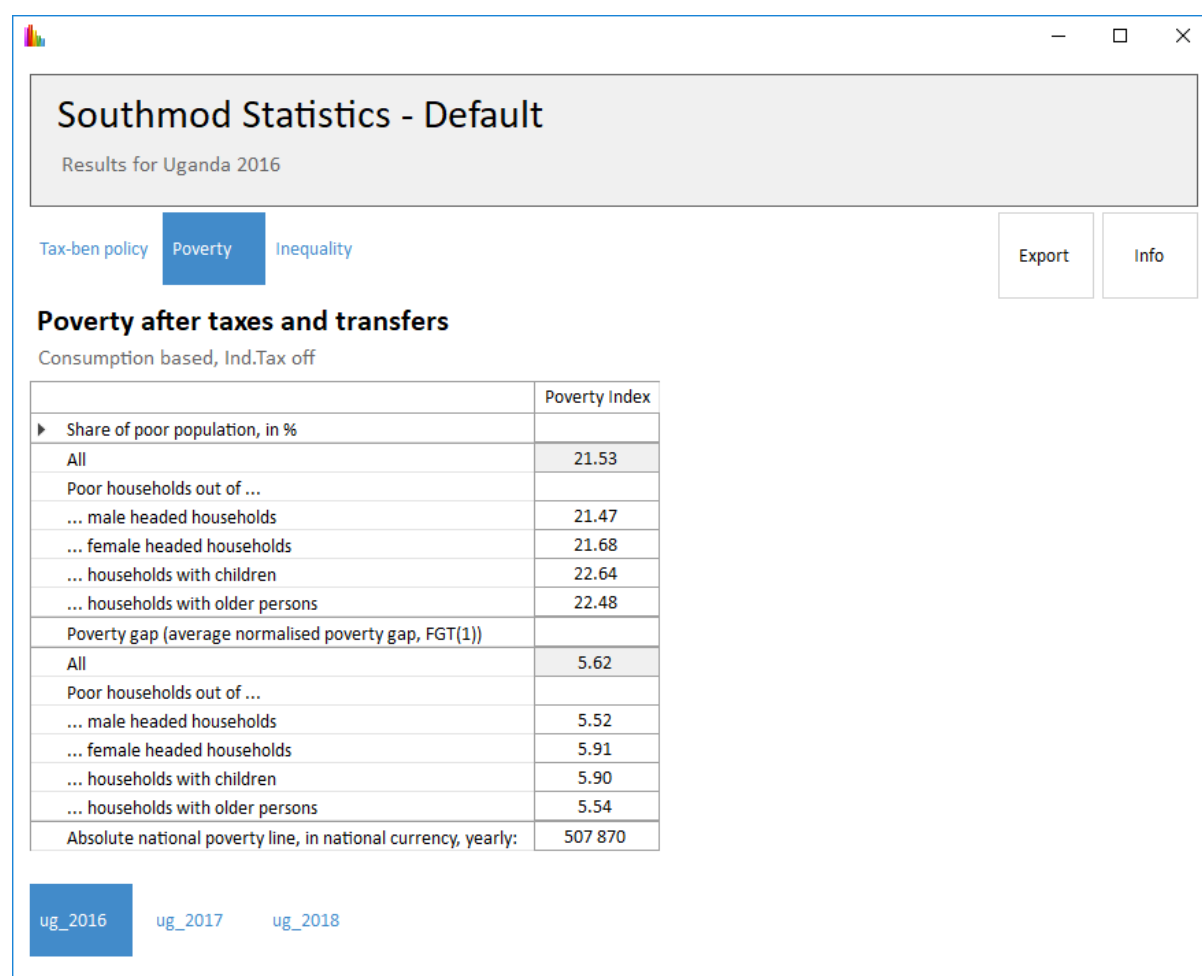


This is referred to as the [TAX-BEN POLICY PANEL](#) and presents information on modelled annual government revenue through taxes (direct and indirect) and social security contributions. It also displays annual government expenditure on various kinds of social transfer. What is displayed under different headings is controlled through [INCOMELISTS](#). These will be discussed later in this section.

The [POVERTY PANEL](#) (see next page) and the [INEQUALITY PANEL](#) are accessed using the tabs to the right of the [TAX-BEN POLICY](#) tab (see screenshots below). The [POVERTY PANEL](#) shows both the percentage of the population in poverty (also shown broken down by the following types of household: male headed households; female headed households; households with children and households with older persons). The [POVERTY PANEL](#) also displays the average normalised poverty gap FGT(1). The poverty line used in the [POVERTY PANEL](#) (reported in the last row of information) is specified in a special [POLICY](#) in the model and can be modified as needed. This poverty [POLICY](#) will be discussed later in this section.

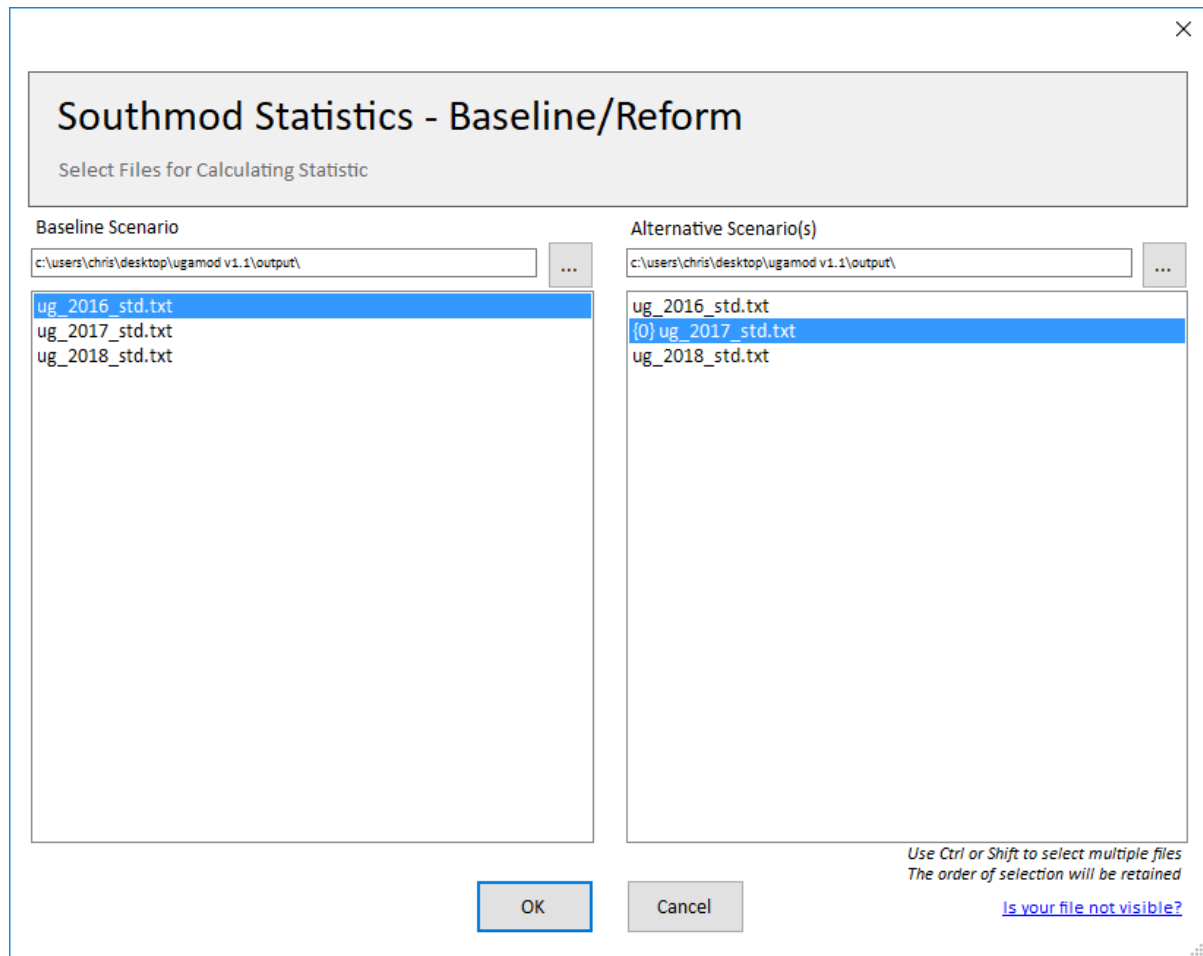
In addition to the GINI coefficient, the [INEQUALITY PANEL](#) also displays the P80/P20 ratio measure of inequality (the ratio of the income of those at the eightieth percentile of the distribution compared to the income of those at the twentieth percentile). The income at each quintile of the income distribution is also shown.

If the user had selected more than one output file then tabs at the bottom of the screen for each output file selected would be shown.



The initial dialog also allowed users to select [SOUTHMOD STATISTICS COMPARISON](#). If this option is selected then the user is presented with the dialogue shown below. In this example the base output file selected is *ug_2016_std.txt* (the selection is undertaken by clicking on the base scenario required) and the comparison output file selected is *ug_2017_std.txt* (also selected by clicking the output file required). After this dialogue the user is presented with the choice of whether to proceed with consumption based or income based estimates and the final output shows the difference between the base system and the comparator system.

This option is particularly useful when examining the impact of policy reforms.



The output for all three panels takes the form of a comparison between the base system (i.e. the output file selected in the leftmost part of the dialogue box shown above) and the comparator system.

Before turning to the model requirements to enable the **STATISTICS PRESENTER** to work correctly it should be noted that on each panel, there is a button to enable the user to Export the results. Various options are presented. Perhaps 'ALL' is most useful as it will export to Excel the tables on each of the panels.

Compulsory information required by the [STATISTICS PRESENTER](#)

There are two sets of information that the [STATISTICS PRESENTER](#) requires in order to function correctly. Certain information is generated when the baseline data is prepared and forms part of the data preparation process: this will not need to be altered by the user and is recorded here only for information purposes. The second set of information is contained in the model: some of these will require alteration in certain circumstances but again most will not require alteration.

Data requirements for [STATISTICS PRESENTER](#) - generated during the data preparation stage and not requiring alteration by the user

The following variables are generated during the data preparation step and **will not require alteration:**

Variable *xhh* - household consumption as used by the Uganda National Bureau of Statistics for calculating consumption poverty measures in the base year (2016).

Variable *ses* – Equivalence scale in adult equivalents used in Uganda for poverty and inequality measurement.

Variable *dhh* – set to 1 for household head, 0 for other members

Variables representing social benefits reported in data (where applicable).

Variables representing direct taxes reported in data (where applicable).

Variable *xivot* - value of home-grown produce where available to add to income to estimate income poverty. This will be set to 0 when not known or where it is desirable to estimate income property without taking into account home-grown produce.

Requirements for [STATISTICS PRESENTER](#) – to be included in the model

Apart from new variables generated within the data preparation stage, the [STATISTICS PRESENTER](#) requires certain information mainly in the form of [INCOMELISTS](#) to generate the requisite output. As with the variables generated during the data preparation stage many will never require amendment. However, some will require amendment particularly if a new policy is introduced. Where this is the case, this will be clearly indicated in the following sections. The sections indicate the information required for each of the three panels in the [STATISTICS PRESENTER](#) output and are grouped accordingly.

i) The [INCOMELISTS](#) required for the [TAX-BEN POLICY PANEL](#)

The first three [INCOMELISTS](#) represent the revenue from taxation reported in the first part of the table in the [TAX-BEN POLICY PANEL](#). They will only require amendment if new taxes or Social Security contributions are simulated.

ils_tax: contains simulated direct taxes (income tax and presumptive tax)

ils_taxind: contains simulated indirect taxes (such as VAT and excise taxes)

ils_sic: contains simulated social security contributions (employee and employer)

The second group of six [INCOMELISTS](#) contain the benefits simulated in the model. They are required for the second part of the table in the [TAX-BEN POLICY PANEL](#) - the costs of social benefits. **It is important to note that these six lists are mutually exclusive:** a simulated benefit can only occur

in one of the lists. So, for example, if a disabled child benefit were simulated, a decision would need to be made whether to insert this into the child benefit income list or the disability benefit income list. Amendments to these lists will need to be made any time that a reform scenario is introduced that introduces a new benefit, or when a new benefit is introduced by government.

ils_bch: contains all simulated child-related benefits

ils_bsa: contains all simulated social assistance-related benefits

ils_bsu: contains all simulated orphan and widowhood-related benefits

ils_bdi: contains all simulated disability-related benefits

ils_bun: contains all simulated unemployment-related benefits

ils_pen: contains all simulated pension benefits

ii) The [INCOMELISTS](#) and other requirements for the [POVERTY PANEL](#) and the [INEQUALITY PANEL](#)

There are a number of income lists required for the [POVERTY PANEL](#) and the [INEQUALITY PANELS](#) to work correctly.

For the consumption based measures the following [INCOMELISTS](#) are essential. **However, in general, they will not need to be altered.** Any alterations will have been made to component income lists e.g. if a new benefit is simulated it will have already been added to *ils_bensim* just as any new tax simulated will have been inserted within *ils_taxsim* (see section 4.3 above). This is good practice and should have been carried out whether or not the [STATISTICS PRESENTER](#) is used.

The [INCOMELISTS](#) for consumption poverty are as follows:

ils_tistn -this is composed of the following items recorded or imputed in the underpinning data: employee social security, turnover tax and any other direct tax.

ils_bendata – these are social benefits as reported in data.

The key Income list supporting consumption-based poverty is

ils_xhh_s – this comprises the following components which are either added (+) or deducted(-): *xhh* (+), *ils_bensim* (+), *ils_sicee* (-), *ils_tistn* (+), *ils_taxsim*(-) and *ils_bendata* (-)

or, if indirect taxes are selected as ‘on’ i.e. measures are ‘post fiscal income’

ils_xhh02_s – this comprises the following components which are either added (+) or deducted(-): *xhh* (+), *ils_bensim* (+), *ils_sicee* (-), *ils_tistn* (+), *ils_taxsim*(-) and *ils_bendata* (-) and *ils_taxind* (-)

There are two [INCOMELISTS](#) used for income based measures of poverty and inequality. The first is

ils_dispy2. It is comprised of the standard [INCOMELIST](#) of *ils_dispy* (see Section 3.2 above) together with the variable *xivot* (which may or may not contain any value – see above).

For calculation of distributional measures using post fiscal income there is an additional **INCOMELIST** *lls_dispy3*. It is comprised of the standard **INCOMELIST** of *lls_dispy2* minus indirect taxes (*ils_taxind*)

In addition to the income lists there is a **POLICY** – *poverty_lines_ug* which specifies the poverty line to be used. The **POLICY** is a simple one. For each poverty line there are two *Arithop* **FUNCTIONS**. The first **FUNCTION** specifies the basic needs poverty line while the other **FUNCTION** specifies the basic needs poverty line after taking indirect taxes into account. This is used to measure ‘post fiscal income’ poverty. The **FUNCTIONS** are very straightforward. The key **PARAMETER** is the formula which takes the constant for the poverty line (the regular poverty line or the one taking indirect taxes into account) and multiplies it by the overall CPI factor *\$f_CPI_Overall*. The output variables are *spl* and *spl02* and these are fed into the **STATISTICS PRESENTER**. The following screenshot illustrates the basic needs poverty line functions:

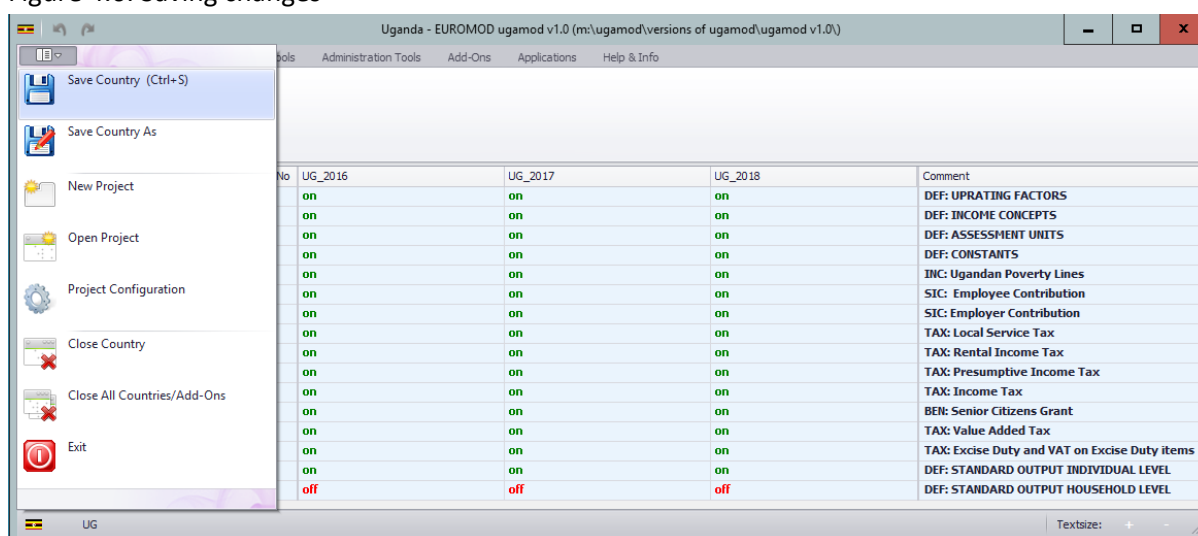
	Policy	Grp/No	UG_2018	Comment
5	poverty_lines_ug		on	INC: Ugandan Poverty Lines
5.1	DefVar		on	
5.1.1	spl	1	0	
5.1.2	spl02	2	0	
5.2	ArithOp		on	
5.2.1	Formula		$\$basic_needs_pov_line * \$f_CPI_Overall$	Basic Needs Poverty Line
5.2.2	Output_Var		spl	
5.2.3	TAX_UNIT		tu_individual_ug	
5.3	ArithOp		on	
5.3.1	Formula		$\$indirect_basic_povline * \$f_CPI_Overall$	Basic Needs Poverty Line for post fiscal income
5.3.2	Output_Var		spl02	
5.3.3	TAX_UNIT		tu_individual_ug	

4.6 OTHER TASKS

Saving changes

To save your changes open the main menu (above the *Run EUROMOD/UGAMOD* button) and select the menu item *Save Country* (see Figure 4.6). Alternatively press *Ctrl-S*.

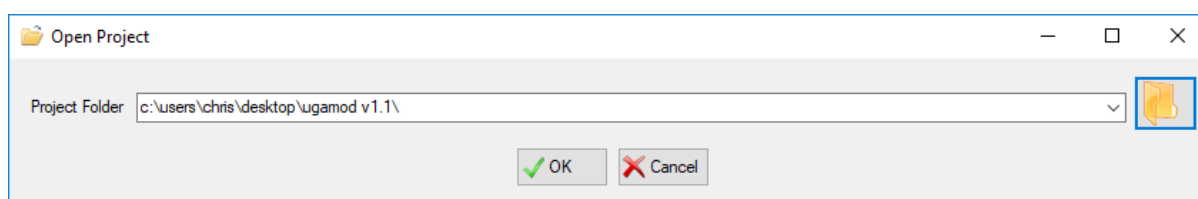
Figure 4.6: Saving changes



Opening a new project

The main menu also has an option to *Open Project*. This opens a dialog which allows the content displayed by the user interface to be changed, as well as the default input and output paths (see Figure 4.7). This may be required if, for example, you wish to create a version of UGAMOD for testing purposes, or a new version of UGAMOD is issued and you wish to switch to using the new version.

Figure 4.7: The *Open Project* dialog



In the dialog box you are asked to provide the locations of the *EUROMOD* (i.e. *UGAMOD*) Folder (e.g. *C:\UGAMOD V1.1*).

References

Byaruhanga, C., T. Kaidu, S. Kavuma, F. Masekesa, N. Musoke, D. Nakyambadde, M. Nalukwago Isingoma, M. Noble, J. Okello Ayo, and R. Waiswa (2019). 'SOUTHMOD Country Report Uganda-UGAMOD v1.1'. Helsinki: UNU-WIDER.

Mitton, L., Sutherland, H. and Weeks, M. (Eds.) (2000) *Microsimulation Modelling for Policy Analysis*, Cambridge: University Press.

Sutherland, H. and F. Figari (2013) 'EUROMOD: the European Union tax-benefit microsimulation model', *International Journal of Microsimulation* 6(1) 4-26.

Zaidi, A. Harding, A. and Williamson, P. (Eds.) (2009) *New Frontiers in Microsimulation Modelling*, Vienna: Ashgate.