The Bayesian Estimation, Analysis and Regression (BEAR) Toolbox

User guide

Version 3.0

Alistair Dieppe
Romain Legrand
Björn van Roye

Contents

CHAPTER 1

Introduction

The new Bayesian Estimation, Analysis and Regression (BEAR) toolbox has been developed for forecasting and policy analysis. There has been an increasing use of Bayesian VARs to address a wide range of issues in macroeconomics, including forecasting, scenario analysis as well as exploring the contribution of different factors (e.g. via historical decompositions) to the evolution of macroeconomic variables. Bayesian estimation is particularly useful in cases where there is limited data availability. It is also one approach to address the issue of large dimensionality in estimation and has the advantage of providing an empirical measure of uncertainty surrounding the estimates. In many ways Bayesian VARs are preferable to traditionally estimated VARs.
Some codes and software for Bayesian VARs already exist. However, they have very limited features. Furthermore, they tend to lack flexibility, are difficult to augment, and are typically not very user-friendly. Accordingly, it has been decided to create a more comprehensive Bayesian VAR toolbox where the programme is easy for non-technical users to understand, augment and adapt.
In particular the BEAR toolbox includes a user-friendly graphical interface which allows the tool to be used by country desk economists. Furthermore, the toolbox is well documented, both within the code as well as including a detailed theoretical and user's guide.
By making available the toolbox we aim at sharing expertise and hope this could become a key tool for macroeconomic analysis which exploits synergies and increases efficiency as well as avoids unnecessary duplication of work.

1.1. BEAR-Toolbox - available features and applications

We next list an overview of the applications available in BEAR. BEAR Version 3.0 offers the following applications:

- Estimation techniques of VAR models
  - OLS (maximum likelihood) VAR
  - Standard Bayesian VAR (**?** and Litterman (1986))
  - Mean-adjusted BVAR with informative prior on the steady-state (Villani (2009))
  - Bayesian Panel VAR (as in Canova and Ciccarelli (2013))
- Alternative priors for Bayesian VAR models
  - Minnesota (Litterman (1986))
  - Normal Wishart (**?**)

  - Independent Normal Wishart with Gibbs sampling
  - Normal diffuse (**?**)
  - Dummy observations (**?**)
- Prior extensions for Bayesian VARs
  - Hyperparameter optimisation by grid search (similar to **?**)
  - Block exogeneity
  - Dummy observation extentions: sum-of-coefficient, dummy initial obervation (**?**)
- Panel models
  - OLS Mean-group estimator (Pesaran and Smith (1995))
  - Bayesian pooled estimator
  - Random effect model, Zellner-Hong (Zellner and Hong (1989))
  - Random effect model, hierarchical (Jarocinski (2010))
  - Static factor model (Canova and Ciccarelli (2013))
  - Dynamic factor model (Canova and Ciccarelli (2013))
- Structural VARs
  - Choleski factorisation
  - Triangular factorisation
  - Sign, magnitude and zero restrictions (Arias et al. (2014))
- Applications
  - Unconditional forecasts
  - Impulse response functions
  - Forecast error variance decomposition
  - Historical decompositions
  - Conditional forecasts: shock approach (Waggoner and Zha (1999))
  - Conditional forecasts: tilting approach (Robertson et al. (2005))
  - Forecast evaluation: standard and Bayesian-specific criteria

1.2. What's new in BEAR? BEAR 3.0 vs. 2.3

Some of you may have already used the BEAR toolbox in its previous version, which was version 2.3. While version 3.0 does not involve any fundamental change with respect to version 2.3, some modifications have been applied to account for the feedback we have received and our own user experience. These modifications pertain to both BEAR environment and BEAR applications.

- BEAR environment

1. Interface
   The graphical interface of BEAR has been simplified. It is now faster and more consistent between models. There is also a "Back" button which avoids to abort the current run in order to correct previous information.
2. Tables
   All the tables that used to be filled by the way of an interface have been switched to Excel. This change was motivated by two main reasons. First, in the case of large models, the tables could exceed the size of the screen, making it impossible to fill them. Secondly, the Java table used by Matlab proved quite unstable: entries filled by the user would sometimes be ignored by the code without a reason. On these grounds, it seemed preferable to use Excel. As an additional bonus, the new Excel configuration renders the use of the toolbox more flexible: it is now possible to modify some elements of the model without having to update the table.
3. Developer's version
   It is now possible to run BEAR without having to use the graphical interface. The information has then to be manually entered into a specific file which then replaces the interface. This option is useful for advanced users who may wish to make a more customised use of the toolbox.

- BEAR applications
  1. Sign restrictions
     The sign restriction setup now allows to apply zero restrictions at any period, and not only at impact, as it used to be. Different restrictions can also now apply to different periods.
  2. Dummy observation extensions
     The sums-of-coefficient and dummy initial extensions now act as two completely separated applications. In particular, they are now attributed separate hyperparameter values for shrinkage: λ_6 for the sum of coefficient extension, and λ_7 for the dummy initial observation extension. The grid optimises has also been updated to optimize the two hyperparameters independently.

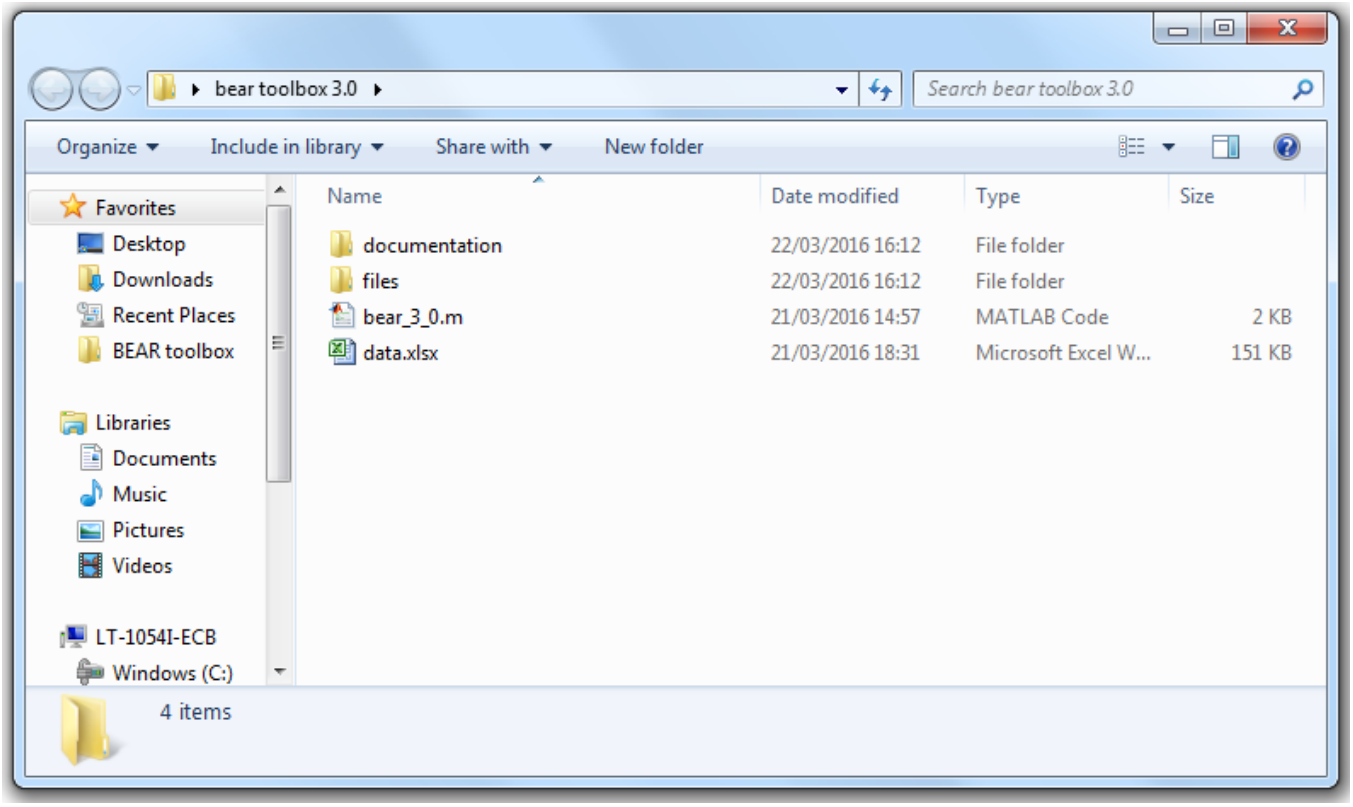
CHAPTER 2

Preparing your project

2.1. Preparing your folder

The BEAR toolbox package is delivered to you as a zip file. Your unzipped folder should look like this:

Figure 2.1.: *Unzipped folder*

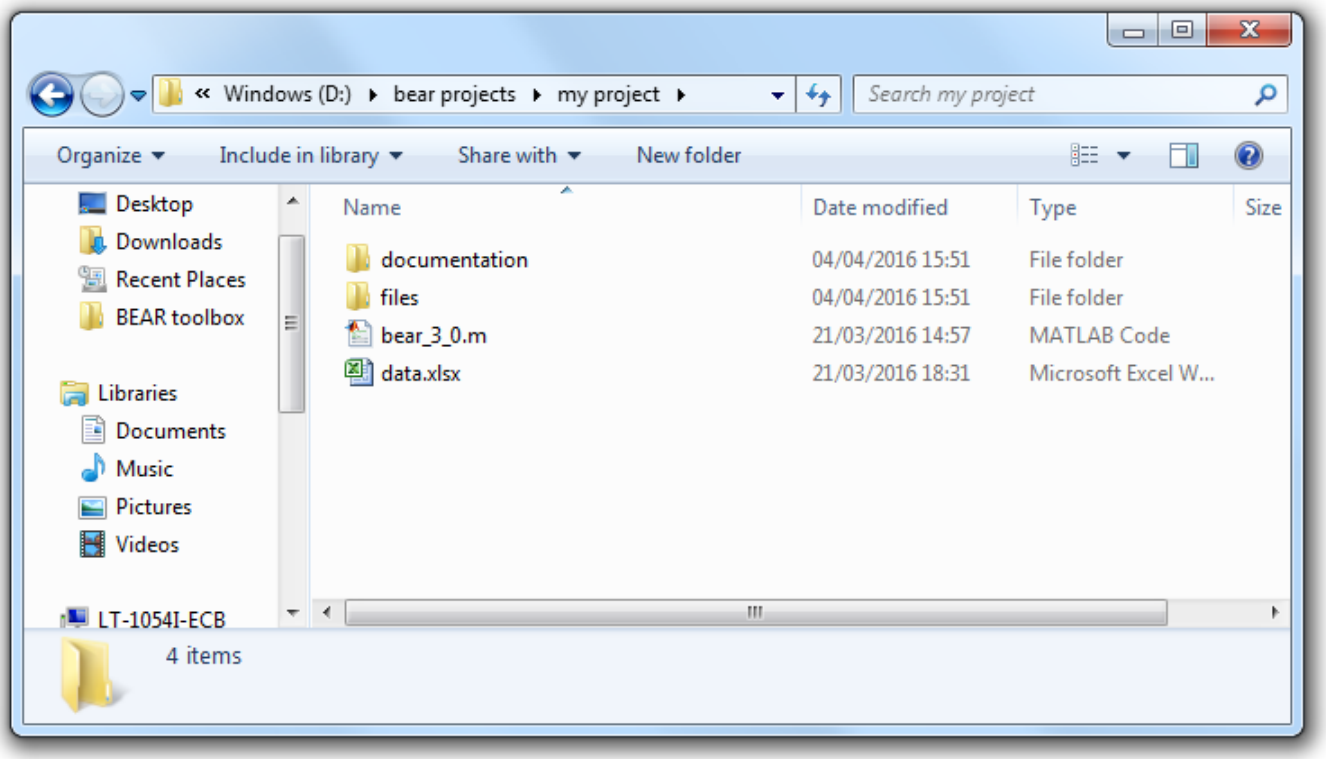


The folder contains four elements:
- the "documentation" folder. This folder contains two documentation files in pdf format: the user guide (the document you are currently reading), explaining how to use the toolbox; and a technical guide providing the mathematical derivations for the models and applications implemented by the toolbox.
- the "files" folder. This folder contains the set of matlab functions run by the code. Except in a

few specific cases indicated in this guide, you should not alter nor delete any elements in this folder. Doing so may result in the code not working properly anymore.
- the "bear_3_0.m" matlab file. This file is used to start the toolbox and run the estimation. Further details can be found in chapter 3: Running your project.
- the "data.xlsx" Excel file. This file contains the data that will be used by the toolbox to run the estimation.

By default, the toolbox folder is called "bear toolbox 3.0". However, it is possible to rename the folder to suit your particular project, for instance "project Japan", or "BVAR US". You may then place the folder in any directory of your choice. For example, you may create a new project, and for this project change the folder name from "bear toolbox 3.0" to "my project". Assume that for convenience you also create a new folder on your D drive called "bear projects" that gathers all your different projects. You may then move "my project" into "bear projects". Your project folder will then look as follows:

Figure 2.2.: *Project folder*



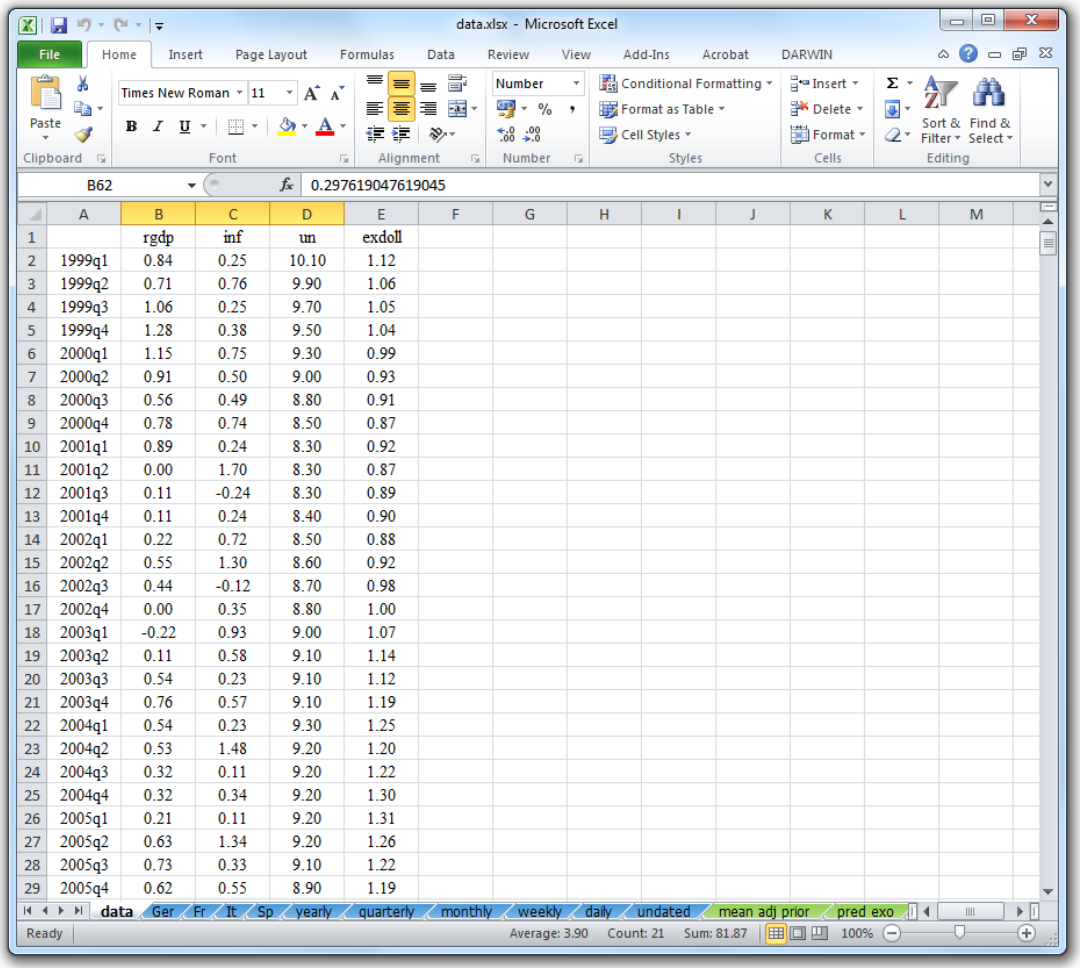
Note that it is possible to use a single folder (here for instance, "my project") for all your different projects. In practice however this is not recommended. Different projects imply different datasets, and you may not want to overwrite the data from former projects. Also, the results from your former projects saved by the toolbox will be overwritten if you decide to run a new project. For these reasons, it is preferable to create a new (separate) folder for each of your projects, which is usually not a problem since the unzipped file only takes 6 megabytes of space (and less than one megabyte if

you delete the pdf guides). Gathering all your projects in a single directory like in the above example with the "bear projects" folder can then be a convenient option for organisation purposes.

2.2. Opening your data sheet

Now that your project folder is ready, you can prepare the dataset for you project. This is done by the way of the Excel file called "data.xlsx". When you open the file, it should look like this:

Figure 2.3.: *Excel data spreadsheet*



The Excel spreadsheet contains many different tabs, distributed into three colours:

- the blue tabs are related to data. They are the tabs that will be of interest for the creation of the datasets.
- the green tabs are related to applications with the Bayesian VAR and mean-adjusted VAR models. They can be ignored for now.
- the orange tabs are related to applications with panel Bayesian VAR models. They can also be ignored for now.

2.3. Preparing your data sheet for a standard OLS VAR, Bayesian VAR or mean-adjusted BVAR model

If you intend to run any of these three models, then the only sheet you have to care about is the sheet called "data". You should not rename the sheet as this would cause the code to return an error. As you may notice, the sheet is not blank: it is filled it with an example in order to make your work simpler. The easiest and safest way to create the data for your own project set is probably to use this example spreadsheet and adapt it to your own dataset. The default example dataset is quarterly, but BEAR accepts six different date formats (yearly, quarterly, monthly, weekly, daily, and undated). Therefore, the file contains 6 different worksheets respectively called "yearly", "quarterly", "monthly", "weekly", "daily", and "undated" that all contain example datasets for their corresponding formats (see Appendix A for more details on these datasets). Given the date format of your project, you may thus copy the content of the corresponding sheet and paste it into "data", then adapt it to your own dataset. Note also that you may just delete these example worksheets if you do not need them.

Considering the example sheet, the organisation is fairly straightforward: the first column is used for the date labels, while the first row is used for the labels of your data series. The default example for instance considers a dataset with four variables for the euro area: real GDP, inflation, unemployment, and the euro-dollar exchange rate. The dataset is quarterly, starts in 1999q1 and ends in 2015q3. More details are now provided about the different elements.

Variable names

The first row of the spreadsheet is always used for the labels of the data series. The list must start in cell (1:B), as in the example, and then develop rightward. The series names can just be any names, but a name cannot comprise spaces. For instance, "exdoll" or "ex_doll" would be correct, but "ex doll" would result in a crash of the code. Also, it is recommended you use names containing less than 20 characters, as otherwise you may face minor display issues of the results.

Dates

The first column of the spreadsheet is always used for the labels of the dates. The list must start in cell (2:A) and then develop downward. As already stated, BEAR accepts six formats of dates: yearly, quarterly, monthly, weekly, daily, and undated data. Each format has its own specification. The specification of each format must be respected exactly, otherwise the code will return an error. Do not use either the automatic date formatting provided by excel. This format will not be recognised by the code and will result in an error. The six example sheets are there to help you building your own set. The different formats have to be specified as follows:

- yearly: dates have to specified as the year (4-digit number) followed by a lower "y", without a space.
correct example: 1999y
wrong examples: 1999, y1999, 1999Y, 1999 y.

- quarterly: dates have to specified as the year (4-digit number) followed by a lower "q", then by the quarter (number between 1 and 4), without spaces.
correct example: 1999q2
wrong examples: 2q1999, 1999Q2, 1999 q 2.
The dataset does not have to start at quarter 1. For instance, starting the set at 1999q2 is perfectly possible. Similarly, the dataset does not have to end at quarter 4. However, the labelling must be consistent between the start and end dates and there should be no missing quarter.

- monthly: dates have to specified as the year (4-digit number) followed by a lower "m", then by the month (number between 1 and 12), without spaces.
correct example: 1999m6
wrong examples: 6m1999, 1999M6, 1999 m 6.
The dataset does not have to start at month one, nor to end at month 12. The dates have to be consistent between the start and end dates, there should be no missing month and no disordering, and any full year should be made of 12 months.

- weekly: dates have to specified as the year (4-digit number) followed by a lower "w", then by the week, without spaces.
correct example: 1999w12
wrong examples: 12w1999, 1999W12, 1999 w 12.
The code is flexible: different years may include a different number of weeks. For instance, the first year in the set may end at week 52, the second at week 51, and the third at week 53. Any number of weeks per year is possible: the code will simply identify the numbering which has been used and adapt to it. Similarly a year does not have to start at week 1, whether it is the first or last data year, or any year in between.

- daily: dates have to specified as the year (4-digit number) followed by a lower "d", then by the day, without spaces.
correct example: 1999d208
wrong examples: 208d1999, 1999D208, 1999 d 208.
Similarly to weeks, different years may include a different number of days. Indeed, in the weekly example file, missing data led to suppress many entries, resulting in 2012 to comprise 261 days, but 2014 to include only 249 of them. This is not an issue, the code will identify the number of days provided for each year and adapt to it. Any year in the dataset does not have either to start at day 1 and may start at any day value.

- undated: dates have to specified as the period, followed by a lower "u", without a space.
correct example: 53u
wrong examples: 53, u53, 53U, 1999 y.
The dataset does not have to start at period 1 and may just start at any period value. Periods however have to be consecutive, and skipping period values is not allowed. For instance, jumping from period 56 to period 58, ignoring period 57, is not possible.

Note that in general the dates don't have to correspond to the dates of the model to be estimated. That is, the window of the data in the data sheet can be larger than the dates of the model. For

instance, the example sheet starts date in 1999q1 and ends it in 2015q3. It is perfectly possible to use this data sheet to estimate a model that will start in 2000q4 and end in 2014q2.

Data

The remaining entries of the "data" worksheet are used for the data itself. The set has to be balanced and consistent with both the variable and date labels. That is, the dataset must be a rectangular array, with exactly as many columns as the number of variables, and exactly as many rows as the number of periods, similarly to the example datasets. Also, note that non-numerical or blank values are not permitted, as the code needs to generate a complete numerical array from the spreadsheet. Values identified as blanks or non-numerical will generate an error and prevent the code from running correctly.

Similarly to dates, the number of data series can be larger than the number of variables effectively used in the model. The example worksheet contains for instance the series rgdp, inf, un and exdoll, but it is perfectly possible with this sheet to estimate a model which would only involve rgdp and un.

BEAR allows you to include both endogenous and exogenous variables in your model. Exogenous variables can be constant terms, time trends, dummy variables, but also actual economic variables such as oil price. It does not matter whether a variable in the spreadsheet will be used as endogenous or exogenous by the model, the spreadsheet only uses a unique format. Indeed, if you look at the example sheet for instance, the variable exdoll is just entered as any other variable while it will be typically used as an exogenous variable. Note finally that if you plan to use a constant term, you do not need to create a series of ones in the spreadsheet: the constant is a special term that the code will handle internally.

2.4. Preparing your data sheet for a panel VAR

The preparation of your data for a panel VAR model is overall similar to that of a normal VAR model, except that a bit more planning is required. As a panel involves different units, you need first to create and name a new worksheet for each unit. In the example file, for instance, four worksheets were created called Ger, Fr, It, Sp, respectively corresponding to Germany, France, Italy and Spain. Those names will be the ones used by the code for the estimation of the model, so that rules similar to variable names apply: any name is possible, but unit names cannot contain spaces.

The variable, dates and data formats for each individual unit are rigorously similar to that of the standard VAR, and you are referred to section 2.3 for further details. Note that your dataset must be consistent across units: it must include the same variables over the same periods. The main difference with a standard VAR model comes from the treatment of exogenous variables. While in a standard VAR, endogenous and exogenous were treated indifferently, a panel VAR differentiates the two types of variables: endogenous variables share the same names across units but their values are unit-specific, while on the other hand exogenous variables have to be common to all units, that is, their values are similar across units. Hence, unlike a standard VAR model, for a panel VAR you

need to plan in advance which variables will be endogenous (variable values will differ from one unit to another) and which variable will be exogenous (identical values for all units). In the example worksheet, for instance, each unit comprises four variables: real GDP, inflation, the unemployment rate, and the exchange rate between the euro and the US dollar. The first three variables are proper to their respective countries and will be used as endogenous variables: their values are allowed to be unit-specific. The exchange rate with the dollar on the other hand is common to all countries: it will be used as an exogenous variable, and the values are the same for all units.

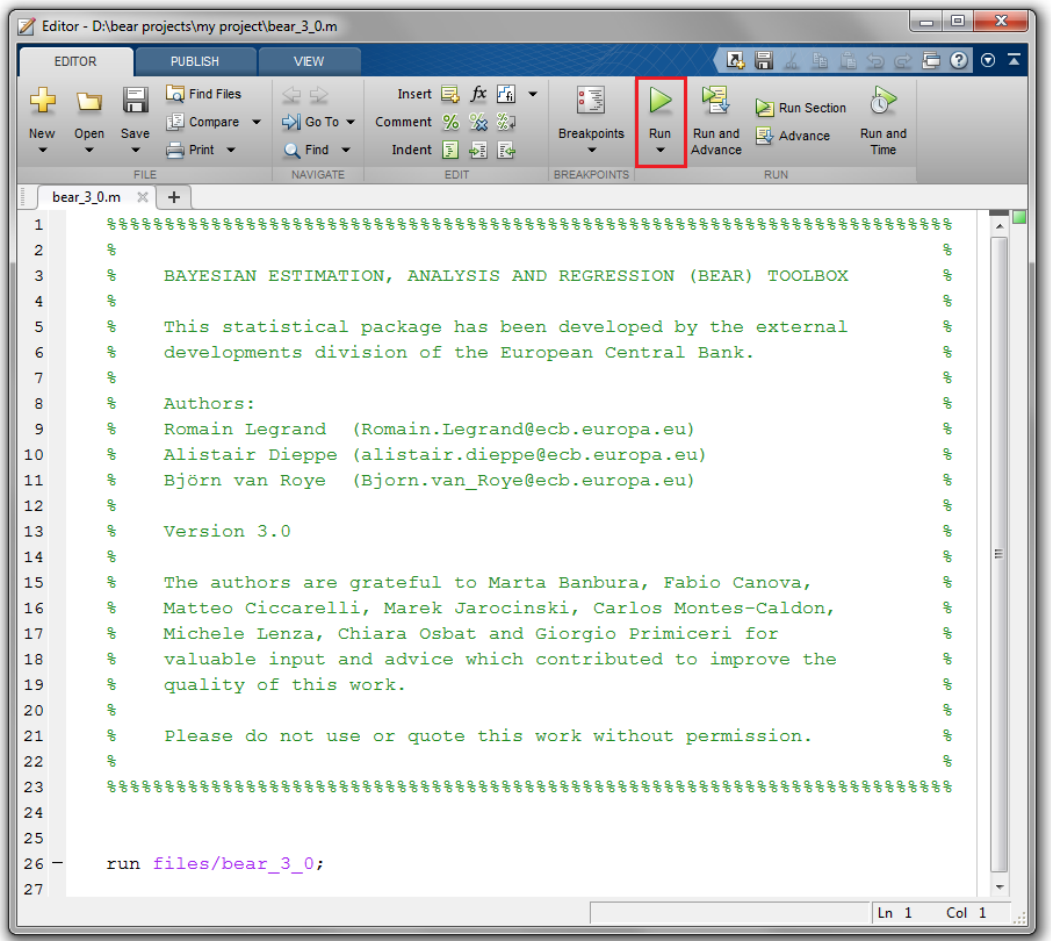
CHAPTER 3

Running your project

3.1. Launching the toolbox

Now that your dataset is ready, it is possible to run your project. To start the BEAR toolbox, double-click on the file "bear_3_0.m" on your project folder. This will initiate a Matlab session, and open the following window:

Figure 3.1.: *Matlab window*

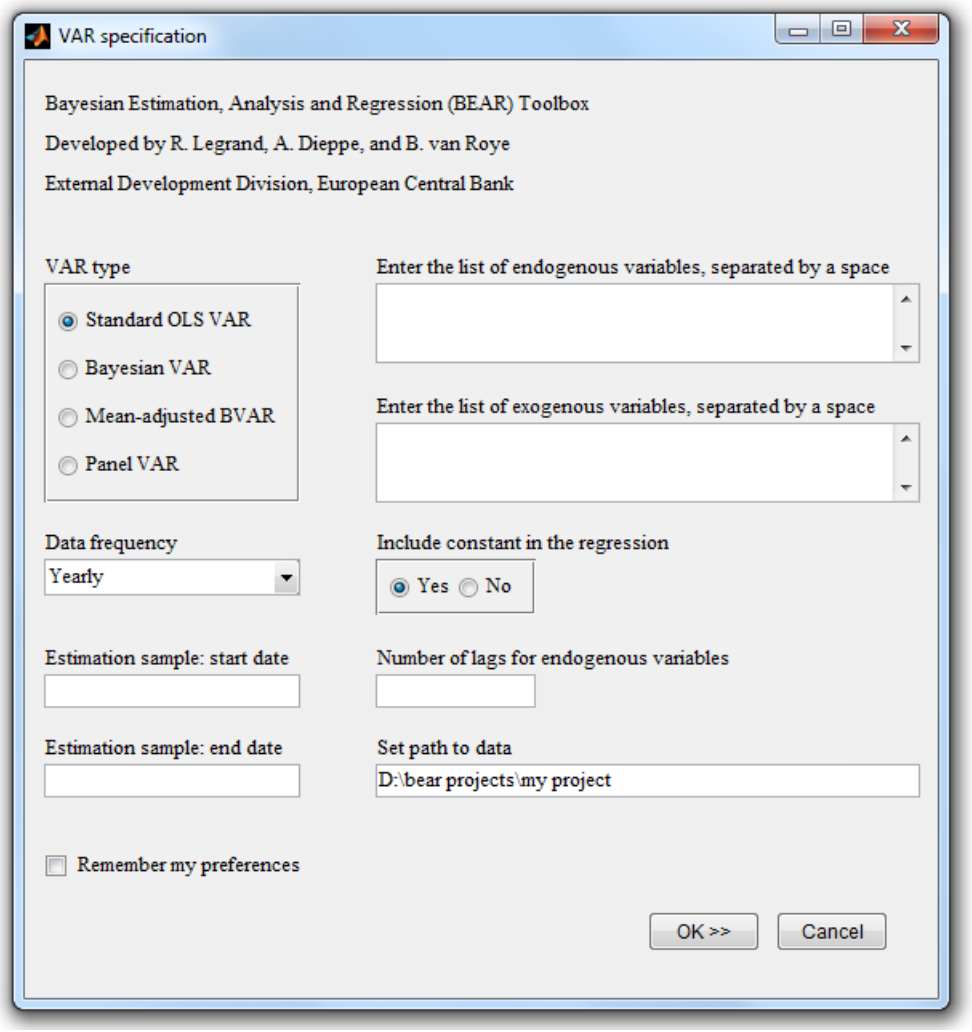


All what you need to do is to press the "Run" button (circled in red) in order to start the toolbox.

3.2. Setting basic information for your model

Pressing the "Run" button will open the first interface, which looks as follows:

Figure 3.2.: *Initial interface*



The different entries are now detailed, in turn.

VAR type The BEAR toolbox proposes four different estimation techniques of VAR models. the standard OLS VAR, the Bayesian VAR which constitutes the core of the toolbox, the mean-adjusted model developed by Matthias Villani, and the Bayesian panel VAR. For more details about these different models, please refer to the technical guide.

Data frequency Simply select the choice corresponding to your dataset in this menu. Selecting a type inconsistent with your data will result in an error from the code.

Estimation sample: start date Type the start date of your estimation sample. This date does

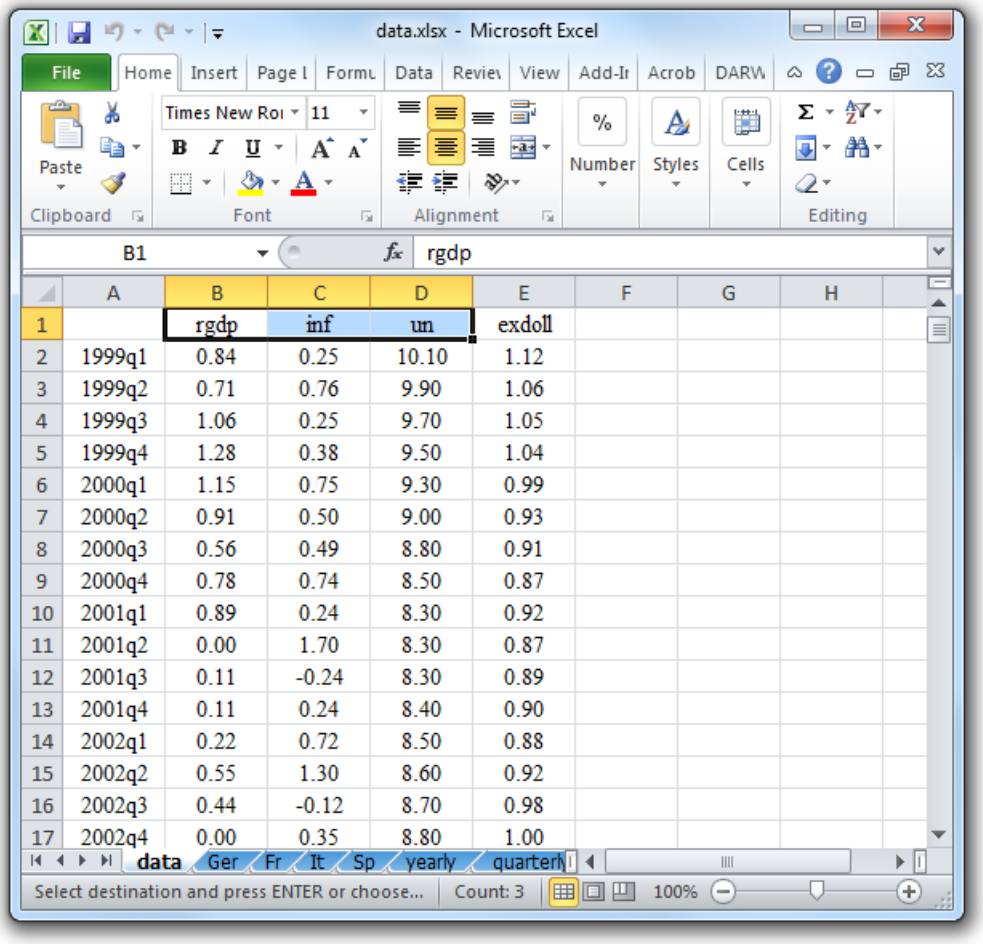
not have to be the first date of your dataset, but has of course to be included in your dataset.

Estimation sample: end date Similar to the sample start date: the date does not have to be the final date of your dataset, but has to be included in the dataset.

List of endogenous variables Type in this box the names of the variables you want to include as endogenous in your model, separating them with a space. You may type only a subset of variables from your Excel spreadsheet, and the order in which you enter them into your model does not have to be that of your spreadsheet. Note however that names are case-sensitive, and that any case or spelling mistake will result in a programme error.

tip: if you want to save time, you can directly copy the cells of names from your Excel spreadsheet and paste them without further ado in the window of the interface. While the spaces will look odd, the code has an automatic correction function that will replace them with regular spaces once you validate the interface. For instance, if with the example file you want to estimate a model with "rgdp", "inf" and "un", you can just copy the cells as shown:

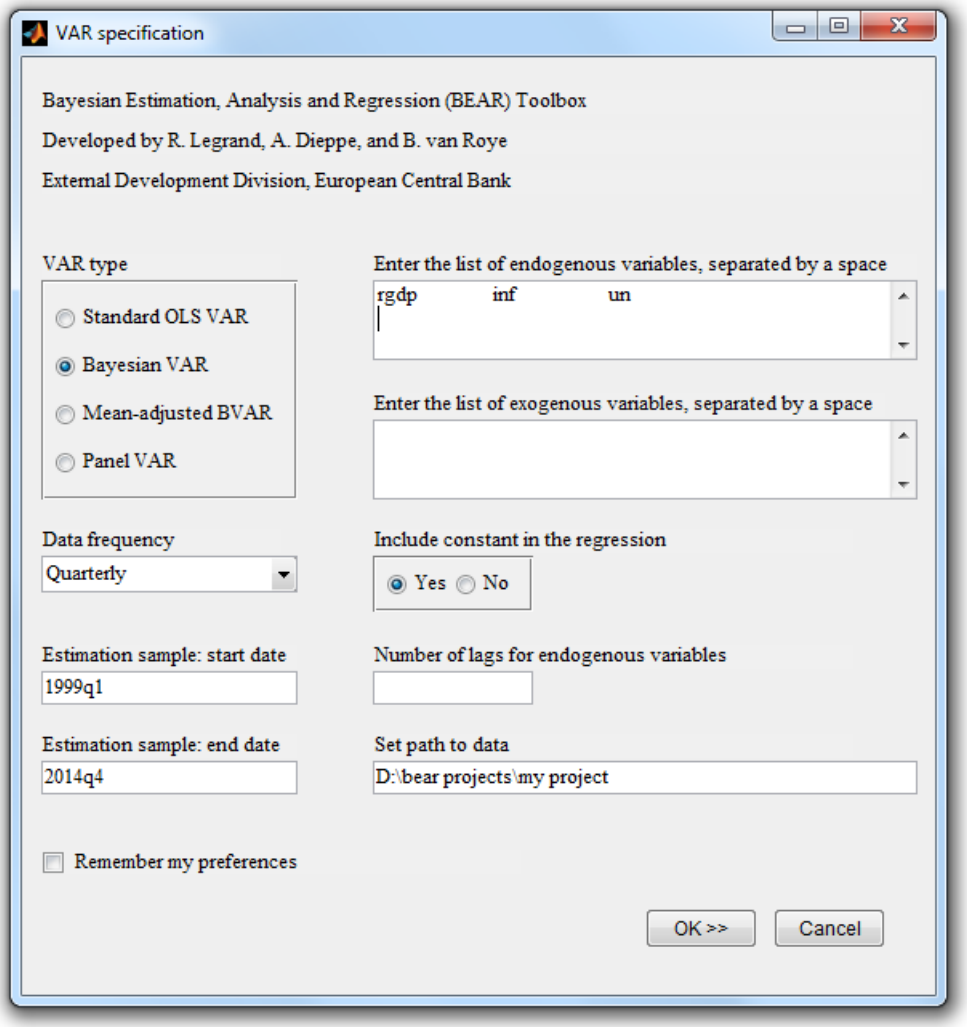
Figure 3.3.: *Copying cells from Excel*



	A	B	C	D	E
1		rgdp	inf	un	exdoll
2	1999q1	0.84	0.25	10.10	1.12
3	1999q2	0.71	0.76	9.90	1.06
4	1999q3	1.06	0.25	9.70	1.05
5	1999q4	1.28	0.38	9.50	1.04
6	2000q1	1.15	0.75	9.30	0.99
7	2000q2	0.91	0.50	9.00	0.93
8	2000q3	0.56	0.49	8.80	0.91
9	2000q4	0.78	0.74	8.50	0.87
10	2001q1	0.89	0.24	8.30	0.92
11	2001q2	0.00	1.70	8.30	0.87
12	2001q3	0.11	-0.24	8.30	0.89
13	2001q4	0.11	0.24	8.40	0.90
14	2002q1	0.22	0.72	8.50	0.88
15	2002q2	0.55	1.30	8.60	0.92
16	2002q3	0.44	-0.12	8.70	0.98
17	2002q4	0.00	0.35	8.80	1.00

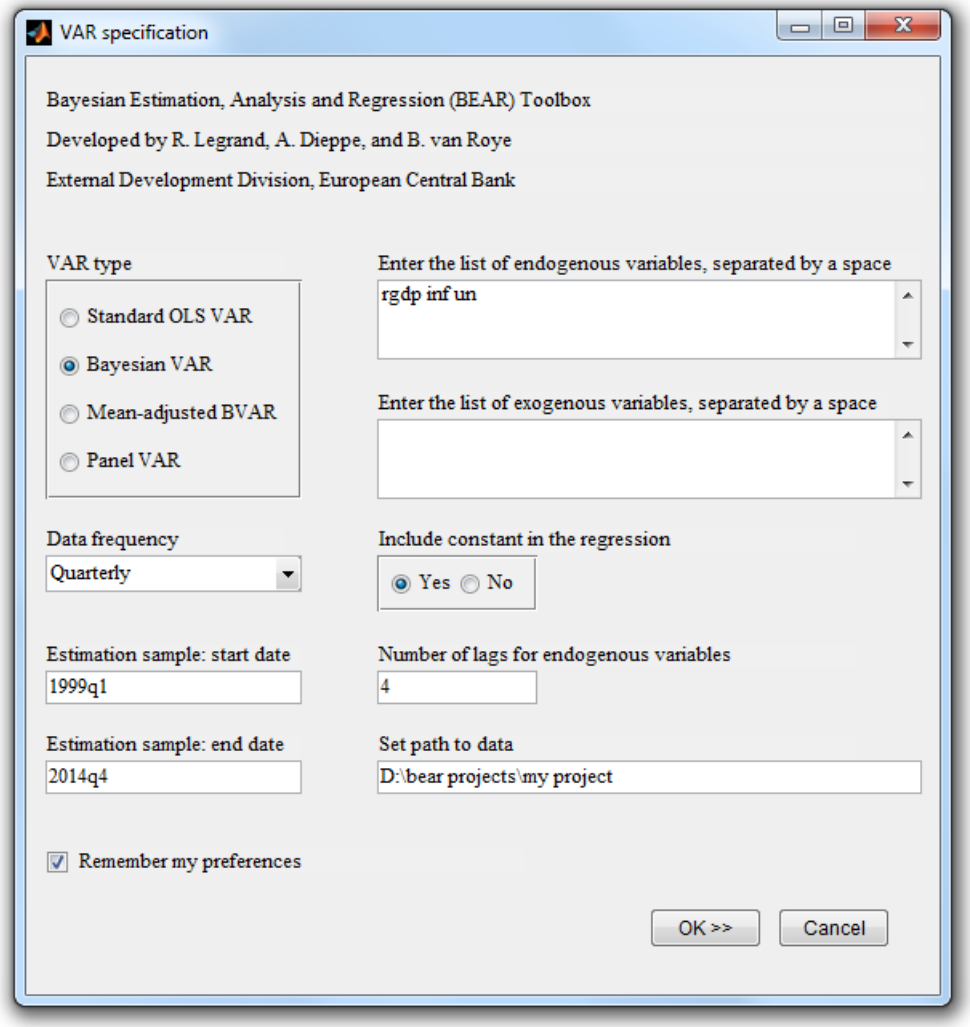
Then paste the cells into the interface. It will look as follows:

Figure 3.4.: *Pasting into the interface*



The spaces look odd, but you do not need to change anything. If you validate the interface, and then come back to it (for instance pressing the back button), it will then look like this:

Figure 3.5.: *Automatic space correction*



As you can see, the spaces have been corrected automatically.

List of exogenous variables Type in this box the names of the variables you want to include as exogenous in your model, separating them with a space. Exogenous variables are facultative: you may leave the box blank. As already stated, constant terms do not require the creation of a column of ones in the excel spreadsheet. Instead, the "include constant in the regression" control should be used, and the code will manage the constant during the whole estimation process. Finally, note that the tip allowing to copy the labels directly from Excel can be used here as well.

Number of lags Simply type the number of lags you want to include in the model.

Include constant Set this control on "Yes" if you want your model to include a constant term. The code will then include a constant in every subsequent step of the estimation process. This choice is set to Yes by default if you select the mean-adjusted BVAR model, since then a constant term is

always included.

Path to data Enter the path to the folder containing the Excel data spreadsheet. By default, the code proposes your project folder. If for any reason you want to place your excel spreadsheet in another folder, you may type instead the path to this folder. For your own convenience it nevertheless advised that you simply leave the spreadsheet in your project folder.

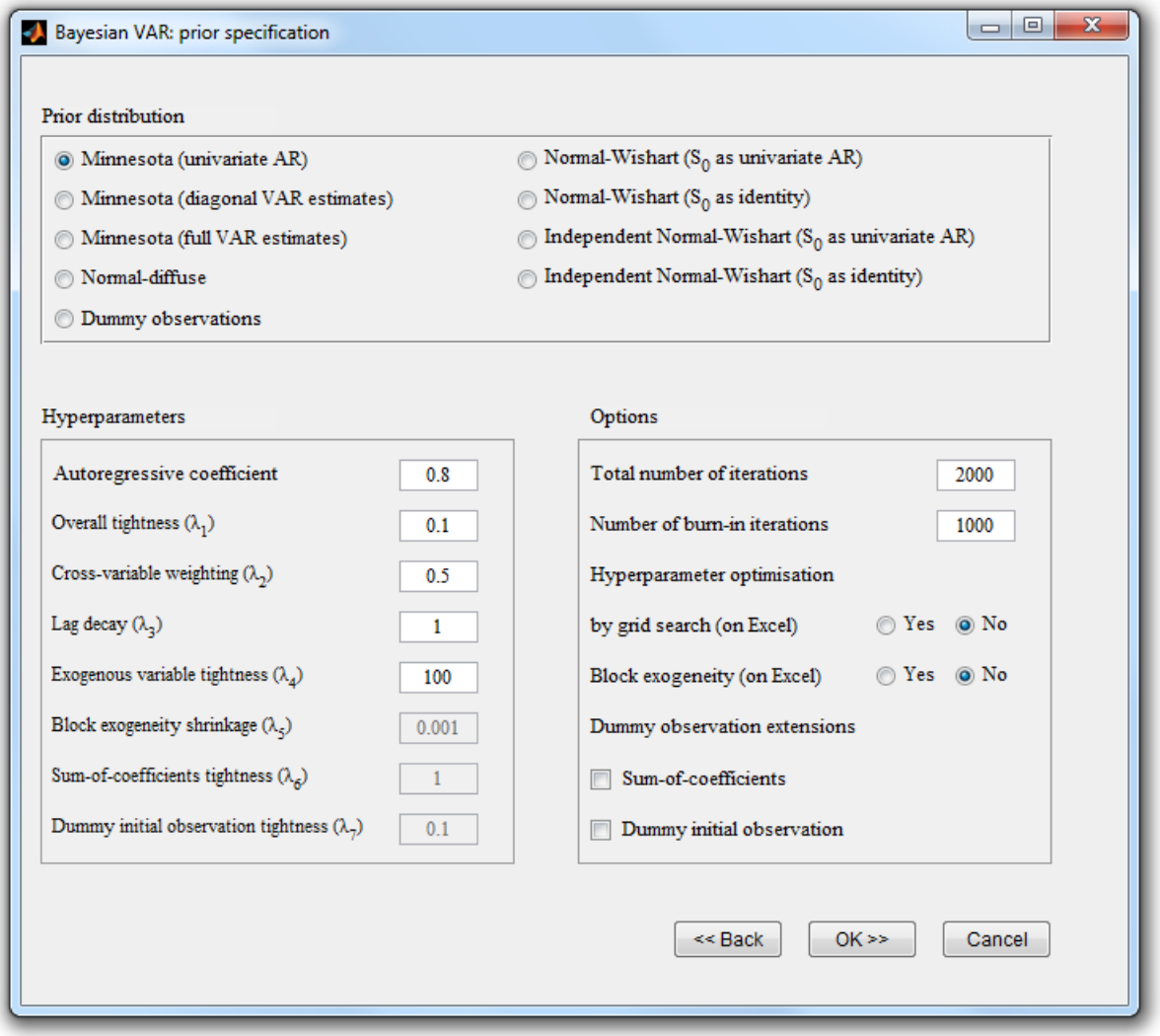
Remember my preferences This is a very important element in this interface. By default, the box is unchecked. If you leave it unchecked, the programme will not save any of your inputs, so that the next time you will run the code, you will have to type again all the information for your model. This can prove quite time-consuming. If you check this box, on the other hand, the programme will save all your inputs, so that the next time you will run the application, this information will be proposed as default values. Of course, you will then be able to change your previous choices if you wish. Checking this box will actually instruct the code to save not only your inputs for this first interface, but also your inputs for all the subsequent interfaces. For this reason, it is highly advised you check this box. If at any time, you uncheck the box later on, it will erase all the saves and re-establish the initial default values.

Once you validate this interface, what will come next will depend on your choice of model. If you select the Bayesian VAR, the mean-adjusted BVAR or the panel VAR, the next interface to open will be specific to the model. Once this specific interface is validated, an application interface common to all models will open. If you select the standard OLS VAR, there will be no model-specific information to input so that the application interface will open directly. All the interfaces are now detailed in turn.

3.3. Setting prior information for a Bayesian VAR

If your selected model is the Bayesian VAR, the interface that will open will look like this:

Figure 3.6.: *Bayesian VAR interface*



This interface comprises three parts: prior distribution, hyperparameters, and options.

Prior distribution: The BEAR toolbox proposes five different prior distributions and their declinations. The first is the original Minnesota prior proposed by Litterman (1986), the second is the normal-Wishart (also called sometimes the natural conjugate prior), the third is the independent normal-Wishart prior with Gibbs sampling, the fourth is the normal-diffuse prior and the last one is a dummy-observation prior. For more details on these different priors, please refer to the technical guide.

Hyperparameters The set of hyperparameters is used to compute the mean and variance of the prior distribution for the VAR coefficients. As you can see, the code proposes default values: these are values typically found in the literature. Hence, even if you do not have a precise idea of the interpretation of these hyperparameters, you can still estimate a BVAR model by using these values

as a reasonable starting point. Note that some values are shaded and deactivated: this is because they correspond the optional applications. λ_5 is associated with block exogeneity, λ_6 with the Sum-of-coefficients application, and λ_7 with the Dummy initial observation application. Therefore, these hyperparameters will activate only if the corresponding applications are selected. For more details about the interpretation of the hyperparameters, please refer to the technical guide.

Options The options allow for additional flexibility in the design of the prior distribution and estimation process. The first two options are related to the Gibbs sampling algorithm: they are used to indicate the total number of iterations of the algorithm, and the number of preliminary iterations discarded as burn-in iterations. More burn-in iterations and a larger total number of iterations lead to more accurate posterior distributions but are more time-consuming. The default values of 2000 total iterations including 1000 burn-in iterations is typically sufficient for most BVAR models.

You may also decide to optimise your hyperpameter values with a grid search, based on the combination optimising the marginal likelihood. While this is a very useful option, it can be very time-consuming. Note also the mention that this application requires additional input on Excel: please refer to section 4.3 for details.

BEAR also allows you to implement block exogenity in your model in order to prevent certain variable to be impacted by other given variables, effectively generating exogeneity between them. This also requires additional input on Excel, and you are refered to section 4.4 for details.

Finally, you may consider implementing dummy observation extensions, especially if you are using non-stationary data (which is typically the case for data in level). Note that you may select either or both the sum-of-coefficients and dummy initial observation extension: simply check the box of the applications of interest. These extensions can apply to any of the five prior distributions proposed in BEAR.

For further theoretical details on block exogeneity and the dummy observation extensions, please refer to the technical guide.

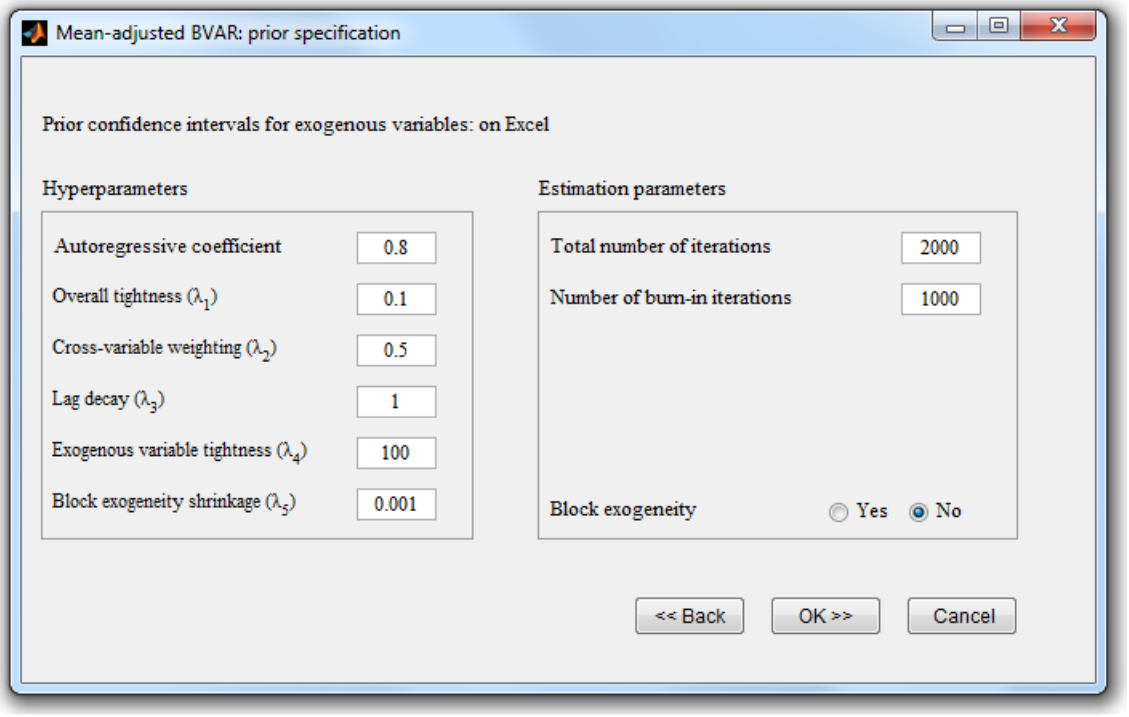
3.4. Setting prior information for a mean-adjusted BVAR

If your selected model is the mean-adjusted BVAR, the interface that will open will look like this:

This interface is very similar to the interface for the Bayesian VAR model, except that it includes fewer options. The interpretation of the different hyperparameters and options is similar to the Bayesian VAR, hence please refer to section 3.3 if you need additional details.

One essential difference with the Bayesian VAR nevertheless is the fact that the mean-adjusted BVAR requires to set prior distribution values for the coefficients on exogenous variables. As indicated on top of the interface, this has to be done by the way of a worksheet in Excel. To obtain more details on how to set your worksheet for the mean-adjusted BVAR, please refer to section 4.1.

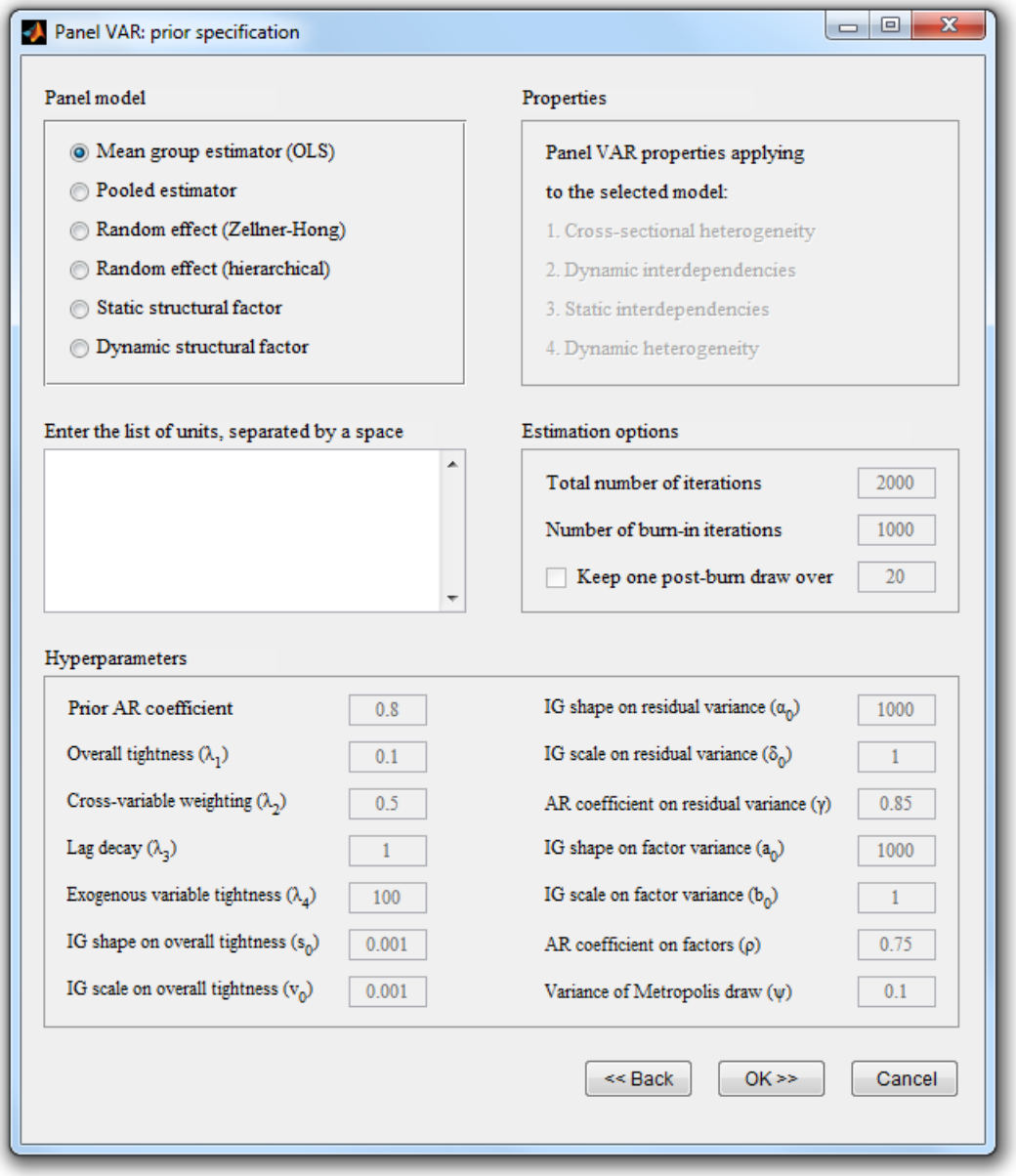
Figure 3.7.: *Mean-adjusted BVAR interface*



3.5. Setting prior information for a panel VAR

If your selected model is the panel VAR, the interface that will open will look like this:

Figure 3.8.: *Panel VAR interface*



There are four distinct parts in this interface: the panel model, the list of units, the estimation options, and the hyperparameters.

Panel model This box is associated with the right box "Properties". The left box allows you to choose your panel model. BEAR includes six types of panel: an OLS mean-group estimator, a Bayesian pooled estimator, two different random effect models, a static and a dynamic structural factor model. The right box indicates the properties associated with the selected model. Shaded properties do not apply, while unshaded properties apply. In the above example, the mean-group estimator is selected, and all the properties are shaded, indicating that none of the four properties

apply for this model. For further details on the different panel models and their properties, please refer to the technical guide.

List of units The text box is used to type the list of the units involved into estimation. The names must correspond to those of the worksheets in the "data.xlsx" file. You do not need to enter the names of all the unit worksheets, it is perfectly possible to work only with a subset of them.

Estimation output This part lets you set the total number of iterations run by the algorithm, and the number of burn-in iterations. The final option allows you to operate a selection over the values obtained during the post burn phase. If the option is selected and the value say for instance to 20, then the results will only be retained for every one out of 20 iterations of the post burn phase. By not retaining successive iterations, this option allows to obtain a better mixing of the posterior distribution and thus improve the quality of the estimation. While it is available for any of the last three models, it is especially useful for the last one (the dynamic factor model). Indeed, this model includes a Metropolis-Hastings step which can lead to repeated values over successive iterations. By retaining only one iteration over so many, it makes sure that the algorithm will be able to obtain a new value between any two retained draws. Note that you do not have to modify the total or burn-in number of iterations to account for this option. For instance, if you set the total number of iterations to 2000, and the number of burn-in iterations to 1000, then activate the option with a value of 10, the code will run 1000 initial burn-in iterations, then in the post burn phase run a total of 10000 iterations keeping only one out of ten, which will result in a final 1000 post-burn iterations and a total of 2000 retained iterations when including the burn-in phase.

Hyperparameters This box allows you to set the hyperparameter values for your panel VAR model. As not all the models are using the same hyperparameter, the interface will only activate the hyperparameters which are used by the selected model. For more details about these hyparameters, please refer to the technical guide. If you want to estimate a model despite limited knowledge on these hyperparameters and their interpretations, you can use the provided default values.

⚠ **Warning** the Zellner and Hong model assumes that the residual variance is common to all units and all variables. Therefore, if you plan to use this model, you should normalise all your variables beforehand, for instance by dividing them by their standard deviation.

⚠ **Warning** the time-varying structural factor model is prone to produce explosive models. This is due to the time-varying structure of the model which can easily result in unit-root behaviours. In this respect, the key parameters are the shape parameters on residual variance (α_0) and structural factor variance (a_0), and the auto-regressive coefficients of the law of motion of residual variance (γ) and structural factors (ρ). The shape parameters are both attributed a default value of 1000, but in case of explosiveness of the model it is possible to increase these values, say to 10000. This reduces the prior variance on the laws of motion which favours stationary behaviours. The auto-regressive coefficients are respectively set at 0.85 and 0.75 by default, but here also it is possible to lower the values (for instance to 0.75 and 0.6) in order to favour stationarity.

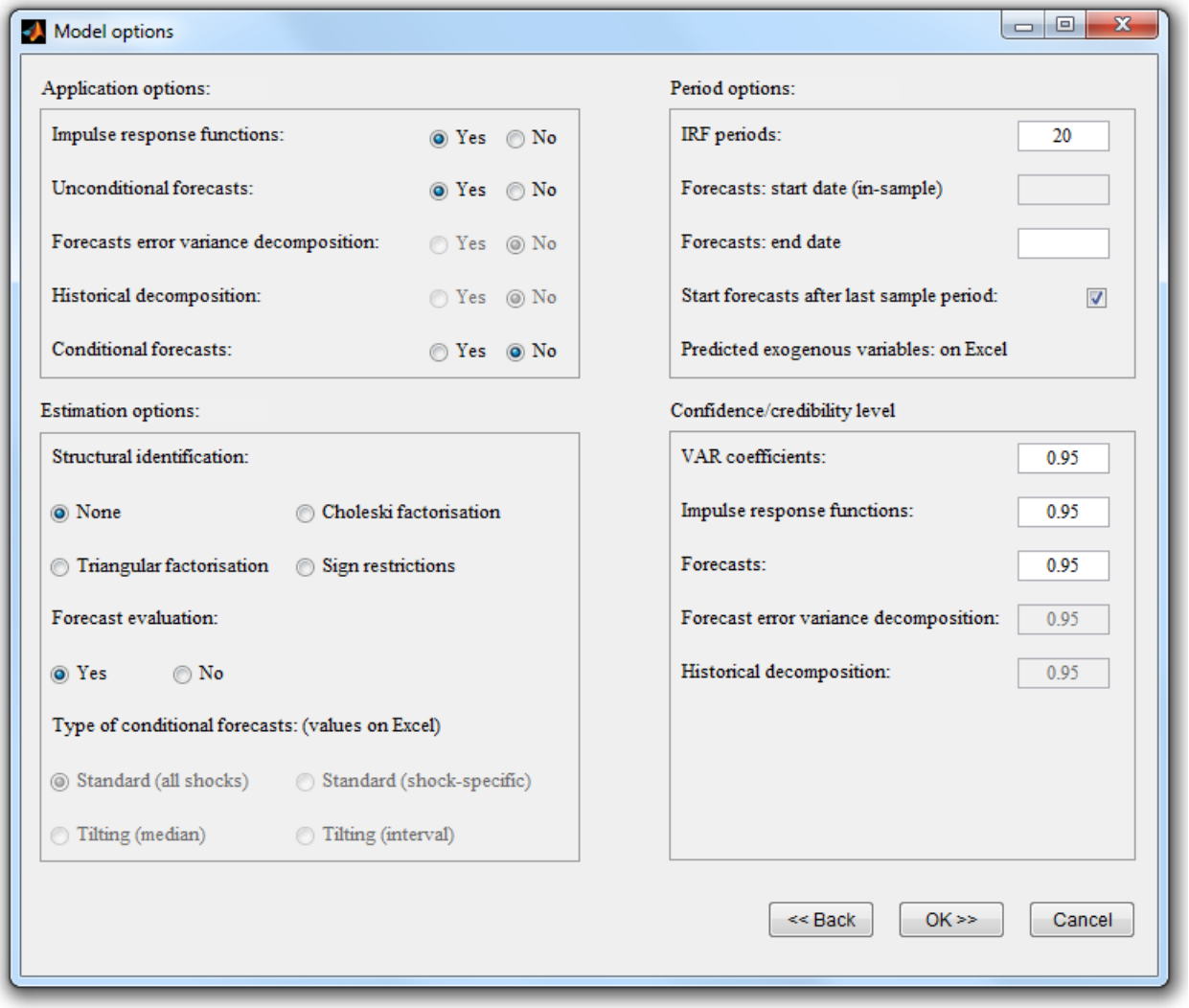
⚠ **Warning** the final hyperparameter (ψ) is used only in the dynamic factor model. It determines the variance and thus the acceptance rate on the Metropolis-Hastings step. This acceptance

rate is reported in your estimation output, and should typically be comprised between 20% and 30%. If it is below 20%, you should decrease the value of (ψ) in order to increase the acceptance rate, and conversely if it is above 30% you should increase the value of (ψ) in order to decrease the acceptance rate. For more details on the Metropolis-Hastings methodology, please refer to the technical guide.

3.6. Setting the applications for your model

The final interface will open once you validate the prior information for your model, or immediately after the first interface if you selected the OLS VAR model. If you selected the Bayesian VAR, it will look like this:

Figure 3.9.: *Application interface*



If you selected any other model, it will be similar, but may contain fewer options. This interface comprises four parts: application options, period options, estimation options, and confidence/credibility levels.

Application options The BEAR toolbox allows you to run five different applications: impulse response functions, unconditional forecasts, forecast error variance decomposition, historical decomposition, and conditional forecasts. If you set an application on no, it will be skipped altogether by the code and no results will be estimated. As an application relies on the Gibbs sampling algorithm, it can prove time-consuming to run, especially with large models. If you are not interested in certain applications, deactivating them can thus constitute an efficient way to save time.

Some applications can be shaded: this is because the toolbox does not allow you to select applications which are incompatible with your current setting. For instance, forecast error variance

decomposition and historical decomposition require a structural identification as they assume orthogonal disturbances. Hence they are unavailable if structural identification is set to "none", as in the default case. Conditional forecasts can be activated only if either unconditional forecasts is activated (in which case only the tilting methodology will be available), or if impulse response functions and a structural identification are activated (in which case the standard methodology will be available).

Period options In this part, you can set the information related to the number of periods for which the applications have to be run. IRF periods allow you to choose the number of periods for which you want to estimate the impulse response functions. The other three controls are related to forecasts. Since most of the time you want to start your forecasts at the period immediately following your final sample period, in order to obtain post-sample forecasts, the code checks by default the box "Start forecasts after last sample period". However, it is possible to start forecasts at an earlier point of the sample. If you want to do so, uncheck the box and enter an in-sample start date for forecasts. Whatever your choice for the start date, you then have to provide an end date for the forecasts. The dates have to be entered in a format that is similar to that of the Excel data spreadsheet.

Remarks

- ⚠ **Warning** If you are using any exogenous variable(s) other than the constant, you also need to provide predicted values for these variables are they cannot be endogenously determined by the model. See section 4.2 for details. If you do not provide such values, the code will fail.
- Forecast error variance decomposition, historical decomposition and conditional forecasts (standard methodology) all require a structural identification scheme. If you set structural identification to "none", these applications will become unavailable.
- As the forecast error variance decomposition application uses the values obtained for impulse response functions, the number of period is automatically the same for the two applications. If you want to increase the number of periods for the forecast error variance decomposition then increase the number of periods for the impulse response functions.
- ⚠ **Warning** Forecasts may start before the end of the sample, however, they cannot start after it (except for the period immediately following the sample end, which is obtained by checking the box). If you enter a date which is after the sample end date, the code will fail.
- The forecast dates you enter will apply to both unconditional and conditional forecasts.
- While the dataset for weekly and daily data may comprise years with any number of periods, forecast years are standardised: forecasts for weekly data will be computed assuming that any post-data year comprises 52 weeks, and forecasts for daily data will assume post-data year of 261 working days

Estimation options This part determines three estimation options. The first is the type of structural identification scheme that applies to the model. This choice will affect the estimation of impulse response functions, and thus consecutively the estimation of forecast error variance decomposition application, historical decomposition and possibly conditional forecast (for the standard methodology). If set to none, the model estimated is a reduced-form VAR model, and the impulse response

functions are obtained from non-orthogonalised unit shocks. If set to Choleski, triangular factorisation or sign/magnitude and zero restrictions, impulse response functions are obtained from the orthogonalised shocks, in accordance with the selected scheme.

The second option is the computation of forecast evaluation criteria. You may want to set this option to no if you are not interested in forecasts or in their evaluation, since some criteria (the log predictive scores) require the use of a Gibbs sampler procedure and can be time-consuming. Note that to obtain forecast evaluation criteria, there needs to be some overlap between your forecast periods and actual data values in your Excel spreadsheet. For instance, if you run a quarterly model and your Excel dataset ends in 2015q4 while your forecasts start in 2015q1, you will have 4 periods of overlap to compute forecast evaluation. If there are no overlapping periods, the code will not return an error but will signal that forecast evaluation was not possible.

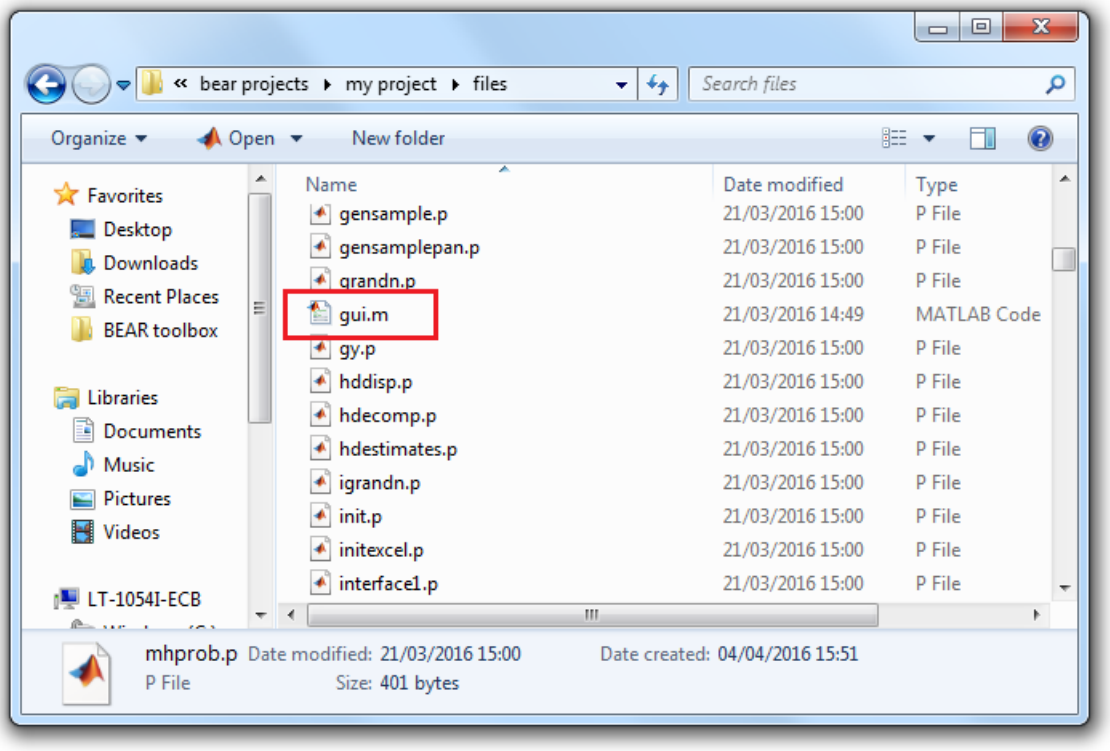
The third option selects the way conditional forecasts are implemented. There can be up to four different options proposed for this exercise. The first two rely on the standard methodology using structural shocks to generate the path of the data, while the last two use a relative entropy approach to generate the conditions. All the methodologies require you to input additional information on Excel: please refer to section 4.6 for details.

Confidence/credibility level Any Bayesian application results in a posterior distribution, from which it is straightforward to obtain probability bands. Therefore, this part proposes to set the confidence level of the bands obtained for the VAR coefficients, along with any application provided by the model. The default values are 95%, following the canonical frequentist approach, but it is customary to lower these values at 65-70% in the Bayesian framework.

3.7. Running BEAR without interfaces

While graphical interfaces represent a convenient way to input your model specifications, some users may prefer to input their information directly in a Matlab script . This avoids spending time validating every interface each time the code is reinitiated. BEAR now allows you to use such a developer's version to input your model information. To do so, first open the "files" folder. Most of the files will appear as p-files (files with ".p" extension), and these files constitute the functions used by the code to run the toolbox. There are however two files respectively called "gui.m" and "settings.m" which appear as editable matlab files. They appear like this in your "files" folder:

Figure 3.10.: *gui.m file*



If you want to switch to the developer's version, first double click on the "gui.m" file to open it. This will open the following window:

Figure 3.11.: *settings.m file*

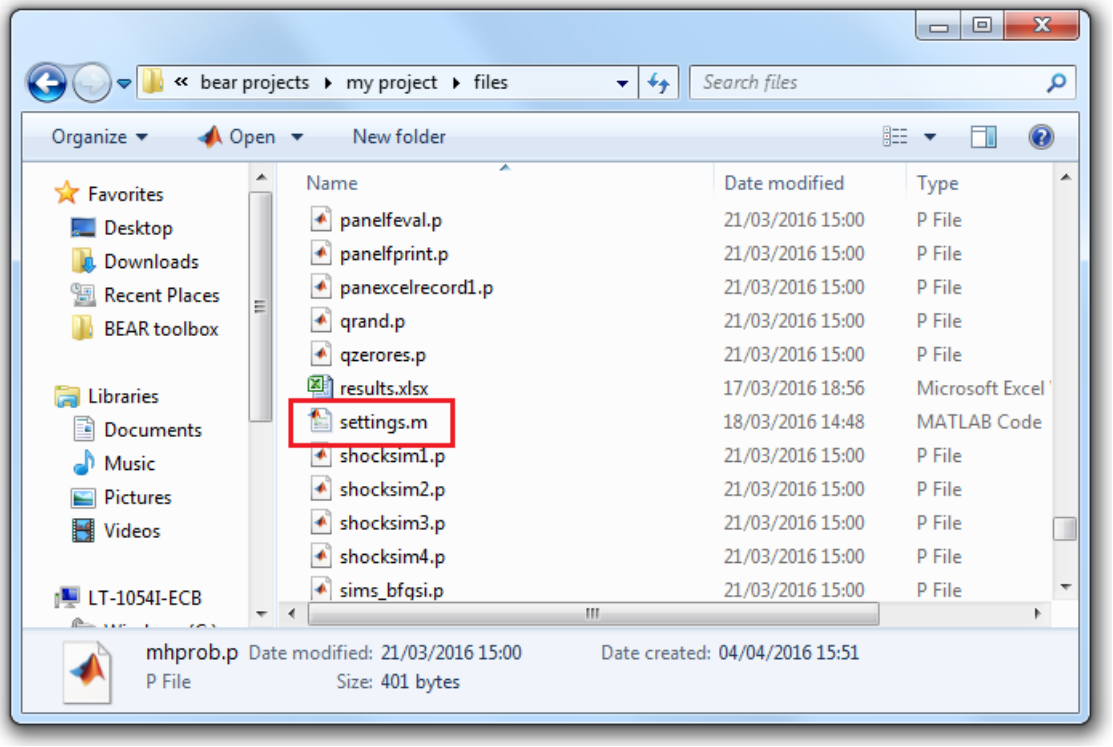
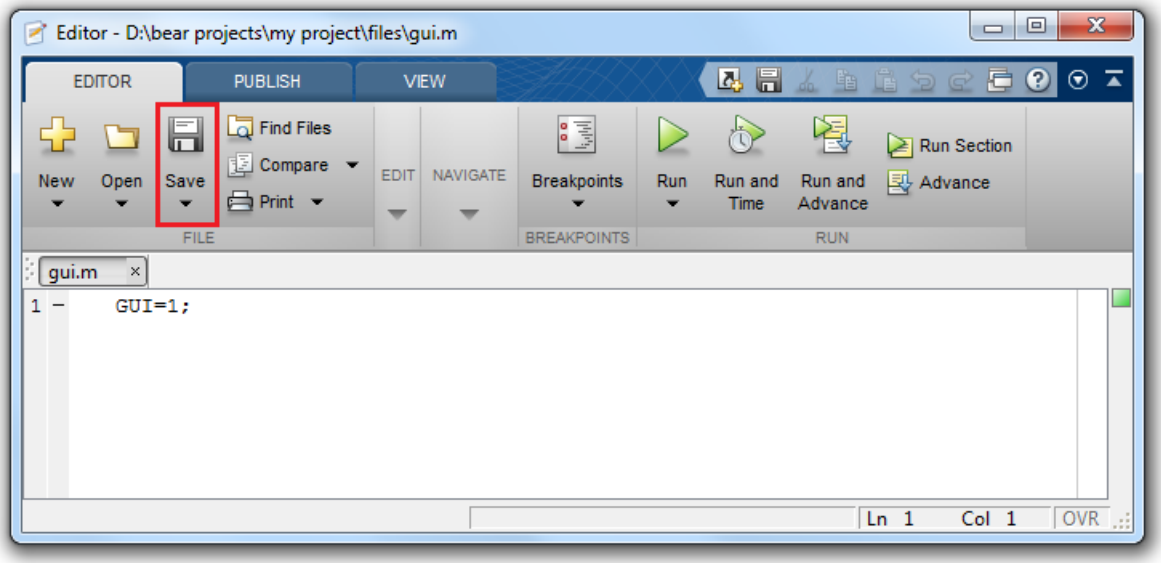


Figure 3.12.: *gui.m window*

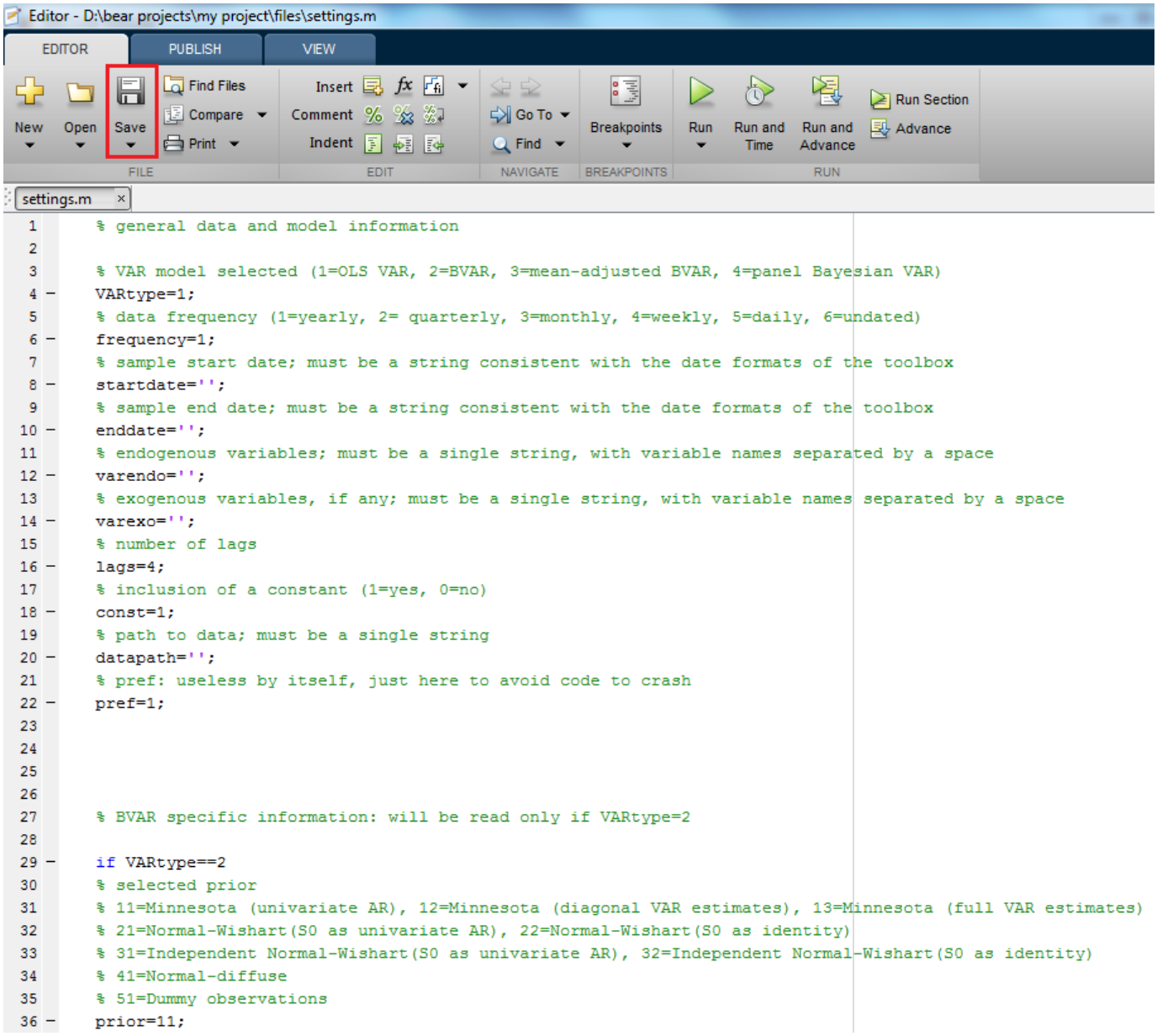


There is a single line of code in this file which states "GUI=1;" . To use the developer's version, change this line into "GUI=0;", then press the save button (circled in red) to save the change. This instructs the code to use the file "settings.m" rather than the interface to run the toolbox. If at any

point you want to use the interface again, reverse the change by setting again "GUI=1;".

So turn now to the file "settings.m". If you double-click it, it will open like this:

Figure 3.13.: *settings.m window*



The correspondence between the interface entries and the script file should be clear. All what you have to do is to modify the file to match the settings of your model, then save by pressing the save button (circled in red). You can then run the toolbox in a standard way by using the "bear_3_0.m" file.

⚠ **Warning** Although the developer's version is available by default, it is recommended that you use it only if you have a sufficient level of familiarity with Matlab and its programming language.

⚠ **Warning** Unlike the interface, the developer's version will not prevent the user to input inconsistent settings. If the setting you are using is inconsistent (for instance: activating forecast error variance decomposition while no structural identification is selected), no warning will be issued and the code may either produce incorrect results or fail altogether.

ï»¿

CHAPTER 4

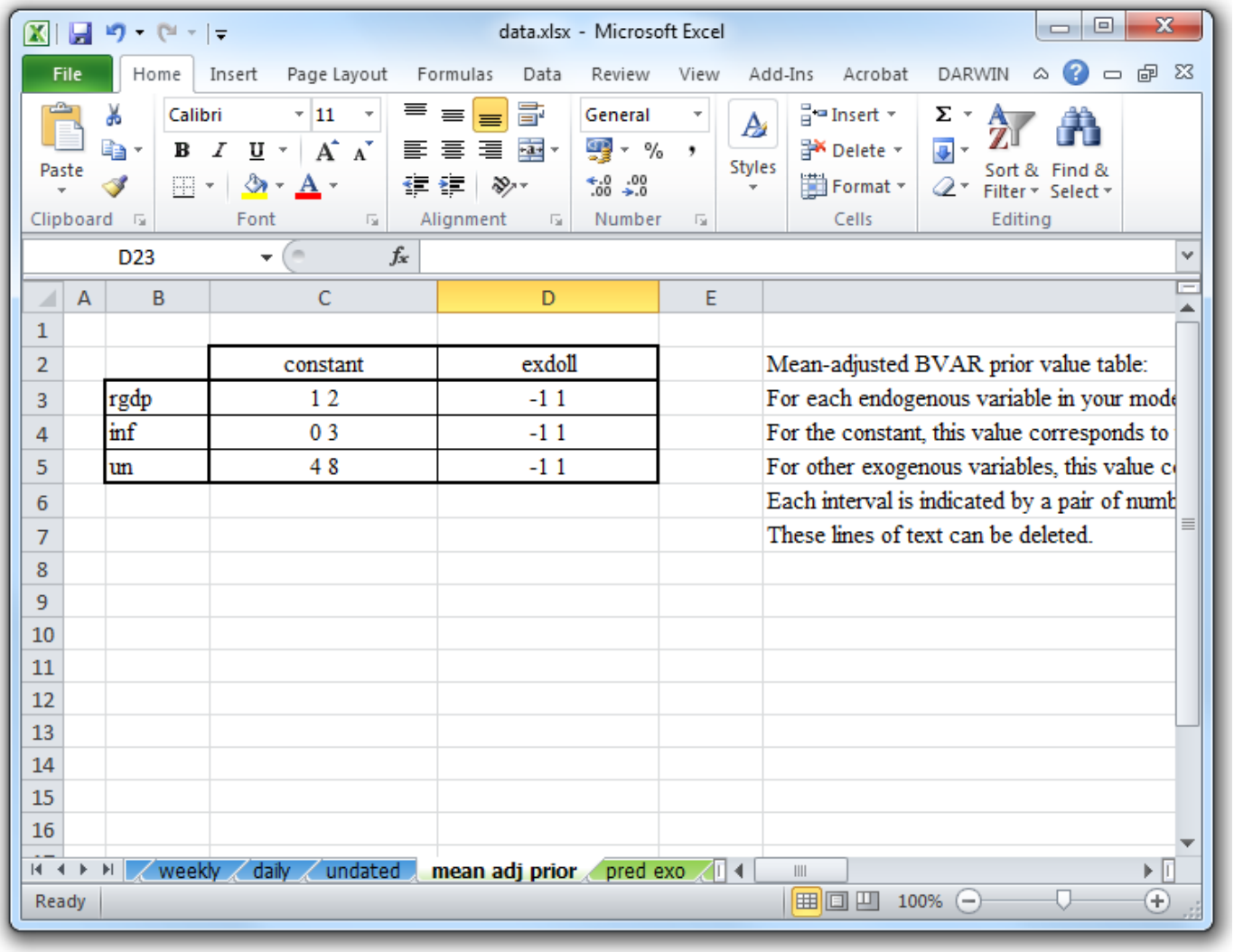
Setting your applications on Excel

Most applications run by BEAR rely on user-specified inputs typically organised in tables. For this reason, Excel tables are used for those applications as they constitute a simple and efficient solution. All the tables are gathered in the same file "data.xlsx", on different worksheets. While the blue worksheets were all used for the data, the green and orange worksheets are dedicated to the application tables. The green worksheets correspond to the OLS VAR, Bayesian VAR and mean-adjusted VAR models, while the orange ones are used for the panel VAR models.

4.1. Prior values for the mean-adjusted model

If you selected the mean-adjusted model, you need to specify prior distribution values for the coefficients on exogenous variables before you can run the model. To do this, open the Excel file "data.xlsx", and select the green worksheet "mean adj prior". It looks like this:

Figure 4.1.: *Excel sheet: mean-adjusted prior*



In the table, the rows correspond to the endogenous variables, and the columns to the exogenous variables. In this example, the endogenous variables are "rgdp", "inf" and "un", while the exogenous include a constant term and "exdoll". Focus for the time being on the constant. Each pair of entries in the column is used to determine the prior distribution for the value of the constant term for the corresponding endogenous variable. It represents a subjective 95% confidence interval set by the user for the value of the constant term, based on a normal distribution. The first term represents the lower bound of the interval, and the second term represents the upper bound. Using the properties of the normal distribution, the prior mean of the distribution is determined as the centre of the interval, while the variance is obtained by the fact that the bounds of a 95% interval are located at 1.96 standard deviations from the mean.

Hence for instance the above example states that our prior belief is that with 95% confidence the

long-run or steady-state value of real GDP growth in the euro area is comprised between 1% and 2% per annum, which corresponds to a steady-state mean of 1.5%, and a variance of approximately 0.065. A tighter interval would imply smaller prior variance and hence greater confidence that the steady-state value corresponds to the specified prior mean, while a looser interval would imply larger prior variance and give more weight to the data. Similar intervals have then to be created for the constant with all the endogenous variables. Note that the constant is mandatory for the mean-adjusted model: as the computation of the steady-state represents the main contribution of this model, the constant term cannot be suppressed. Deleting the column for the constant will indeed cause the code to return an error.

If there are additional exogenous variables (such as the variable "exdoll" in the above example), an additional column must be created on the sheet for each extra exogenous, to the right of the constant. A 95% confidence interval has then to be specified for each endogenous variable and each additional exogenous variable. The interpretation however becomes more difficult: unlike the constant, it does not represent the steady-state value, but the value of the coefficient relating the endogenous variable to the specified exogenous variable. For instance, the interval specified for rgdp relative to exdoll represents the value of the coefficient giving the impact of the euro-dollar exchange rate on the growth rate of GDP of the euro area. It may typically not be obvious to determine what such a value could be.

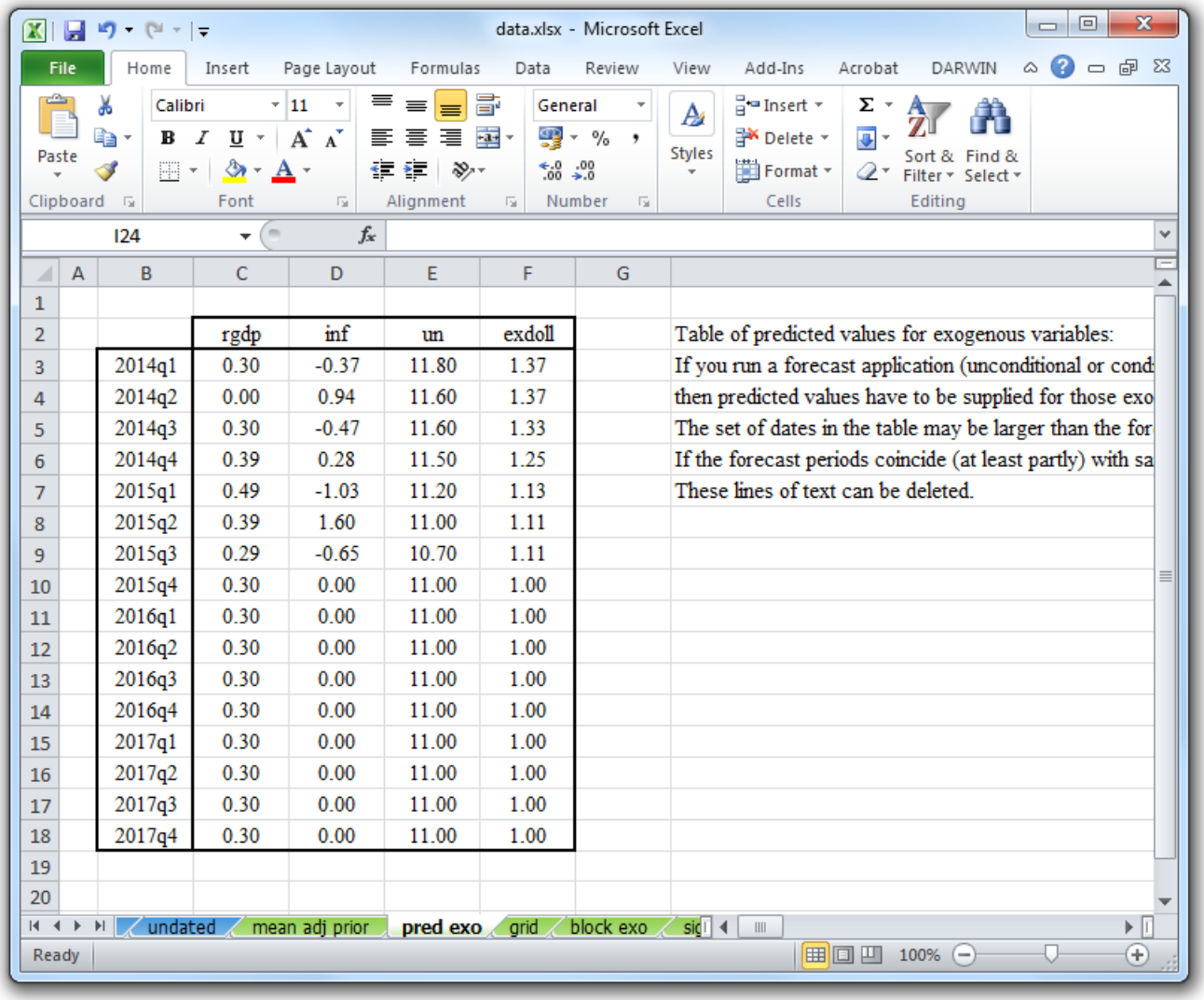
The solution adopted in the example is to set an interval centred around 0 to express prior agnosticism about the coefficient value, but this is a rather poor solution. Indeed, if the coefficient is actually different from 0, then the prior is biasing the information used by the model towards a 0 value for the coefficient, which may result in a lower quality of estimation. It could be tempting to set a very large prior interval in order to implement a diffuse prior, but this is not an option: Villani (2009) shows that if the prior distribution is not at least mildly informative, the posterior distribution obtained from the Gibbs sampler will not be correct. This is what motivates the use of a reasonably small interval here with values [-1 1]. For these reasons, it may be preferable to avoid using exogenous variables other than the constant for a mean-adjusted VAR model, if this is possible.

Finally, note that the sheet allows for some flexibility: it can contain information about more variables than the ones actually used for the model. For instance, the sheet for the above example could be used to estimate a model which would only include "rgdp" and "un" as endogenous, and "constant" as exogenous. This is not a problem: the code will only select the relevant variables.

4.2. Predicted exogenous (OLS VAR, Bayesian VAR and mean-adjusted BVAR)

If you want to produce forecasts for your model and if your model includes any exogenous variable(s) other than the constant, you will need to provide predicted values for these variables over your forecast windows. Indeed, the model is unable to produce forecast values for such variables, so that they have to be user-specified. To do so, open the Excel file "data.xlsx", and select the green worksheet "pred exo". It looks like this:

Figure 4.2.: *Excel sheet: predicted exogenous*



	rgdp	inf	un	exdoll
2014q1	0.30	-0.37	11.80	1.37
2014q2	0.00	0.94	11.60	1.37
2014q3	0.30	-0.47	11.60	1.33
2014q4	0.39	0.28	11.50	1.25
2015q1	0.49	-1.03	11.20	1.13
2015q2	0.39	1.60	11.00	1.11
2015q3	0.29	-0.65	10.70	1.11
2015q4	0.30	0.00	11.00	1.00
2016q1	0.30	0.00	11.00	1.00
2016q2	0.30	0.00	11.00	1.00
2016q3	0.30	0.00	11.00	1.00
2016q4	0.30	0.00	11.00	1.00
2017q1	0.30	0.00	11.00	1.00
2017q2	0.30	0.00	11.00	1.00
2017q3	0.30	0.00	11.00	1.00
2017q4	0.30	0.00	11.00	1.00

Table of predicted values for exogenous variables:
If you run a forecast application (unconditional or cond
then predicted values have to be supplied for those exo
The set of dates in the table may be larger than the for
If the forecast periods coincide (at least partly) with sa
These lines of text can be deleted.

The sheet is self-explanatory: the first column of the table contains the dates, while the first row of the table contains the variable names. The entries within the table correspond to the user-specified predicted values. The table has to provide values for each exogenous variable included into your model and every period comprised in your forecast window. The table however is flexible: it may include more variables and cover more dates than what you use in your model. This is not a problem: the code will only select the dates and variables which are included in your model. This is useful as it may allow you to modify the set of exogenous variables and forecast dates in the interface without having to update your table.

Indeed, remember that BEAR allows you to specify any variable of the spreadsheet as exogenous. In the above example, even though "exdoll" will be typically the only variable used as an exogenous, it may be useful to input predicted values for the other variables as well. This way, if you decide at any point for instance to switch "un" from endogenous to exogenous, you will not have to modify the spreadsheet as predicted values will already be specified. Of course, if you are certain that you will only ever use "exdoll" as an exogenous, you may just delete the other columns. For similar reasons, the forecast window in the example is very large in terms of dates. This permits to change freely the start or end date for the forecasts without having to update the table.

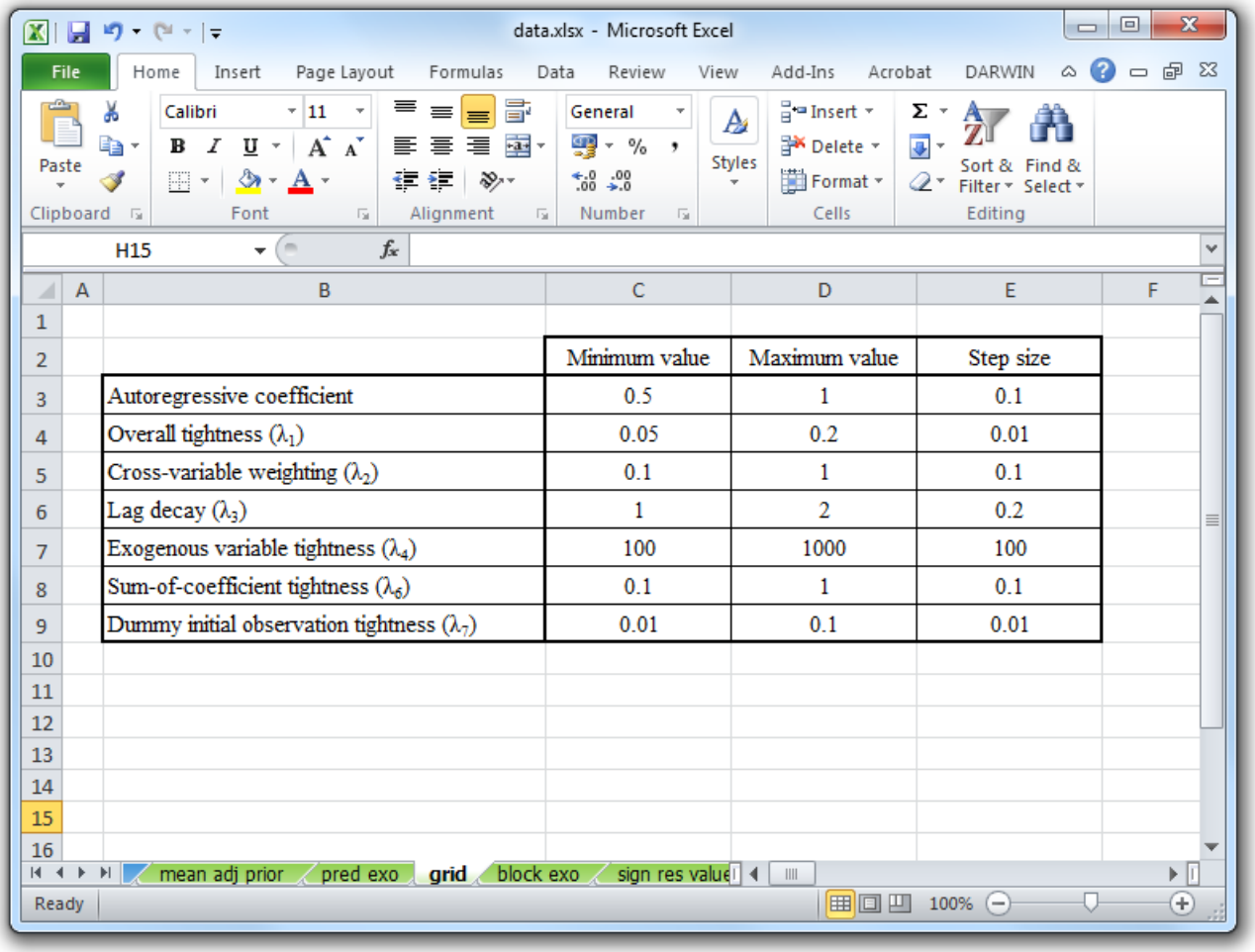
While the specification of predicted exogenous may look like a constraint, it can also constitute a powerful tool for scenario analysis. First, note that if your forecasts overlap (at least partly) with your dataset, you may just use actual values for your predicted exogenous. This is the case in the above example: since actual values are available for the period 2014q1-2015q3, they are used as predictions. You may then compare the results obtained when using actual values with the ones you would obtain by using an alternative scenario. For instance, assume you run a forecast exercise over the period 2014q1-2015q1, with "exdoll" as only exogenous. You may first obtain forecast using the actual values (1.37, 1.37, 1.33 and 1.25), then compare the results with a scenario where "exdoll" would have instead jumped to 1.5 over the same period. In this sense, predicted exogenous represent a useful tool for conditional forecasting.

In general, if you do not have actual values to use for your predictions, the following strategies are possible: you can specify your own values, which is then again a form of scenario analysis; or you can replicate the last known value, which represents an agnostic approach; or you can obtain forecasts for each exogenous considered on its own and use those forecasts values for your predictions. The last option can be implemented easily: simply use the OLS VAR option of BEAR to estimate a VAR model for each exogenous independently (which then reduces to estimating individual AR models) and produce forecasts for these models.

4.3. Grid search (Bayesian VAR)

If you want to run a Bayesian VAR model, you have the possibility to optimise your hyperparameters by using a grid search. To do so, you need to select the option in the Bayesian VAR interface, and then to specify the values of your grid on Excel. To do this, open the Excel file "data.xlsx", and select the green worksheet "grid". It looks like this:

Figure 4.3.: *Excel sheet: grid search*



	Minimum value	Maximum value	Step size
Autoregressive coefficient	0.5	1	0.1
Overall tightness (λ_1)	0.05	0.2	0.01
Cross-variable weighting (λ_2)	0.1	1	0.1
Lag decay (λ_3)	1	2	0.2
Exogenous variable tightness (λ_4)	100	1000	100
Sum-of-coefficient tightness (λ_6)	0.1	1	0.1
Dummy initial observation tightness (λ_7)	0.01	0.1	0.01

The grid search works as follows. For each hyperparameter, the code first defines the grid by creating a range of values, starting from the minimum value and then incrementing sequentially this value by the step size until the maximum value is reached. Then it estimates the model for every possible combination of values, and for each model calculates the marginal likelihood. The optimal hyperparameter combination is then determined as the one which maximises this value.

What you have to do is thus set the minimum value, maximum value and step size for each parameter in the grid. You can use the default values provided for the grid if you are not sure about which values you should test. In general larger ranges and smaller step sizes will result in a finer grid but are more time-consuming to run. Some values of the grid may not be actually used during the search: if your model uses the normal-Wishart prior, the cross-variable weighting λ_2 does not enter into the estimation process and hence this part of the grid will be ignored. Similarly, λ_6 and λ_7 are related to the sum-of-coefficients and dummy initial observation applications, so these parts of the grid will be ignored if these application are not selected. Note that you do not need to clear the contents of the cells that will not be used as the code will simply skip them.

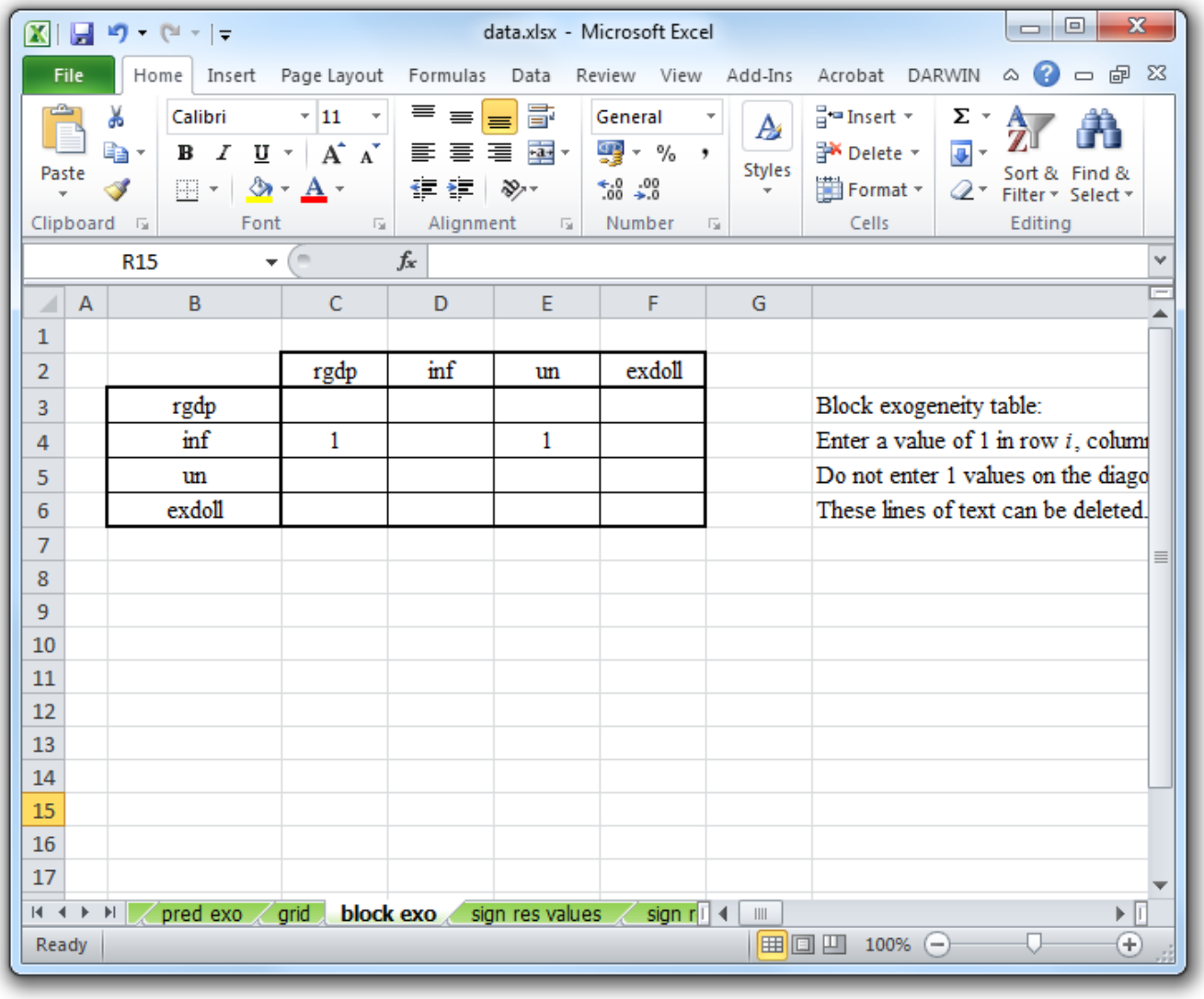
⚠ **Warning** Grid search can be very time-consuming!

⚠ **Warning** The code is designed to read the table as it is found in your Excel sheet. Do not delete nor move any row or columns in the sheet (though you may clear the side text if you wish). Doing so would prevent the code to run properly.

4.4. Block exogeneity (Bayesian VAR and mean-adjusted BVAR)

If you run a Bayesian VAR model or mean-adjusted BVAR model, you can implement block exogeneity to create partial exogeneity within your set of endogenous variables. To do so, you need to indicate in an Excel table which variable of your endogenous set should have no impact on which other. Open the Excel file "data.xlsx", and select the green worksheet "grid". It looks like this:

Figure 4.4.: *Excel sheet: block exogeneity*



The first row and the first column of the tables contain the labels for the variables in your dataset. To implement the fact that variable i will have no effect on variable j, enter a 1 in row i, column j of the table. In the above example, there is a 1 in row 2, column 1 of the table which indicates that variable 2 (inf of inflation) will have no effect on variable 1 (rgdp, or real GDP). This reflects the traditional assumption of neutrality of nominal variables on real variables. Similarly, the 1 value in row 2, column 3 indicates that variable 2 (inf for inflation) will have no effect on variable 3 (un for unemployment).

In this example, the table contains all the variables of the example dataset. This includes "exdoll" which will be typically used as an exogenous variable, even though block exogeneity is only concerned with endogenous variables. This is because BEAR allows you to switch "exdoll" to endogenous at any point, so including it into the table avoids to update the table later on. In general,

you can build your table with just as many variables as you want from your dataset. Variables that are irrelevant for the application of block exogeneity for your model will simply be ignored.

4.5. Sign restrictions (Bayesian VAR and mean-adjusted BVAR)

The BEAR toolbox allows you to implement the structural identification scheme by sign and zero restrictions proposed by Arias et al. (2014). The methodology has been extended to also allow for magnitude restrictions. For more details on the methodology, please refer to the technical guide. Setting your restrictions is done by the way of two Excel tables. Open the Excel file "data.xlsx", and select the green worksheets "sign res values" and "sign res periods". The tables look like this:

Figure 4.5.: *Excel sheet: sign restrictions*

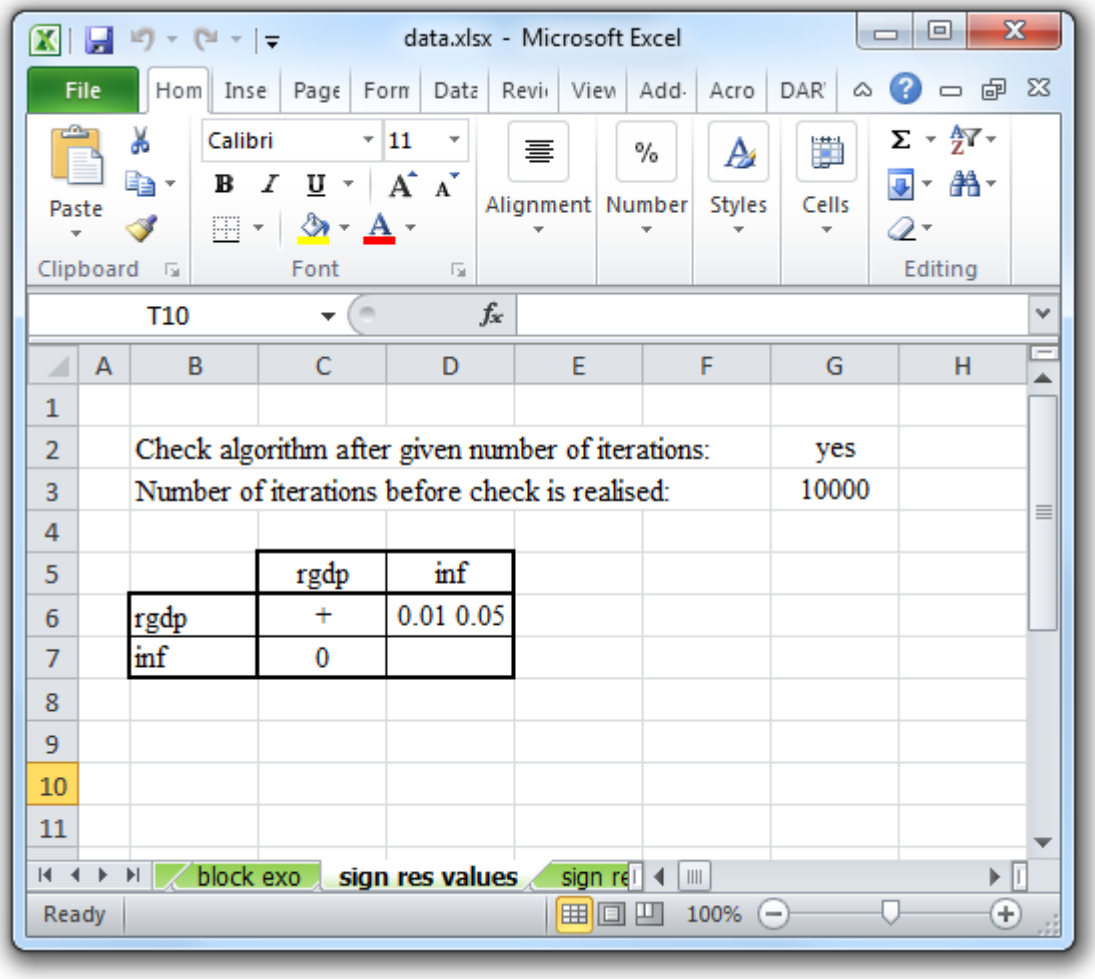
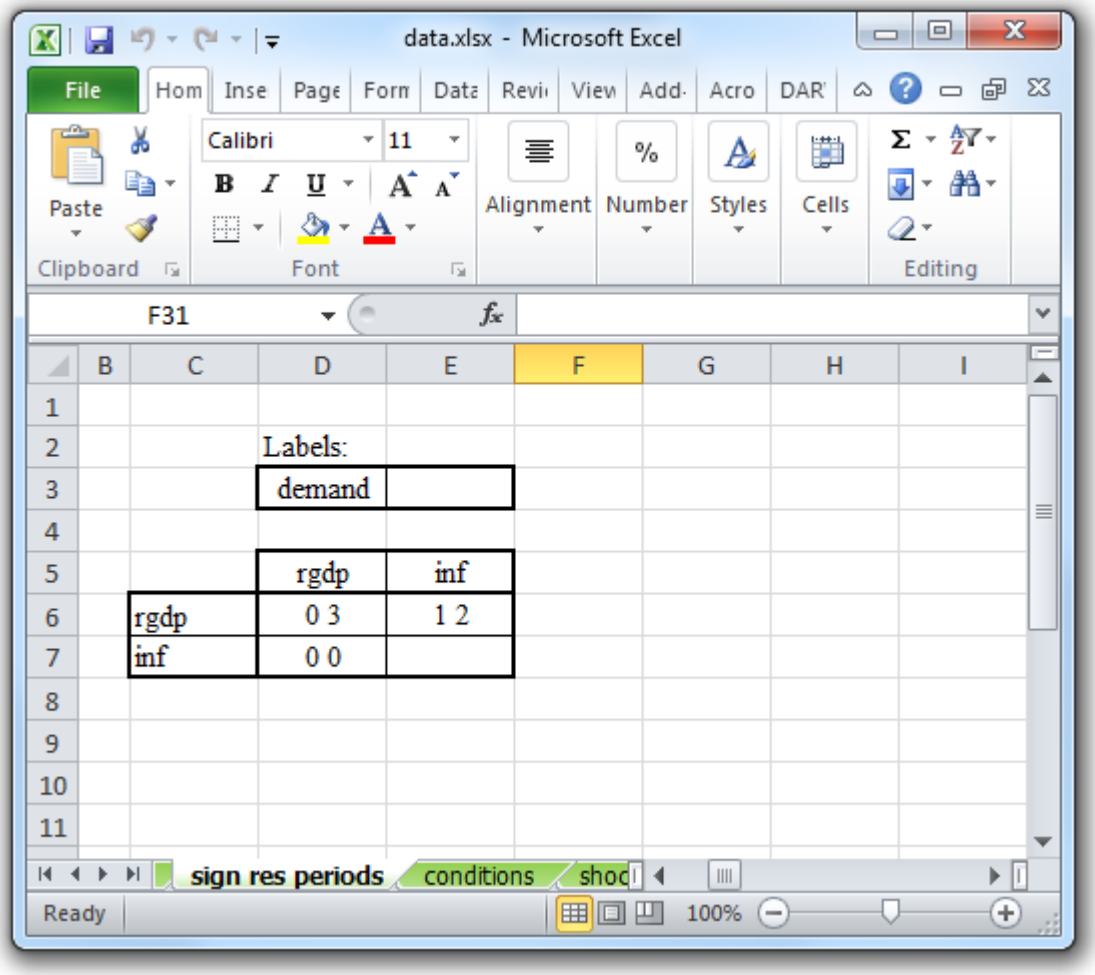


Figure 4.6.: *Excel sheet: restriction periods*



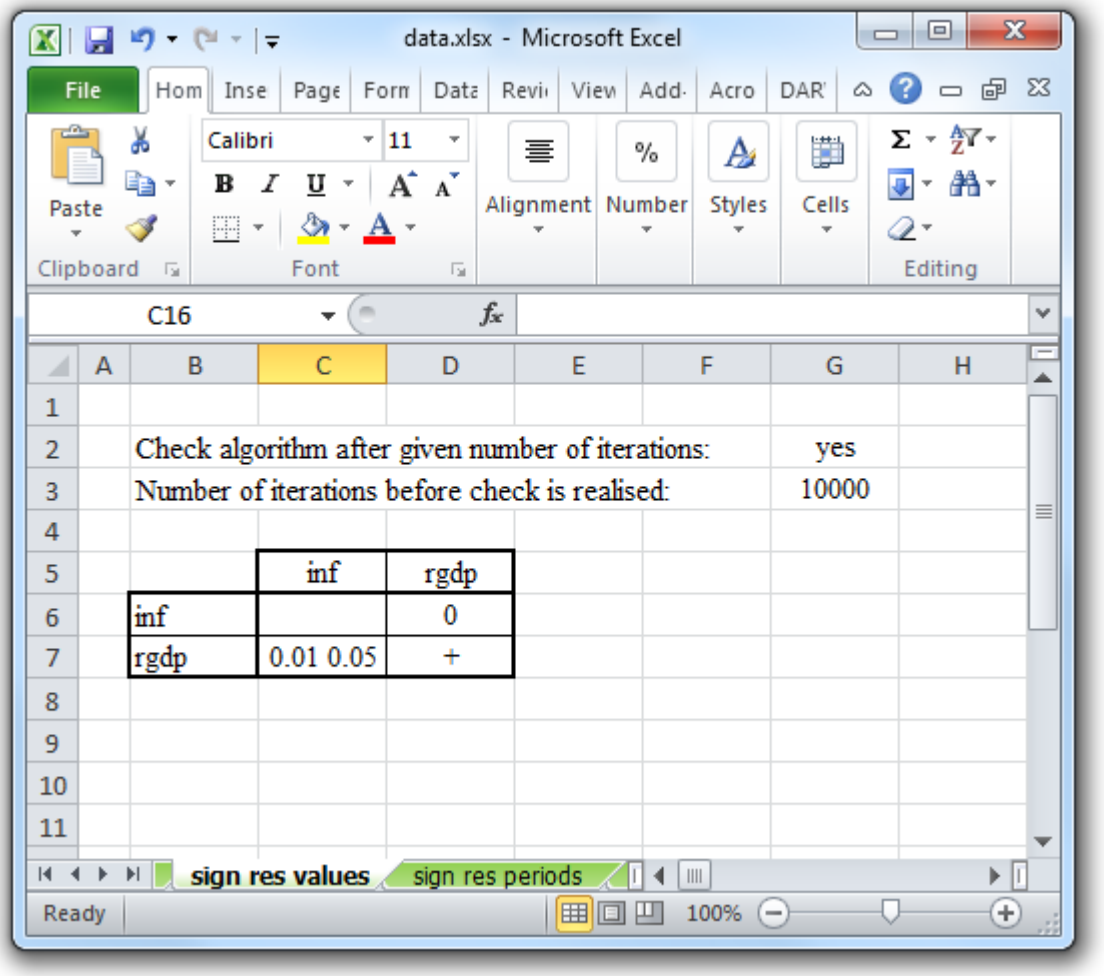
The first sheet ("sign res values") is dedicated to the restrictions. The rows correspond to your variables, while the columns correspond to the associated structural shocks. So for instance the " - " entry in row 1 (rgdp), column 2 (inf) of the table corresponds to a restriction on the impulse response functions of rgdp to the structural shock associated to inf.

There are three possible types of restrictions: - sign restrictions: restricts the impulse response functions to be positive or negative. For a positive restriction, enter a "+" in the corresponding entry, for a negative restriction enter a "-" in the corresponding entry. An example can be found in row 1, column 1 of the table. - zero restrictions: restricts the impulse response functions to take a value of zero (no response). Enter a "0" in the corresponding entry. An example can be found in row 2, column 1 of the table. - magnitude restrictions: restricts the impulse response functions to be comprised between a given range of values. Enter the range as two numbers separated by a space, such as "1 2". An example can be found in row 1, column 2. Note that your range cannot be made of a repeated number, for instance "1 1" will lead the code to fail.

For technical reasons there exists a limitation on the number of zero restrictions you may implement on each variable. For a model with n variables, there should be a most $(n-j)$ zero restrictions associated with shock j. That is, there should be at most $(n-j)$ zero entries in column j of the table, if the table only contains the variables of your model (and not the other variables of your dataset). For this reason, it is recommended that you include only the variables of your model for sign restriction tables, even though you may include other variables of the dataset if you manage correctly your zero restrictions. In the above example, the model only contains two variables (so n=2), and there is one zero restriction in column $j = 1$ of the table. As $n - j = 1$, this sign restriction setting will be properly identified.

If your sign restrictions don't satisfy this requirement, there may still be solutions to solve your problem. Unlike Choleski identification schemes where the order of the variables matters, the order is irrelevant for sign restriction identification. For instance, You may assume that an initial version of the above example model could have been written as:

Figure 4.7.: *Excel sheet: modified sign restrictions*

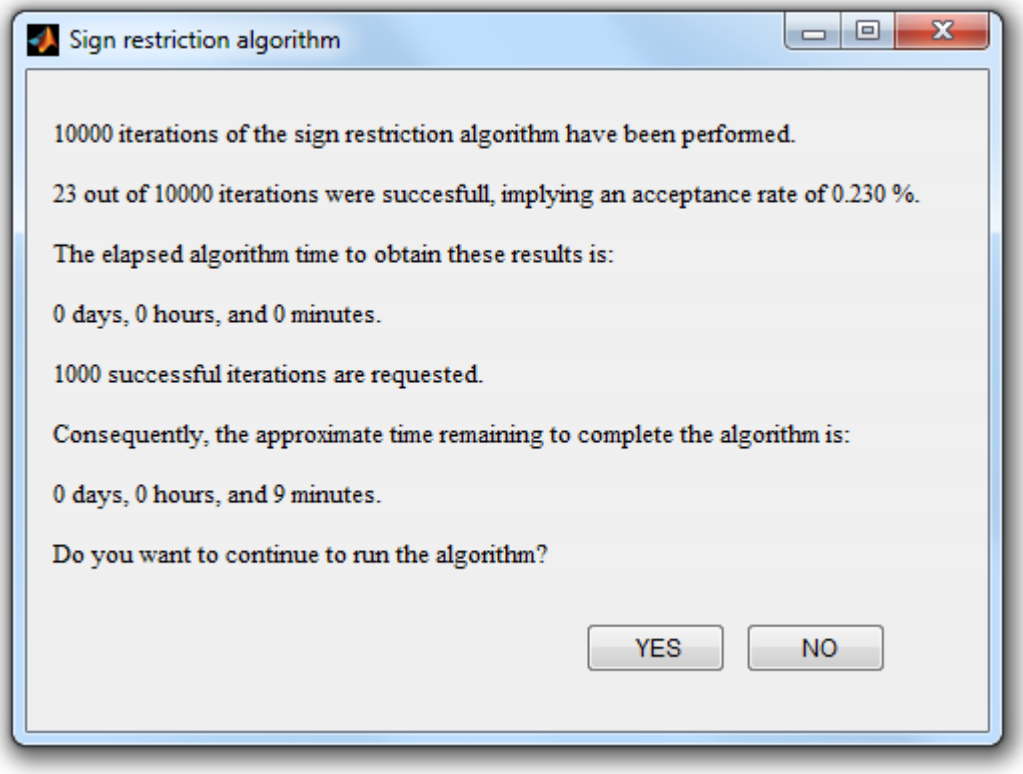


Comparing with Figure 4.5, you can see that this corresponds to the same setting, save for the order of the variables that has been inverted. Even though the model is similar, this sign restriction setting could not be identified since there is one zero restriction in column 2 of the table, while this column can contain at most $n - j = 0$ restrictions. It suffices however to switch the ordering of the variables as in Figure 4.5 to obtain a well identified setting.

The first sheet contains two additional elements. The first line of text "Check algorithm after given number of iterations" allows you to decide whether you want to check the state of the algorithm after a determined number of iterations. The second line determines the number of iterations. Depending on your restriction setting, the time required to run the algorithm can indeed be very long. The algorithm may also fail altogether. If you select this option, a window will appear in the course of the estimation and indicate how many successful iterations have been obtained, the time elapsed to obtain those iterations, and an estimated remaining time:

Depending on the indications obtained from this window, you may then decide to continue or not to

Figure 4.8.: *Remaining time window*



run the algorithm.

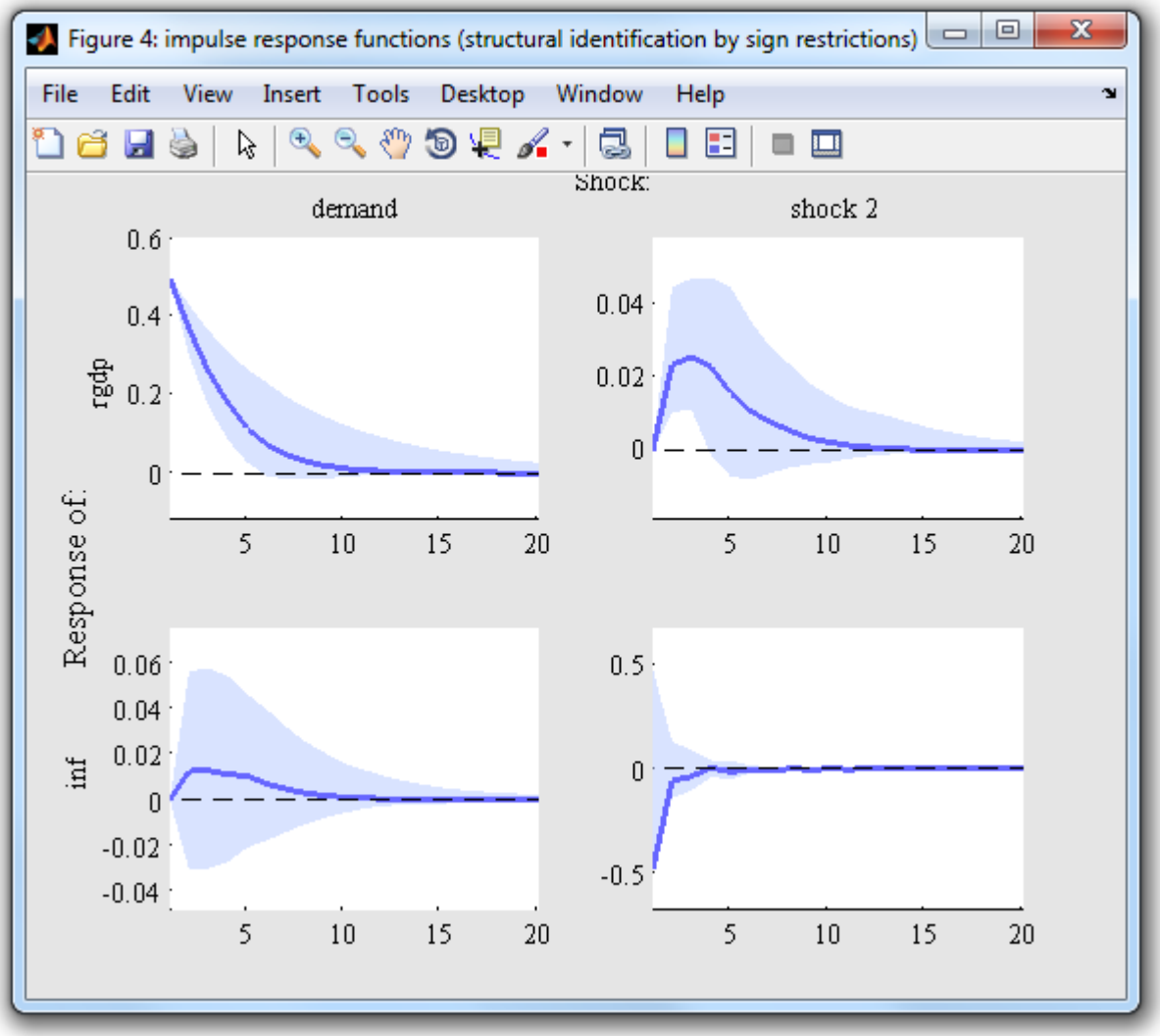
By default, the option of checking algorithm is set to "yes". You may just set it to "no" if you want to deactivate it. You can also modify the number of iterations at which the check is realised.

The second sheet is used to indicate the periods at which the restrictions apply. Periods have to be specified for each restriction created in the first sheet in the corresponding table entry of the second sheet. The periods are defined as a range written as two number separated by a space, the first number being the initial period of application of the restriction, the second being the final period of applications, for instance "2 3". Note that the first period of the impulse response functions (impact) is defined as 0. Hence for example a quarterly model, the periods for a restriction that applies over the first four quarters will have to be written as "0 3" (implying a restriction over periods 0, 1, 2 and 3). To specify a restriction that applies to a single period, simply repeat the value. For instance, "0 0" denotes a restriction that applies only at impact. Of course different restrictions may be defined over different periods, as should be clear from the example table.

There is an additional element in this sheet: row 3 of the sheet allows you to create your labels for the structural shocks you are identifying. If you enter names for your structural shocks, these names will be used in any plot involving the shocks, for instance the impulse response functions:

You can see on the above plot that if for some of the shocks no name is entered, say for instance

Figure 4.9.: *Shock labels on applications*



here the second shock, then the label on the plot will simply be "shock 2".

⚠ **Warning** In general, you should avoid implementing too many restrictions, or restrictions that are too much at odds with your unrestricted results. Doing so will typically result in a very low rate of acceptance of the draws for the algorithm, leading either in extremely long estimation time or to complete failure of the algorithm. You may also question the economic relevance of restrictions associated with an extremely low acceptance rate.

⚠ **Warning** Sign, magnitude and zero restrictions have to be defined carefully by the user. It is indeed possible that the setting you implement is economically irrelevant: if several different shocks end up having the same effect on the variables of your model, it is not possible to disentangle them and your setting is not correctly identified. The program will yet run irrespectively of your scheme and will not issue any warning. It is thus your responsibility to define your restrictions properly.

⚠ **Warning** The code is designed to work with the sheets as they are formatted. While the tables should be modified to suit your model, the iterations and label options found in the first four rows of the two sheets should not be deleted or displaced. Doing so would result in the code to fail.

4.6. Conditional forecasts (Bayesian VAR and mean-adjusted BVAR)

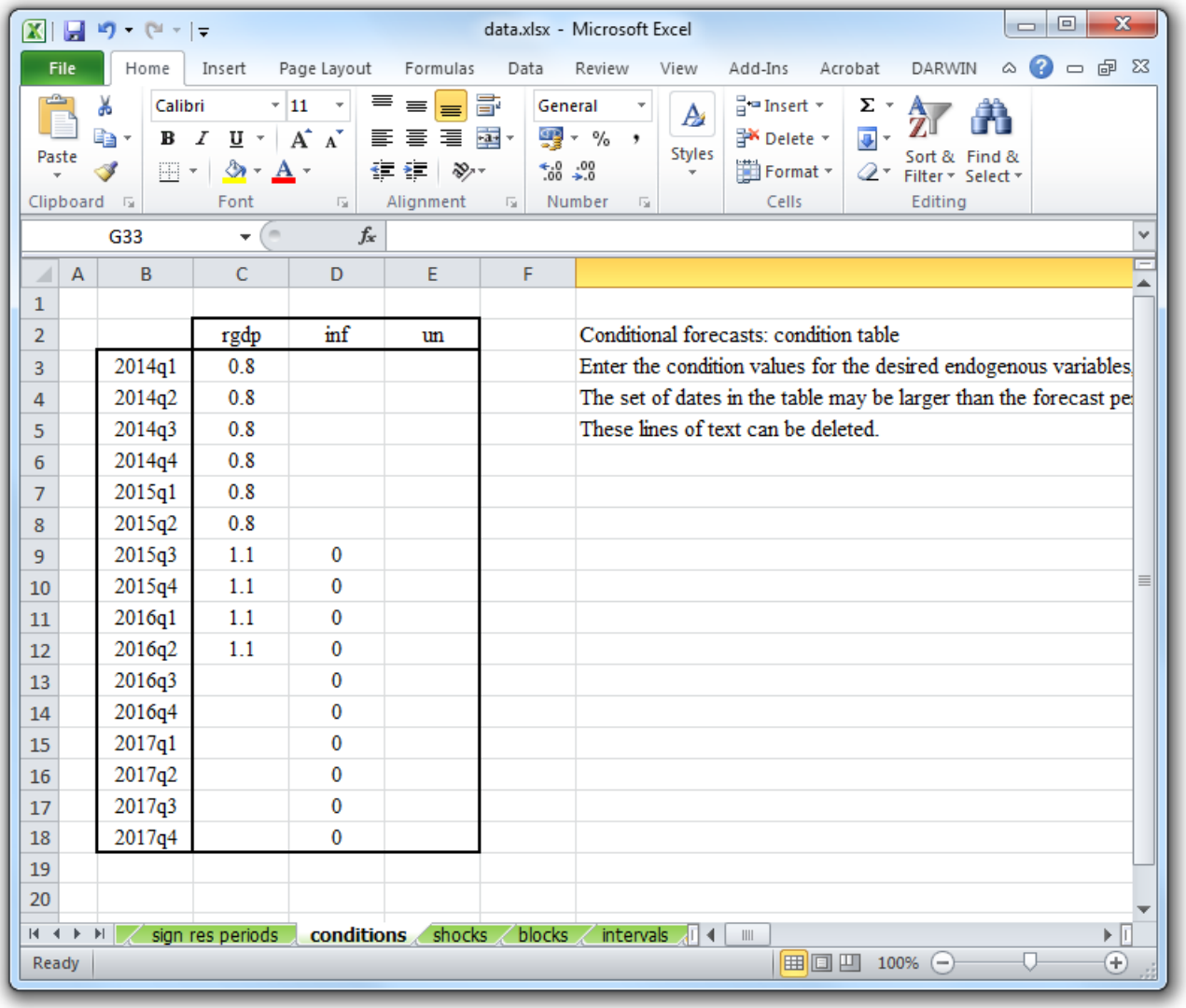
Conditional forecasts constitute a great tool to study scenario analysis. This methodology makes it possible to answer questions like: "what will happen to the Euro area GDP growth rate if the short term interest rate is set at the 0% lower bound for the next eight quarters?", or "what will happen to the unemployment rate in the euro area if GDP growth stabilizes at a modest 1% over the next four quarters?". The approach is powerful and flexible, but is a bit more advanced than the previous applications, and requires some care, for otherwise meaningless results may be obtained, or the technique may fail altogether.

The BEAR toolbox allows you to use two main conditional forecast methodologies, each of them allowing for two declinations. The presentation starts with standard methodology, based on shocks.

4.6.1. Standard methodology (all shocks)

If you opt for the standard methodology (all shocks), all what you have to do is to fill an Excel spreadsheet specifying the path for your variables. Open the Excel file "data.xlsx", and select the green worksheet "conditions". The table looks like this:

Figure 4.10.: *Excel sheet: conditions*



	rgdp	inf	un
2014q1	0.8		
2014q2	0.8		
2014q3	0.8		
2014q4	0.8		
2015q1	0.8		
2015q2	0.8		
2015q3	1.1	0	
2015q4	1.1	0	
2016q1	1.1	0	
2016q2	1.1	0	
2016q3		0	
2016q4		0	
2017q1		0	
2017q2		0	
2017q3		0	
2017q4		0	

The above example is once again based on the euro area model comprising real GDP, inflation and the unemployment rate. Let us assume thus that those three variables enter into the model, that the sample period covers the full dataset (1999q1-2015q3), while the forecasts cover the period 2015q1-2016q4.

To make things clearer, let us state some definitions. A condition is simply a value that a variable is imposed to take at a given period by the user (hence the name conditional forecasts). For instance in this example the conditions over the forecast periods are that GDP growth should be equal to 0.8% for the two periods 2015q1 and 2015q2, then should take value 1.1% until 2016q2; inflation on the other hand should be equal to 0% over the whole period 2015q3-2016q4. This represents a total of 12 conditions. In general, to create a condition, simply type the condition value for the corresponding variable at the desired period. Variables to which no conditions apply will simply

remain blank.

As you can see, at any period there can be several simultaneous conditions. There can actually be any number of conditions for any given period, ranging from no condition at all to one condition for each variable. The only constraint is that there should not be a condition on each variable at each forecast period. This would cause the code to fail.

You may also notice that the number of periods included into the sheet is larger than the forecast window. This is not a problem as the code only selects periods which enter your forecast periods. As for unconditional forecasts, there is no issue either in starting your forecast prior to the end of the sample. The same logic holds for the number of variables comprised in your sheets: you may well include into this sheet all the variables of your dataset, and then run a conditional forecast exercise for a model which only comprises a subset of them. Here again, the code would select only the relevant variables. For instance, the above table could be used for a model only comprising "rgdp" and "un" as endogenous variables.

Note finally that you cannot use the conditional forecast methodology to implement conditions on your exogenous variables. To do so, you may however use the predicted exogenous feature of the toolbox (see section 4.2 for details).

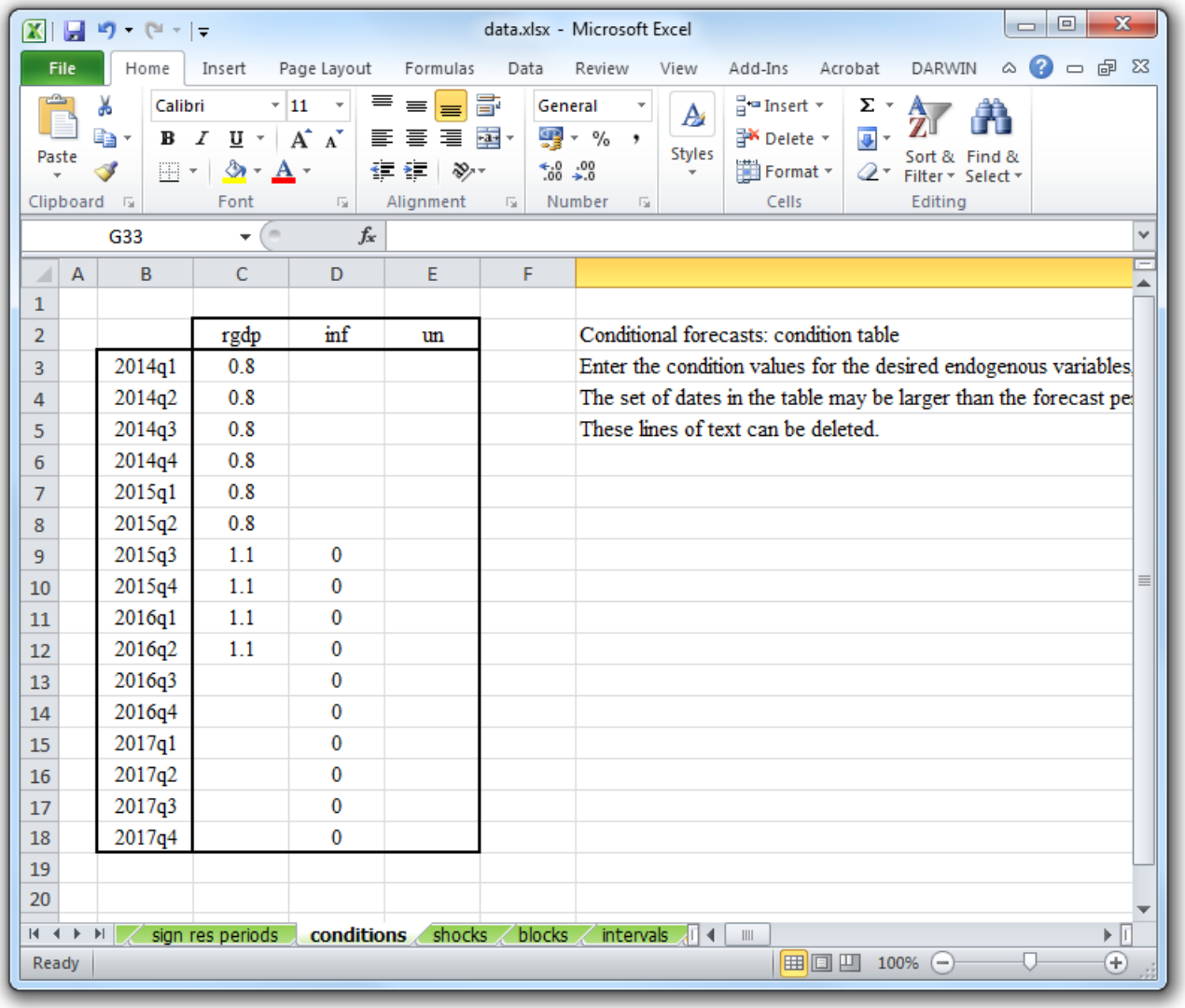
⚠ **Warning** The "all shocks" methodology should not be interpreted as a "neutral" or "agnostic" view on conditional forecasts. The fact that you are including all the shocks to generate the forecasts does not mean that it is left to the model to determine what will be the respective contributions of each shock in the forecast. On the contrary, using the all shocks methodology implies that you are certain that every shocks will effectively contribute to generate the conditions, which represents a very strong and informative assumption. Blindly using this methodology may lead to irrelevant results. Always verify that it reasonable to assume that all the shocks are generating your conditions, and always verify that your results make sense. In most cases, it may be preferable to use the shock-specific methodology.

4.6.2. Standard methodology (shock-specific)

While the "all shocks" approach proves simple and appealing, it is often too restrictive. One may indeed want to run conditional forecast exercises where only a subset of shocks generate the specified conditions. For instance, to run a conditional forecast exercise where the monetary authorities raise the interest rate to 3%, one may typically want this condition to be generated only by monetary policy shocks, and not by real GDP or unemployment shocks. In this case, the "shock-specific" approach should be adopted.

While the BEAR toolbox intends to make this approach as easy as possible to use, it remains more sophisticated than the "all shocks" approach and will probably require some extra effort to be properly used. To implement the shock-specific methodology, you will need three Excel worksheets: the green worksheet "conditions" that was already used for the all shocks methodology, and the additional "shocks" and "blocks" sheets. The "conditions" sheet is shown again for convenience:

Figure 4.11.: *Excel sheet: conditions*



	rgdp	inf	un
2014q1	0.8		
2014q2	0.8		
2014q3	0.8		
2014q4	0.8		
2015q1	0.8		
2015q2	0.8		
2015q3	1.1	0	
2015q4	1.1	0	
2016q1	1.1	0	
2016q2	1.1	0	
2016q3		0	
2016q4		0	
2017q1		0	
2017q2		0	
2017q3		0	
2017q4		0	

The other two sheets look like this:

Figure 4.12.: *Excel sheet: shocks*

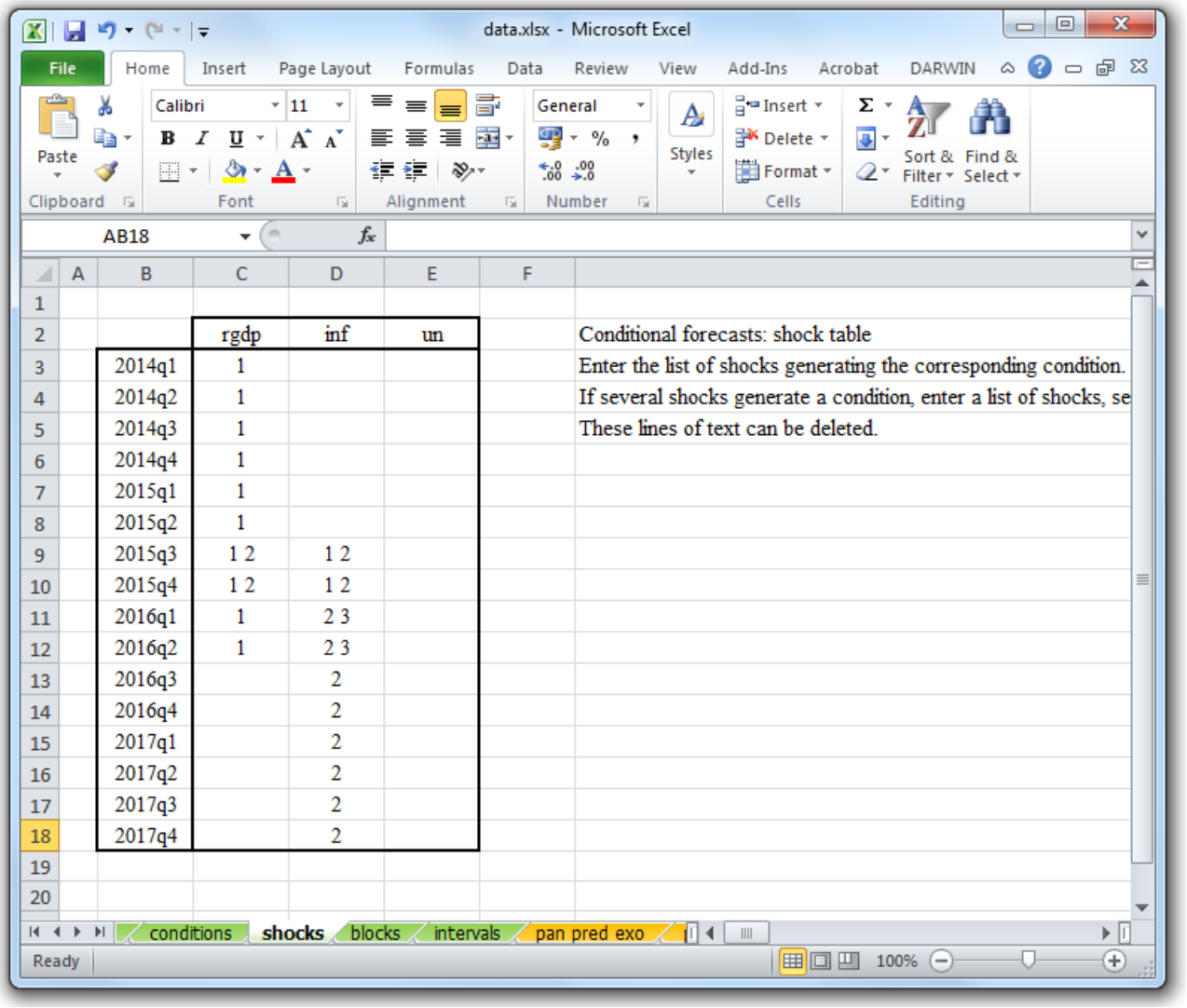
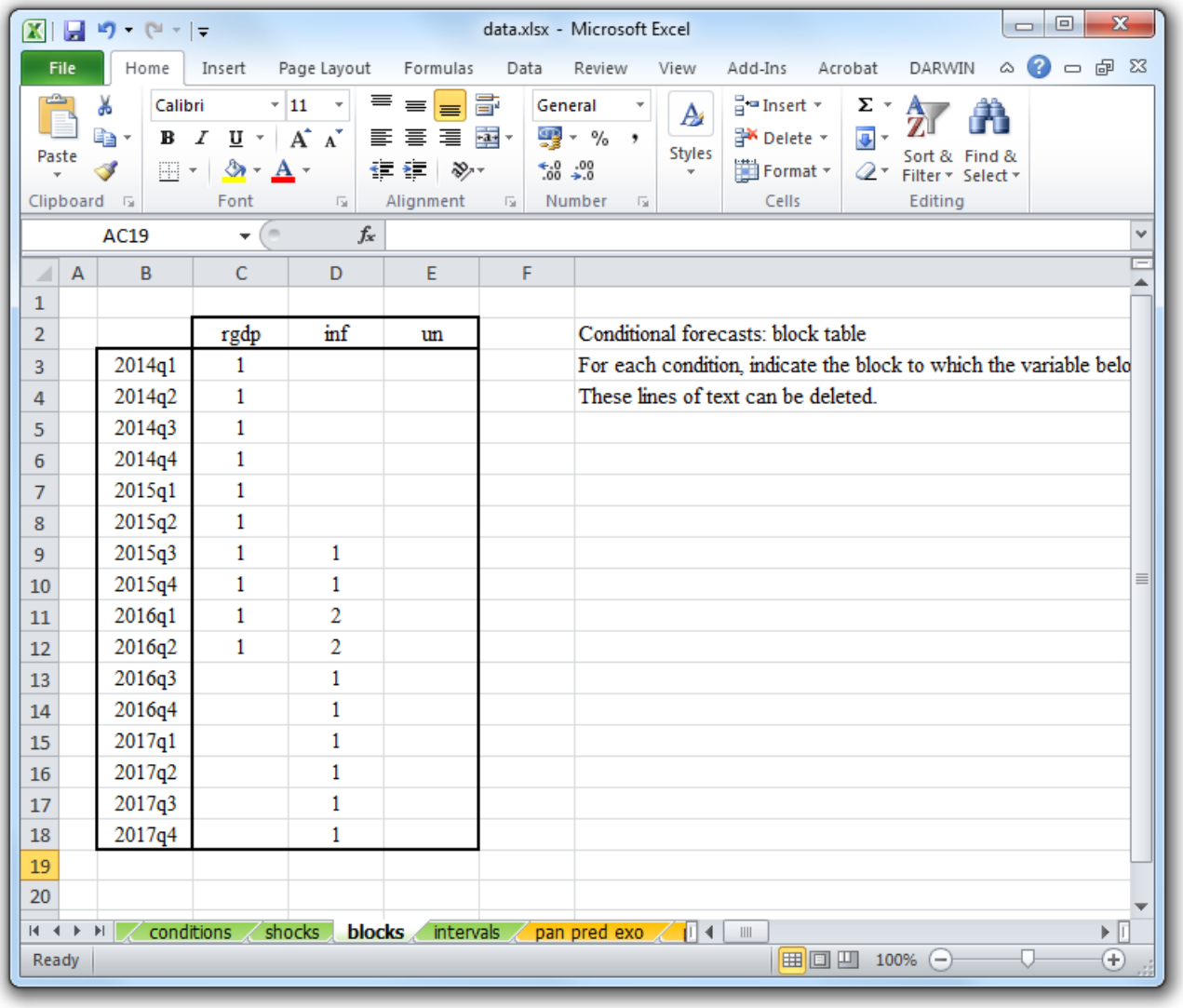


Figure 4.13.: *Excel sheet: blocks*



The first step to use the shock-specific methodology consists in specifying your conditions by the way of the "conditions" sheet. This is done similarly to the "all shocks" methodology, and is hence not developed further (please refer to subsection 4.6.1 in case your need more details).

The second step consists in specifying the shocks generating the different conditions. To do so, simply enter in the table of the "shocks" sheet a list of shocks for each corresponding condition. If several shocks are used to generate a condition, use a space to separate each shock in the list. To make this point more concrete, consider the "shocks" sheet for the forecast periods 2015q1-2016q4, and consider first the variable rgdp. The above example is designed so that the conditions on rgdp are generated by the first shock over 2015q1-2015q2, then jointly by shocks 1 and 2 for 2015q3-2015q4, then again by shock 1 only for 2016q1-2016q2. Considering now the variable inf, its conditions are generated by shocks 1 and 2 jointly for 2015q3-2015q4, then by shocks 2 and 3 for 2016q1-2016q2,

then only by shock 2 for 2016q3-2016q4. As you can see, a condition can indifferently be generated by a single shock or by several shocks, possibly even all the shocks in your model. Also, the conditions applying to a variable need not to be generated by the same shocks for different periods, as is hopefully clear from the example. Finally, if at a given period conditions apply to several variables, those conditions can be generated either by the same shocks (as for 2015q3-2015q4), or by mutually exclusive shocks (as for 2016q1-2016q2).

The final worksheet that you will have to fill is the "blocks" sheet. To understand the implications of this sheet, it is useful to state first some definitions. For a given period, a block of variable is a set of variables on which a condition is defined, and for which the conditions are generated by the same shocks. For a given period, a shock is said to be constructive for a block if is used to generate the conditions of the variables of this block. A shock which is not constructive for any block is said to be non-constructive.

To make this more concrete, consider again the above example, period by period. The forecasts start in 2015q1. For this period, there is only one condition set on the variable rgdp, and it is generated by shock 1. Because rgdp is the only variable with a condition on it, it constitutes the only block for this period and is defined as block 1. The shock used for the condition (shock 1) is thus a constructive shock for block 1. Shock 2 and 3 don't generate any condition and are thus non-constructive shocks for this period. The same holds for 2015q2.

In 2015q3, there are now two conditions: one on rgdp, and one on inf. Note that these two conditions are generated by the same shocks: both are generated by shocks 1 and 2, they thus constitute a single block which is block 1. Because shocks 1 and 2 are used to generate the conditions, they are constructive shocks for block 1. Shock 3 is not used to generate any condition at this period and is thus non-constructive. The same holds for 2015q4.

In 2016q1, there are again two conditions: one on rgdp, and one on inf. Now however they are generated by different shocks: the condition on rgdp is generated by shock 1, while the condition on inf is generated by shocks 2 and 3. rgdp and inf thus constitute different blocks. rgdp is here defined as block 1 and inf as block 2, but, as will become clear shortly, the order could have been inverted. Shock 1 is thus constructive for block 1, and shocks 2 and 3 are constructive for block 2. There is no non-constrictive shock for this period. The same holds for 2016q2.

In 2016q3, there is only one condition on inf: it constitutes the only block for the period (block 1), and the constructive shock for this block is shock 2. Shocks 1 and 3 are thus non-constructive. The same holds for 2016q4.

You can see that the "blocks" sheet contains the information that was established over the example. For a given period and each variable with a condition on it, simply type the number of the block to which it belongs in the corresponding cell. It is important to understand that the definition of the different blocks for a given period is left to the user: it is up to you to decide which block will be block 1, block 2, and so on. For instance in the above example, considering period 2016q1, it would have been possible to define inf as block 1 and rgdp as block 2.

You may wonder what can be the possible consequences of choosing a given ordering for the blocks rather than another. The answer is related to the functioning of the algorithm drawing the shocks for the conditional forecasts. The algorithm works as follows: for each forecast period, the algorithm first draws the non-constructive shocks from their unrestricted distribution; then it draws the constructive shocks for block 1, conditional on the non-constructive shocks; then it draws the constructive shocks for block 2, conditional on the non-constructive shocks and the constructive shocks for block 1, and so on. The ordering of the different blocks hence matters as it determines conditional on which other shocks the shocks of a given block will have to be drawn. This may affect the final results. For this reason, it is recommended that you define the ordering of your blocks by economic importance of the variables constituting them: variables that in your opinion dominate the economic activity should come first in the block ordering, variables of second importance should enter the ordering last.

⚠ **Warning** It should be emphasized that the definition of blocks is a period-by-period exercise, so do not blindly replicate a block number for a given variable over several forecast periods, or you may well trigger an error.

⚠ **Warning** Ideally, the shock-specific application would work smoothly for any set of conditions, shocks and blocks. It turns out that this is not always the case. The above example would work, but not every setting will. This is related to technical aspects of the estimation involving the structural identification scheme and its impacts on the different blocks of the model. For very detailed information, you are highly encouraged to read the technical guide along with the appendix. For more basic information about what will cause a conditional forecast setting to succeed or fail, a few guidelines are now provided, without justification.

Necessary conditions If these conditions are not met, conditional forecasts identification will always fail. These conditions however do not guarantee that the scheme will work, but only that it will not work if they are not satisfied.

- a block should have at least as many shocks as variables.
  example: a block with rgdp and inf for which the condition is generated by rgdp shock only will fail. The same block with rgdp and inf as constructive shocks satisfies the condition.

- different blocks should be generated by different shocks.
  example: block 1 made of rgdp with rgdp as constructive shock, and block 2 made of inf and un with rgdp and inf as constructive shock will fail. The same setting with block 2 having inf and un as constructive shocks satisfies the conditions.

- if the identification scheme is Choleski or triangular factorisation, the conditions for a given block should be generated only by shocks of their own variables, or by shocks of variables preceding them in the variable ordering.
  example: assume the ordering of the variables is: rgdp, inf and un, plus other additional variables in the model coming later on the ordering. The structural identification is Choleski. Assume that block 1 is made of inf and un (variables 2 and 3). If their constructive shocks are shocks 5 and 6, the setting will fail. If the constructive shocks are 3 and 4, it will fail as well: no shock at all should be posterior. If the constructive shocks are 2 and 3, the setting will work, and so will it if the constructive shocks are 1 and 2, or 1 and 3.

Sufficient conditions If these conditions hold, the setting will always produce well identified conditional forecasts. If you are uncertain of what you are doing, it is recommended that you follow these conditions.

- the structural identification scheme is Choleski or triangular factorisation. There is only one block, for which the conditions are generated by their own shocks.
  Example: there is only one block, which is made of rgdp and un, and the constructive shocks are 1 and 3.

- the structural identification scheme is sign restrictions, and there is only one block.
  Example: the only block is made of inf and un.

- the structural identification scheme is Choleski or triangular factorisation. There are several blocks, but for all blocks, the constructive shocks are that of the variables included in the block. The block are defined from left to right with respect to the variable ordering.
  Example: the structural identification scheme is Choleski. There are two blocks. One is made of rgdp and the constructive shock is 1, the other block is made of inf and un and the constructive shocks are 2 and 3. Then if rgdp constitutes block 1 and inf and un constitute block 2, conditional forecasts are always identified.

4.6.3. Tilting methodology (median)

An alternative to the standard methodology for conditional forecasts is the so-called tilting methodology. Unlike the standard methodology, the tilting approach does not rely on structural shocks to

generate the condition and is thus agnostic in terms of economic theory. This represents a considerable asset if you have no preconceived idea about the origin of the condition. Another advantage of the tilting methodology over the standard approach is that it is a "soft" forecast methodology: while the conditions with the standard methodology always hold exactly, the tilting methodology allows for some variability around your conditions. Even better, if you use the "interval" methodology, you can specify yourself the variability around the conditions. Tight bands can thus be used to reflect a high level of confidence around the conditions, while loose bands reflect larger uncertainty.

The idea of the tilting methodology is that a conditional forecast could be interpreted as a normal forecast to which new information (the conditions) would have been integrated. The tilting procedure thus consists in starting from the posterior distribution of the unconditional forecast, then in modifying this distribution to satisfy the conditions while keeping it as close as possible to the original distribution. For this reason, the tilting methodology requires the unconditional forecast application and will thus be deactivated if unconditional forecasts are not selected.

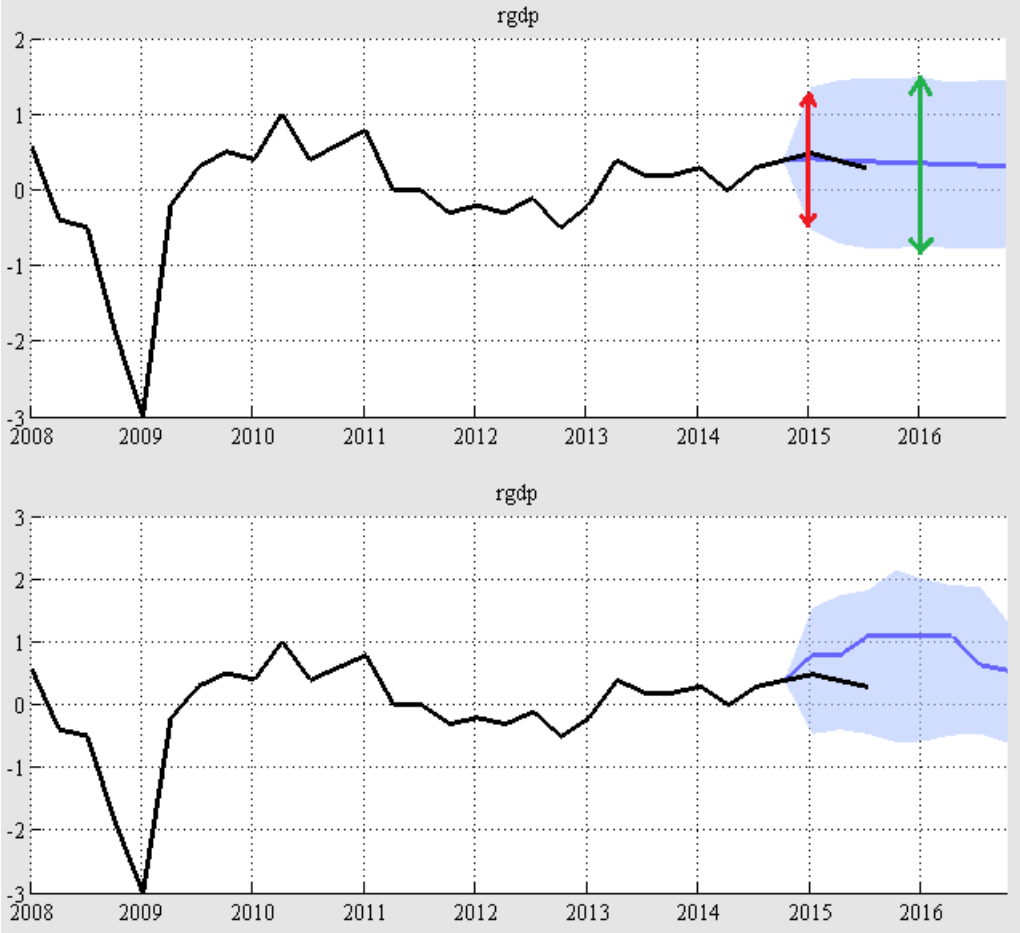
The BEAR toolbox implements the tilting application by defining quantiles of the posterior distribution. In the basic form of the application, the only quantile that you will define is the median of the posterior distribution, that is, the values for your conditions. To do so, you only need to specify your conditions values in the green Excel worksheet "conditions". The procedure is similar to that of the standard methodology (all shocks) and is not developed further: please refer to subsection 4.6.1 for details. No other sheet is required.

The tilting methodology is appealing in many aspects, but it also has its own limitations, which are mainly related to the technical aspects of the procedure implemented to compute the conditional forecasts.

The first limitation has to do with the support of the forecast distribution. As mentioned previously, the tilting procedure starts from the posterior distribution of the unconditional forecasts, and then modifies this distribution to match the conditions. While the procedure can reshape the distribution, it cannot change its support, that is, the range of possible values covered by the distribution. For this reason, you cannot implement as conditions values which do not appear in the support of the unconditional distribution. Roughly speaking, it means that your conditions must be "close enough" to the unconditional values for the algorithm to work. They must in any case be comprised in the support, and for a safer procedure be actually fairly close to the point estimate. To make this point more concrete, let us consider the usual example model with rgdp, inf and un as variables. The following plot shows the forecasts obtained for rgdp over the forecast period 2015q1-2016q4 with a Minnesota prior:

The top panel represents the unconditional forecasts, while the lower panel represents the conditional forecasts obtained from the tilting methodology. Look first at the top panel, and consider the first forecast period 2015q1: the point estimate is around 0.4, the lower bound of the 95% credibility interval is somewhere around -0.5, and the upper bound of about 1.1 (see the red arrow). Because the size of the credibility interval is 95%, it is reasonable to assume that values outside the interval are essentially extreme values that we may ignore. Therefore, the red arrow gives a good representation

Figure 4.14.: *Tilting example*



of the support of the posterior distribution for 2015q1. In order to use the tilting methodology to obtain conditional forecast for this period, the specified condition should be comprised between -0.5 and 1.1, and ideally closer to the point estimate value of 0.4. In the example Excel file the condition for this period is set at 0.8, which constitutes a suitable candidate. As can been seen from the lower panel, the tilting methodology successfully implemented the condition.

For the subsequent forecast periods, the unconditional forecasts become pretty much stable with point estimates at about 0.3, lower bound at -0.8 and upper bound at about 1.2 (see for instance the green arrow for 2016q1). Because the support is getting a bit larger, you can choose conditions which are located a bit further away from the point estimate. The example sheet shows that over the period 2015q3-2016q2 the conditions values were increased to 1.1, and the lower panel of the plot shows that tilting was successful to generate these conditions. The value however was fairly close to the upper bound of the credibility interval of the unconditional forecasts, and this is probably as far as the methodology would be able to go.

In general, if you plan to use the tilting methodology, your methodology should thus go as follows: estimate first your model, select the unconditional forecasts but not (yet) the conditional forecasts. Use the plots of the unconditional forecasts to establish roughly what the support of the unconditional forecast distribution is, for each variables and each period of interest. Establish then a set of conditions which are compatible with these values. Eventually estimate the conditional forecast with the tilting methodology.

Beyond this first limitation, there exists second limitation related to the number of conditions that the tilting methodology is able to handle. To obtain the conditional forecasts, the algorithm implements a numerical optimisation procedure, and this optimisation step will typically fail if there are too many conditions. Concretely, this will result in the code producing conditional forecasts where the conditions do not actually hold. In general, a dozen restrictions is pretty much the maximum that the methodology can handle. In the above example there are a total of 12 restrictions over the whole forecast periods and all the endogenous variables. As the lower panel shows, the code managed to produce successful conditional forecasts for rgdp (the conditional forecasts for inf are not displayed but were also correctly identified). This is probably the maximum number of conditions that the methodology would support.

A final technical remark applies, which is not a limitation of the procedure but rather a methodological precision: in order to obtain accurate results, the tilting procedure typically requires more iterations than the standard methodology. While for the standard methodology 1000 post-burn iterations are usually sufficient, tilting would rather requires around 5000 post-burn iterations. With a lower number of iterations, your results may look imprecise. This implies that the tilting methodology will be significantly more time-consuming than the standard methodology.

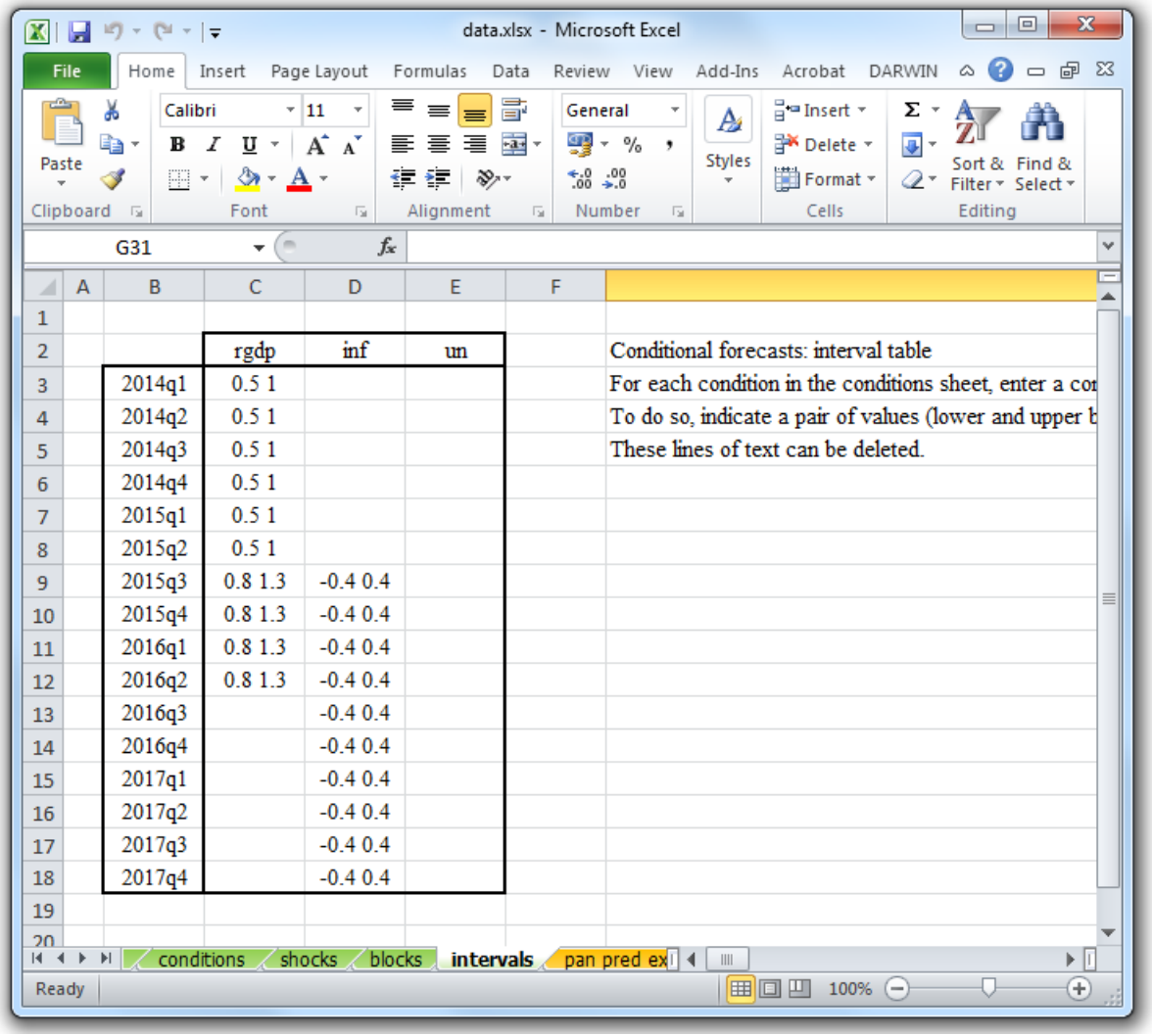
4.6.4. Tilting methodology (intervals)

In its advanced form, the tilting methodology allows you to define not only the median (point estimate) of the conditional forecasts, but also the upper and lower bounds of the credibility intervals around the median value. This way, the methodology can be used to reflect the degree of confidence

you have for your conditions: loose bands around the conditions reflect low confidence, tight bands reflect higher confidence.

To use the intervals methodology, you will need to use two Excel sheets. The first is the "conditions" sheet. On this sheet, you have to enter your conditions: the procedure is once again similar to the standard methodology (all shocks) and is not developed further: please refer to subsection 4.6.1 for details. The second sheet is the green sheet "intervals". It looks like this:

Figure 4.15.: *Excel sheet: intervals*



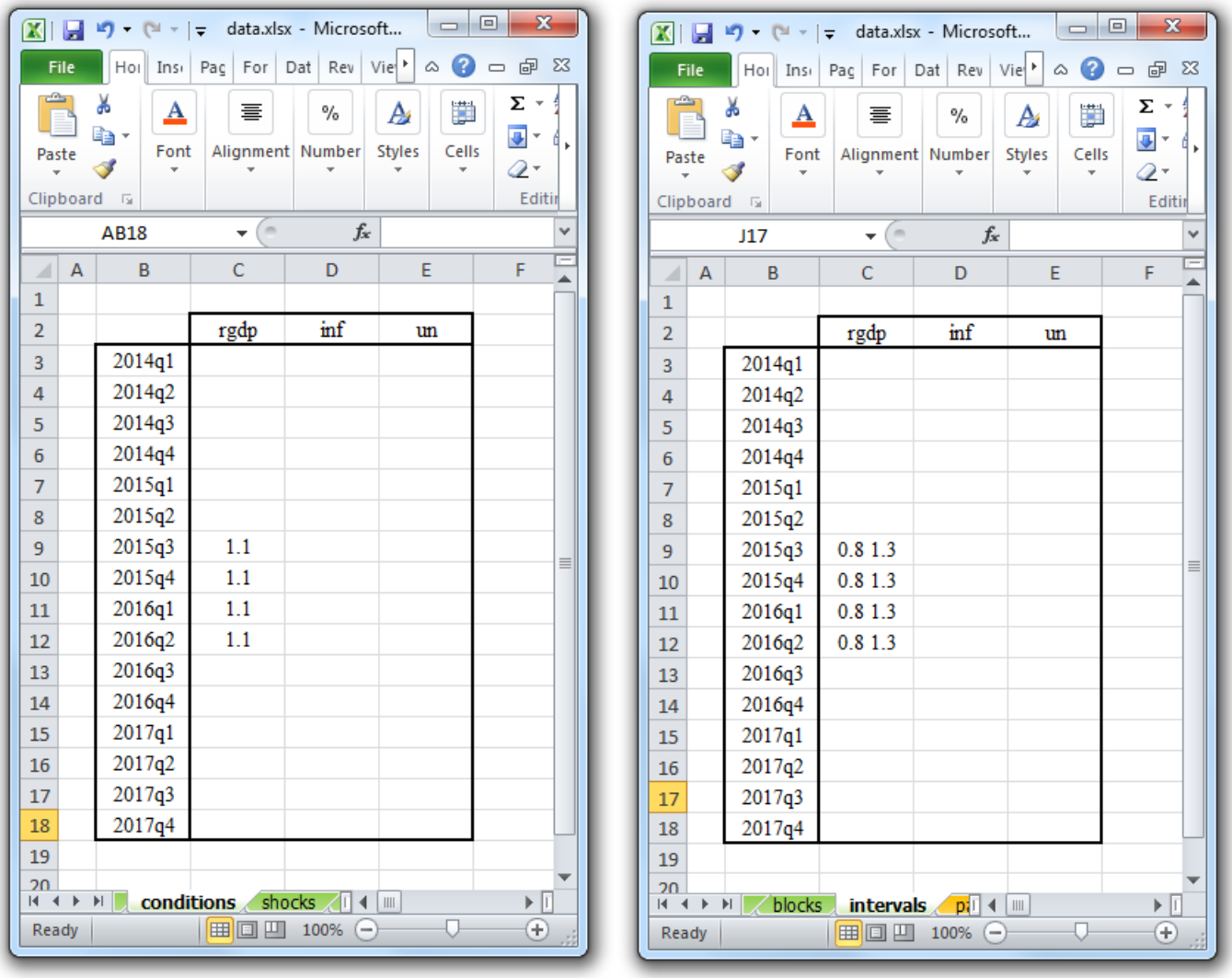
	rgdp	inf	un
2014q1	0.5 1		
2014q2	0.5 1		
2014q3	0.5 1		
2014q4	0.5 1		
2015q1	0.5 1		
2015q2	0.5 1		
2015q3	0.8 1.3	-0.4 0.4	
2015q4	0.8 1.3	-0.4 0.4	
2016q1	0.8 1.3	-0.4 0.4	
2016q2	0.8 1.3	-0.4 0.4	
2016q3		-0.4 0.4	
2016q4		-0.4 0.4	
2017q1		-0.4 0.4	
2017q2		-0.4 0.4	
2017q3		-0.4 0.4	
2017q4		-0.4 0.4	

The correspondence with the "conditions" sheet is straightforward: to each condition in the "conditions" sheet corresponds an interval in the "intervals" sheet. Intervals have to be specified as two numbers separated by a space, the first number representing the lower bound, the second number representing the upper bound. The median specified in "conditions" has to be comprised these two values. On the example sheet similar conditions are associated with similar intervals, but this is not mandatory.

The technical limitations applying to the median case also apply to the intervals methodology. First, the values specified for the bounds of the intervals should not be too far away from the bounds of the unconditional forecasts (in any direction). Second, the total number of restrictions should still be restricted to a dozen. In the intervals setting, each element (median, lower bound and upper bound) counts as one condition. Therefore each triplet median/lower bound/upper bound creates 3 restrictions, meaning that 4-5 triplets of restrictions is the maximum you can afford for the code to

work properly. In the above example for instance, with the forecast dates set to 2015q1-2016q4, the Excel spreadsheets imply 36 conditions. This tends to be too many for the code to succeed and the algorithm would fail. To run the exercise successfully, one possibility would be either to leave the sheet as it is but reduce the forecast window (for instance set it to 2015q1-2015q3 to retain only 12 conditions), or to keep the forecast windows as it is but clear conditions from the sheet until only a dozen or so remains. For instance, if one modifies the "conditions" and "intervals" sheets as follows:

Figure 4.16.: *Excel sheet: modified conditions and intervals*



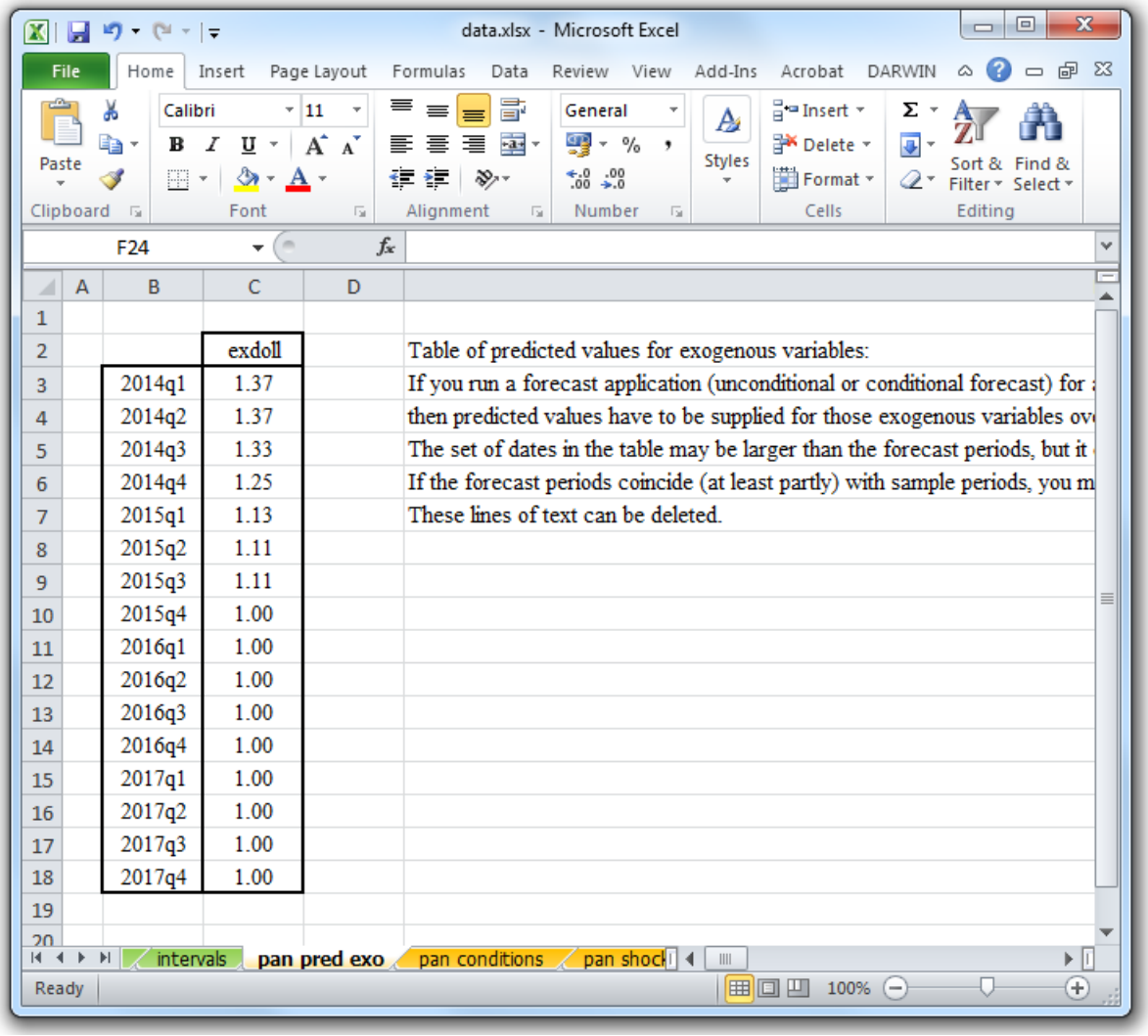
Then the number of restrictions becomes 12 over the 2015q1-2016q4 period, and the exercise can be successfully run, producing the following results for rgdp:

Figure 4.17.: *conditional forecast with tilting (median and intervals)*

4.7. Predicted exogenous (panel BVAR)

If you want to produce forecasts for a panel BVAR model and if your model includes any exogenous variable(s) other than the constant, you will need to provide predicted values for these variables over your forecast windows. To do so, open the Excel file "data.xlsx", and select the orange worksheet "pan pred exo". The table looks like this:

Figure 4.18.: *Excel sheet/ predicted exogenous (panel)*



The procedure is similar to that of the standard BVAR model, and is hence not developed further: please refer to section 4.2 for details. Note nevertheless that in the above sheet, the only variable which is present is "exdoll": this is because unlike a standard BVAR model, a panel VAR requires you to plan in advance which variable will be endogenous and which variable will be exogenous (you may refer to section 2.4 for details). Hence the sheet here only includes the single planned exogenous variable, and omits the endogenous variables. Note that including the endogenous variables would be harmless: the code would only select the relevant exogenous variables. It is however good practice to restrict your sheet to the set of planned exogenous in order to avoid confusion.

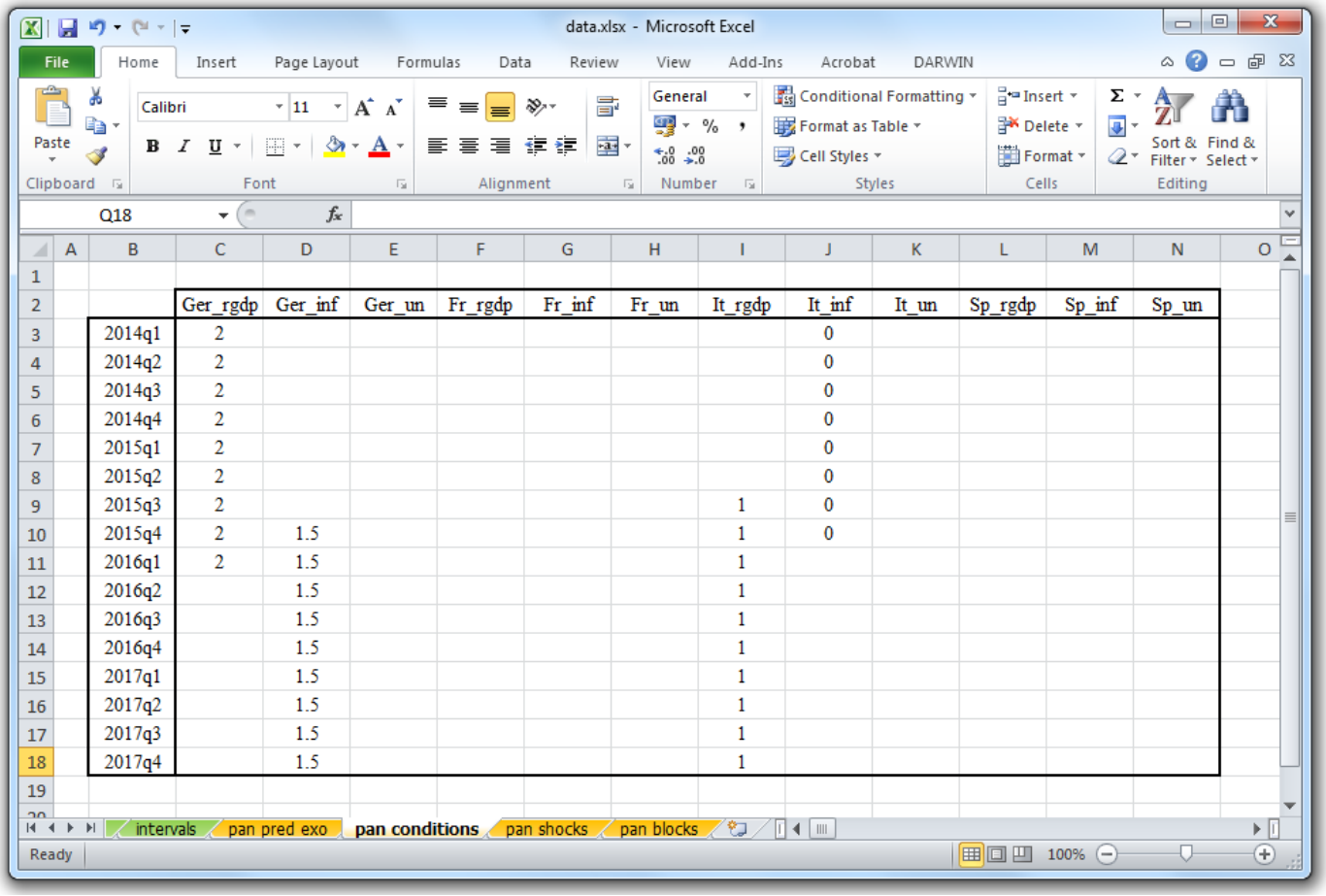
4.8. Conditional forecasts (panel BVAR)

The BEAR toolbox allows you to use conditional forecasts with all Bayesian panel VAR models, but this excludes the OLS mean-group estimator model. Note that the panel framework only allows you to use the standard methodology, either "all shocks" or "shock-specific". The procedure to follow depends on your choice of methodology.

4.8.1. Standard methodology (all shocks)

If you choose the "all shocks" methodology, you will only have to specify your conditions on Excel. To do so, open the Excel file "data.xlsx", and select the orange worksheet "pan conditions". It looks like this:

Figure 4.19.: *Excel sheet: conditions (panel)*



	Ger_rgdp	Ger_inf	Ger_un	Fr_rgdp	Fr_inf	Fr_un	It_rgdp	It_inf	It_un	Sp_rgdp	Sp_inf	Sp_un
2014q1	2							0				
2014q2	2							0				
2014q3	2							0				
2014q4	2							0				
2015q1	2							0				
2015q2	2							0				
2015q3	2						1	0				
2015q4	2	1.5					1	0				
2016q1	2	1.5					1					
2016q2		1.5					1					
2016q3		1.5					1					
2016q4		1.5					1					
2017q1		1.5					1					
2017q2		1.5					1					
2017q3		1.5					1					
2017q4		1.5					1					

The principle is the same as for a traditional Bayesian VAR model. The rows contain the dates: the number of periods you are including can be larger than your forecast window but must at least include it fully. The columns correspond to the endogenous variables of your models. There is a

slight adaption compared to the standard Bayesian VAR model: because a panel models replicates the same endogenous variables over different units, you have to specify to which unit relates to each occurrence of your endogenous. For this reason, the label of each column has to be specified as: "unit name" then "_" (without space before or after) and finally "endogenous name". For instance to specify the endogenous variable rgdp for the unit It, the label must be entered as "It_rgdp". Any other format will not be recognised by the code and return an error. You may as usual fill the table with more endogenous variables and units than used in your model, the code is designed to identify and select only the relevant components.

To implement a condition, simply enter a condition value for the corresponding date, unit and variables. Cells left blank imply no condition. Note that there exists a difference of interpretation of the table depending on the type of panel model you are estimating. If you are using the Bayesian pooled estimator, or the random effect models (Zellner and Hong or hierarchical), you end up with one VAR model applied separately to every unit. This means that the conditions you are specifying will be considered separately from one unit to another, so that the conditions for one unit will have no impact on the conditional forecasts of the other units. If for some units you don't specify any conditions, the code will simply ignore this unit for the conditional forecast exercise. On the other hand, if you use a factor model (static or dynamic), you will end up with a single large VAR in which all units are interacting. This means that the conditions you specify for any unit will affect all the other units, even if no condition is specified on them. While this difference has no implications in the way the conditions are set on Excel, you should be aware of it as it will significantly affect your results.

To make this point more concrete, consider an example applying to the above spreadsheet: a panel BVAR involving three units: Ger, It and Sp. Each unit involves the three endogenous variables rgdp, inf and un. Note first that you can use the spreadsheet as it is for this model: even though the unit Fr does not enter into the model, this does not create any issue: the code will ignore the columns for this unit. Let us assume first that you estimate a random effect model (Zellner and Hong). With this methodology, a separate VAR is estimated for each unit. This means that the estimation process will result in three independent VAR models for Ger, It and Sp, so that the conditional forecasts setting has to be considered for the three units separately. For the unit Ger, there are conditions on rgdp and inf. These conditions will have an impact on the other variable of Ger which is the variable un. However, they will not have any effect on the other two units It and Sp. The case is similar for the conditions on It. As for the unit Sp, it does not have any condition: the code will hence skip it in the conditional forecast exercise. Let us now assume that a static factor model is estimated. With this methodology, a single VAR model is estimated comprising all the variables of all the units. In the above example, one obtains a large VAR model with 9 variables (the three endogenous variables rgdp, inf and un for the three units Ger, It and Sp). The units cannot be considered separately anymore: the conditions on rgdp for Ger for instance will now affect all the other variables in the VAR, even those of the unit Sp on which no conditions are defined.

4.8.2. Standard methodology (shock-specific)

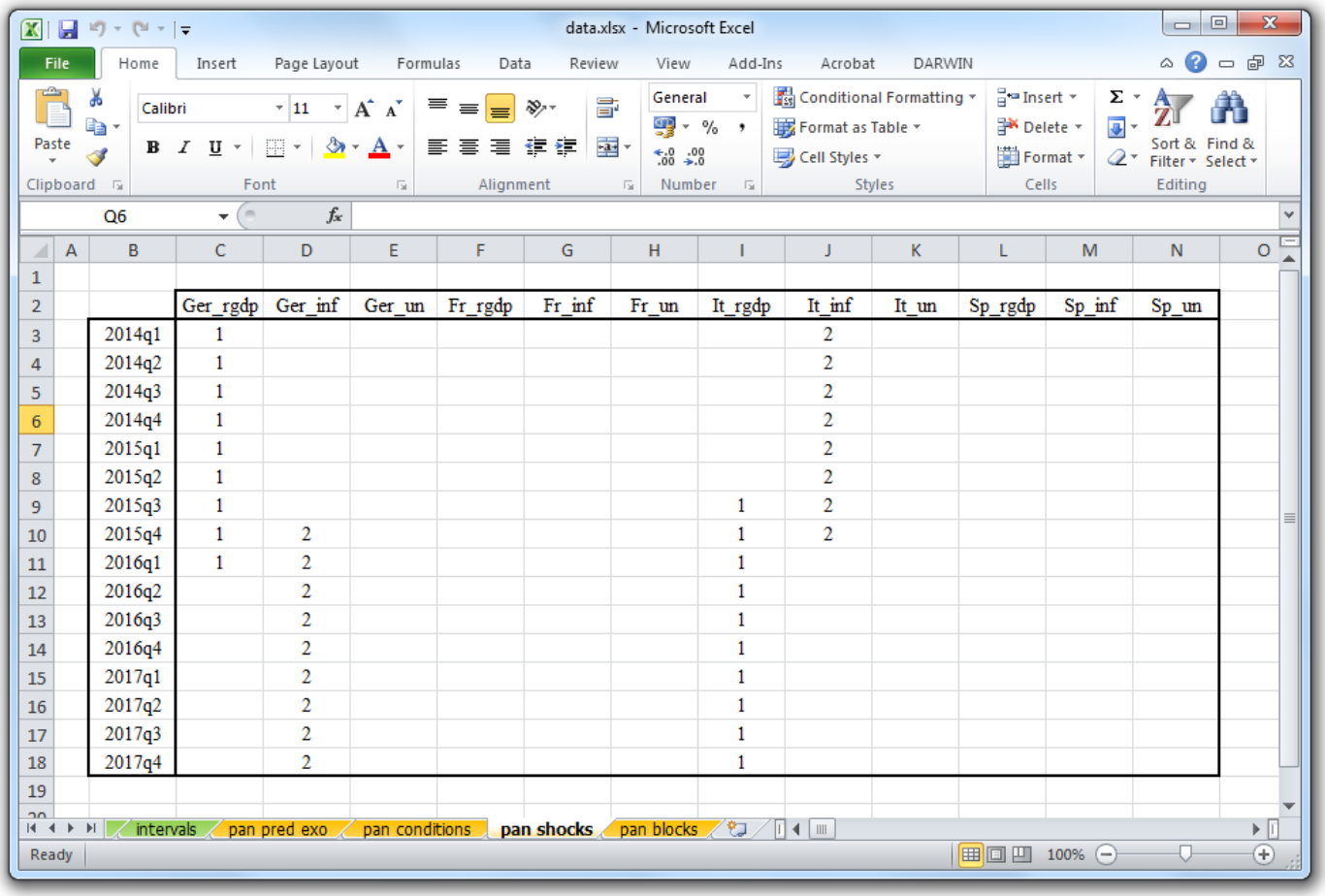
The panel BVAR framework also lets use the shock-specific methodology for your conditional forecasts. The methodology is similar to that of the standard Bayesian VAR: you need to specify the conditions, shocks and blocks for your conditional forecast exercise. To do so, three Excel worksheets

are used: the "pan conditions" orange worksheet, already used for the "all shocks" methodology, the "pan shocks" orange sheet, used to input your shocks, and the "pan blocks" orange sheet, used to define your blocks. The definition of conditions, shocks and blocks is similar to that of the standard BVAR and is not developed further: please refer to subsection 4.6.1 and subsection 4.6.2 for details.

A point should be emphasized: the way you fill the "pan shocks" and "pan blocks" sheets will differ, depending on whether your choice of model results in separate VAR models (Bayesian pooled estimator, random effect models) or in a single large VAR model where all the units interract(static or dynamic factor model). Hence, unlike the "all shocks" methodology for which the distinction only applied in terms of interpretation, here it will results in differences in the writing of your spreadsheets.

Assume first that you want to estimate a Bayesian pooled estimator, or any of the random effect models (Zellner and Hong or hierarchical). Consider again the case of a panel BVAR with the three units Ger, It and Sp, and for each unit the three endogenous rgdp, inf and un. There is no difference with the "all shocks" methodology in the way you specify your conditions, using the spreadsheet "pan conditions". Please refer to subsection 4.8.1 for details. To specify your shocks, you need to use the worksheet "pan shocks". It looks like this:

Figure 4.20.: *Excel sheet: shocks (panel)*



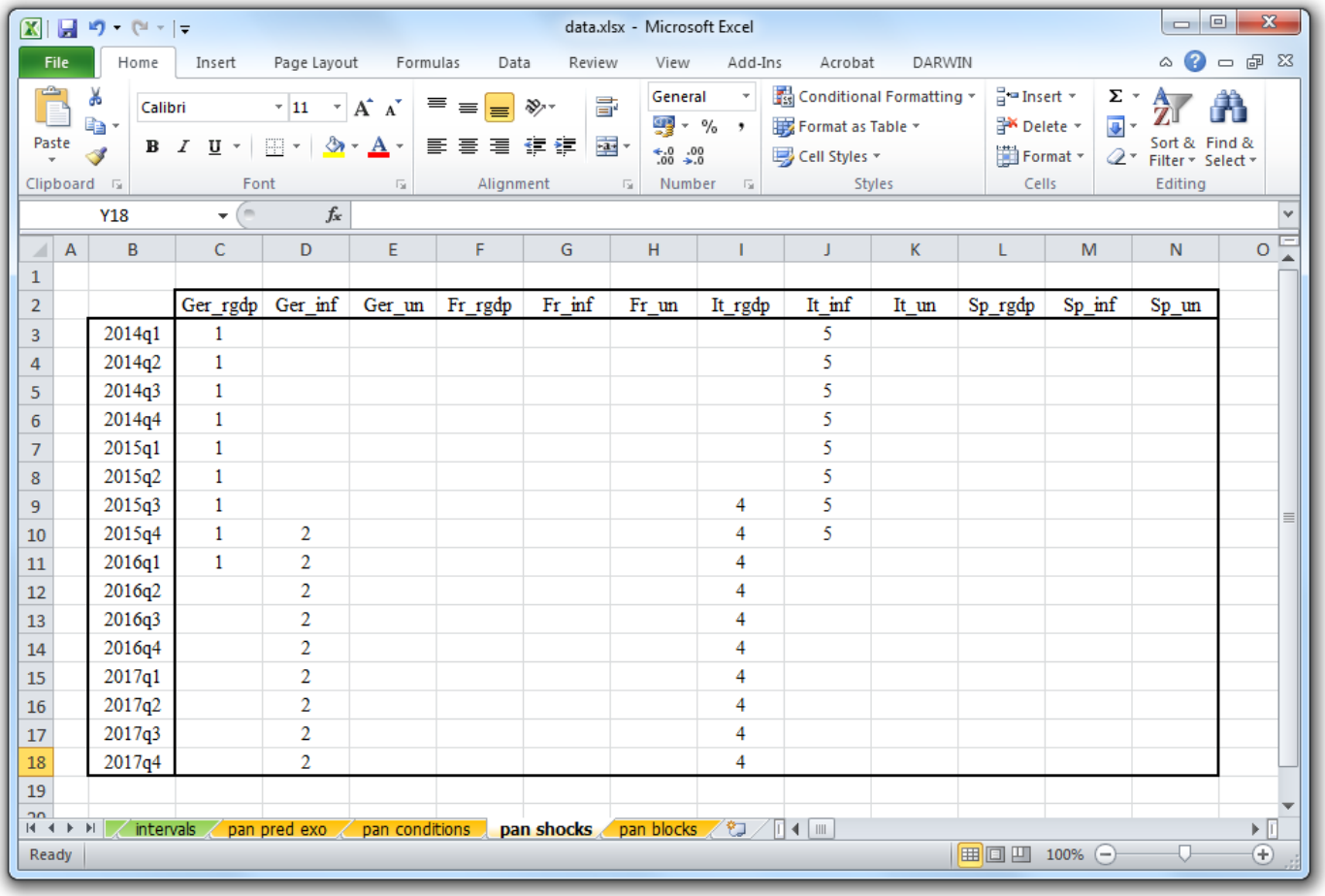
	Ger_rgdp	Ger_inf	Ger_un	Fr_rgdp	Fr_inf	Fr_un	It_rgdp	It_inf	It_un	Sp_rgdp	Sp_inf	Sp_un
2014q1	1							2				
2014q2	1							2				
2014q3	1							2				
2014q4	1							2				
2015q1	1							2				
2015q2	1							2				
2015q3	1						1	2				
2015q4	1	2					1	2				
2016q1	1	2					1					
2016q2		2					1					
2016q3		2					1					
2016q4		2					1					
2017q1		2					1					
2017q2		2					1					
2017q3		2					1					
2017q4		2					1					

The principle behind the specification of your sheet is the following: because the random effect model produces separate VAR models for each unit, you have to consider an independent model and hence an independent conditional forecast exercise for each of your units. In the considered example there are 3 units (Ger, It, and Sp) and 3 variables (rgdp, inf and un) considered separately for each unit. This implies that each unit is characterised by 3 structural shocks applying only to themsleves. In the above sheet for instance the conditions on Ger_inf are generated by shock 2: this actually means shock 2 of its own unit, that is, the unit Ger. The conditions on It_inf are also generated by shock 2: this has also to be interpreted as shock 2 of its own unit, which thus means shock 2 of unit It. Hence, also all the units are gathered in a single sheet, you really have to fill the sheet as if you were specifying separated VAR models.

Let us now assume you want to estimate any of the factor models (static or dynamic). With such a model you are now obtaining a single VAR model where all the units are interacting. Rather than considering your units separately, you need to consider them jointly. In the above example with Ger, It and Sp as units and rgdp, inf and un as variables, you obtain a large VAR with 9 variables: rgdp,

inf, and un for Ger, rgdp, inf and un for It, and rgdp, inf and un for Sp. This implies 9 corresponding structural shocks. The above table is then not valid anymore. Consider for instance the conditions on It_rgdp. The above table specified them as being generated by "shock 1", which actually designates the structural shock associated to the first variable (rgdp) of It, which in the new model is actually shock 4. To be consistent with the new model, the table should be rewritten as follows:

Figure 4.21.: *modified Excel shock sheet for factor model*



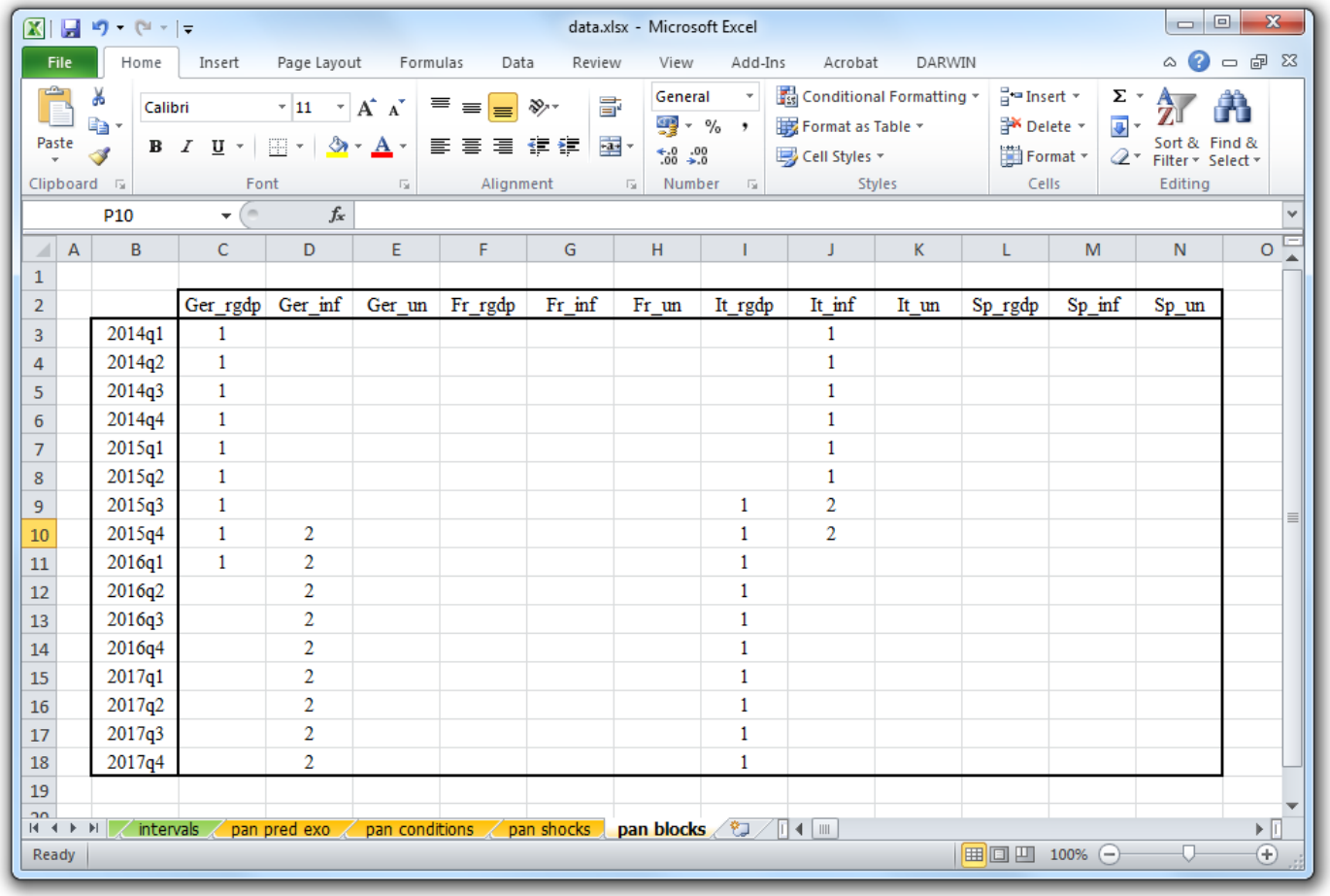
	Ger_rgdp	Ger_inf	Ger_un	Fr_rgdp	Fr_inf	Fr_un	It_rgdp	It_inf	It_un	Sp_rgdp	Sp_inf	Sp_un
2014q1	1							5				
2014q2	1							5				
2014q3	1							5				
2014q4	1							5				
2015q1	1							5				
2015q2	1							5				
2015q3	1						4	5				
2015q4	1	2					4	5				
2016q1	1	2					4					
2016q2		2					4					
2016q3		2					4					
2016q4		2					4					
2017q1		2					4					
2017q2		2					4					
2017q3		2					4					
2017q4		2					4					

Note that it is not mandatory for the conditions of a given unit to be generated by their own shocks. In the above example, it could be possible to specify the conditions on It_rgdp to be generated by the shocks on Sp_rgdp, for instance. In this case, the shock to specify would be 7 and not 4.

The same logic applies to blocks. The "pan blocks" worksheet looks like this:

As it is, the sheet is suitable for the Bayesian pooled estimator, or any of the random effect models (Zellner and Hong or hierarchical). It considers the units separately, assuming one VAR per unit. If you consider for instance the period 2015q4, the condition on Ger_rgdp constitutes block 1, while the one associated with Ger_inf constitutes block 2. The two blocks are considered from the point of

Figure 4.22.: *Excel sheet: blocks (panel)*

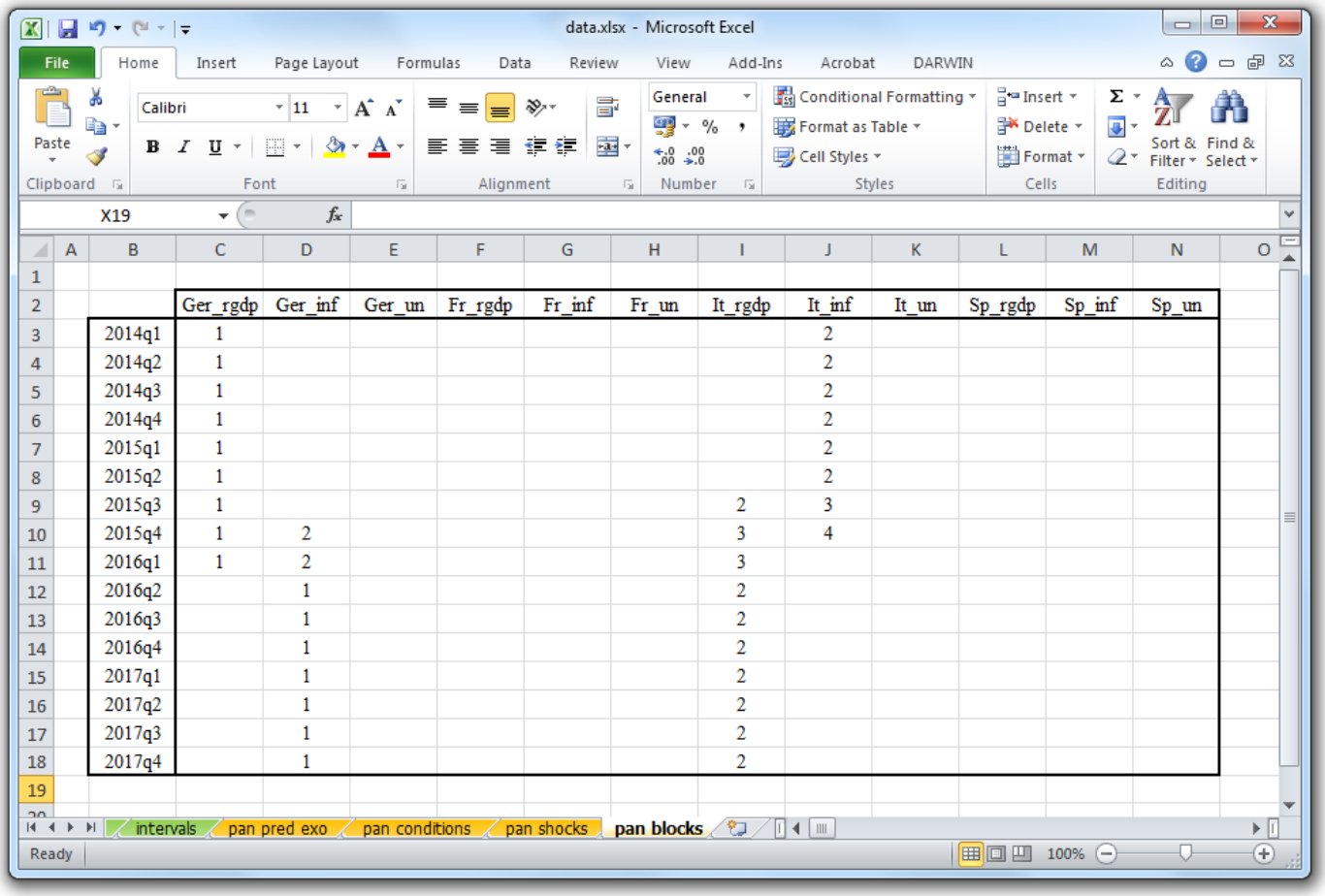


	Ger_rgdp	Ger_inf	Ger_un	Fr_rgdp	Fr_inf	Fr_un	It_rgdp	It_inf	It_un	Sp_rgdp	Sp_inf	Sp_un
2014q1	1							1				
2014q2	1							1				
2014q3	1							1				
2014q4	1							1				
2015q1	1							1				
2015q2	1							1				
2015q3	1						1	2				
2015q4	1	2					1	2				
2016q1	1	2					1					
2016q2		2					1					
2016q3		2					1					
2016q4		2					1					
2017q1		2					1					
2017q2		2					1					
2017q3		2					1					
2017q4		2					1					

view of a VAR estimated only for Ger. For the same period, the condition associated with It_rgdp and It_inf also represent respectively blocks 1 and 2, but from the point of view of a VAR estimated for It on its own.

This table would not be appropriate for a factor model (static or dynamic). For such models a single VAR is estimated for all the units, so that the blocks have to be considered jointly for the whole model. To be correct, the above table should be modified as follows:

Figure 4.23.: *modified Excel block sheet for factor model*



	Ger_rgdp	Ger_inf	Ger_un	Fr_rgdp	Fr_inf	Fr_un	It_rgdp	It_inf	It_un	Sp_rgdp	Sp_inf	Sp_un
2014q1	1							2				
2014q2	1							2				
2014q3	1							2				
2014q4	1							2				
2015q1	1							2				
2015q2	1							2				
2015q3	1						2	3				
2015q4	1	2					3	4				
2016q1	1	2					3					
2016q2		1					2					
2016q3		1					2					
2016q4		1					2					
2017q1		1					2					
2017q2		1					2					
2017q3		1					2					
2017q4		1					2					

The table may look complicated, but if you remember that the identification of blocks is a period-by-period exercise and hence proceed to the labelling of blocks row after row, you will see that it is easy to translate Figure 4.22 into this one. Consider for instance the period 2015q3. If you consider the model as whole, with the conditions defined by Figure 4.19 and the shocks defined as in Figure 4.20, then there three conditions for this period: one on Ger_rgdp, one on It_rgdp, one on It_inf, respectively generated by shocks 1, 4 and 5. As all the conditions are generated by different shocks, they constitute different blocks. The condition on Ger_rgdp thus constitutes block 1, that on It_rgdp constitutes block 2, and the one on It_inf constitutes block 3. Note that you may have defined the ordering of the blocks differently if you wished. The same reasoning could be repeated for each period of the sheet, and one would obtain Figure 4.23.

ï»¿

CHAPTER 5

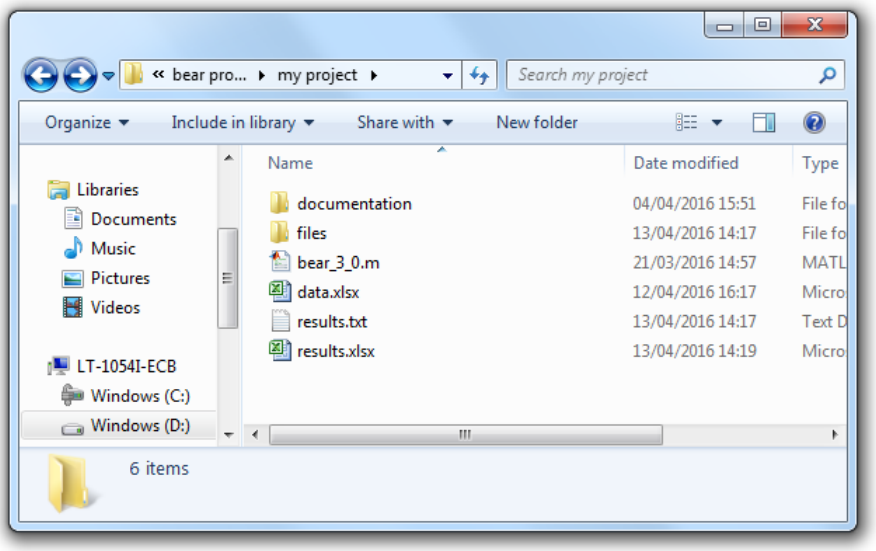
Interpreting your results

Once you run BEAR, it will produce 3 different outputs:

- a summary of your estimation results: the results will appear on the Matlab screen, but they will also be saved on the text file "results.txt".
- a set of plots generated by Matlab for your model.
- an Excel file "results.xlsx" storing the numerical values of the results for all your applications.

The files "results.txt" and "results.xlsx" are stored in the same folder as the one containing the Excel datafile "data.xlsx" (that is, in folder specified by your data path), which is typically your project folder:

Figure 5.1.: *Results files in project folder*



Warning The files "results.txt" and "results.xlsx" will be overwritten each time you run a new estimation. If you want to preserve your current results, you should rename the two files to avoid their replacements.

The different outputs are now introduced. The presentation is done for the Bayesian VAR model, but the other models produce similar outputs, with only slight variations.

5.1. Matlab output

The first set of results is provided on the Matlab window, with a copy created on the text file "results.txt". These results provide a summary of your estimation, and can be divided into 5 parts. The first part recapitulates your estimation settings, and hence allows you to replicate the estimation later on with exactly the same configuration:

Figure 5.2.: *First part of Matlab output*

```
BEAR toolbox estimates
Date: 13-Apr-2016   Time: 15:43


Bayesian VAR
structural decomposition: triangular factorisation
endogenous variables:  rgdp  inf  un
exogenous variables:  constant
estimation sample: 1999q1-2014q4
sample size (omitting initial conditions): 61
number of lags included in regression: 3
prior: normal-Wishart (sigma as univariate AR)
hyperparameters:
autoregressive coefficient (ar):                0.8
overall tightness (lambda1):                    0.1
lag decay (lambda3):                            1
exogenous variable tightness (lambda4):         100
```

The second part provides an estimate of your model coefficients β, equation by equation:

Figure 5.3.: *Second part of Matlab output*

```
VAR coefficients (beta): posterior estimates

Endogenous: rgdp
                              Median          St.dev       Low.bound       Upp.bound
                  rgdp(-1)     0.752           0.072           0.611           0.894
                  rgdp(-2)    -0.016           0.045          -0.104           0.072
                  rgdp(-3)    -0.001           0.031          -0.062           0.059
                   inf(-1)    -0.007           0.061          -0.128           0.113
                   inf(-2)    -0.029           0.038          -0.103           0.045
                   inf(-3)    -0.009           0.026          -0.060           0.042
                    un(-1)     0.105           0.142          -0.175           0.384
                    un(-2)    -0.041           0.137          -0.309           0.228
                    un(-3)    -0.019           0.091          -0.198           0.160
                  Constant    -0.342           0.442          -1.211           0.527

Sum of squared residuals: 13.82
R-squared: 0.476
adj. R-squared: 0.387


Endogenous: inf
                              Median          St.dev       Low.bound       Upp.bound
                  rgdp(-1)     0.049           0.130          -0.206           0.305
                  rgdp(-2)     0.007           0.081          -0.152           0.166
                  rgdp(-3)     0.005           0.056          -0.104           0.115
                   inf(-1)     0.230           0.110           0.013           0.447
                   inf(-2)     0.065           0.068          -0.068           0.199
                   inf(-3)    -0.044           0.047          -0.136           0.049
                    un(-1)    -0.067           0.256          -0.571           0.437
                    un(-2)    -0.009           0.247          -0.494           0.476
                    un(-3)     0.031           0.164          -0.292           0.355
                  Constant     0.771           0.798          -0.797           2.340

Sum of squared residuals: 33.91
R-squared: -0.147
adj. R-squared: -0.342
```

For each equation and each coefficient, the toolbox provides the point estimates (the median value), along with the standard deviation of the coefficient, the lower bound and the upper bound. For each equation, certain in-sample fit measures are also provided: the sum of squared residuals, the R-squared and the adjusted R-squared.

The third part of the output provides results for the model considered as a whole:

Figure 5.4.: *Third part of Matlab output*

```
Log 10 marginal likelihood: -59.86


Roots of the characteristic polynomial (modulus):
0.925   0.348   0.209
0.738   0.348   0.030
0.393   0.341   0.030
No root lies outside the unit circle.
The estimated VAR model satisfies the stability condition
```

Two elements are reported: the first is the log 10 of the marginal likelihood: this is an essential value for the selection of competing models. Note that for some models the marginal likelihood will not be computed, for instance if the model relies on improper prior distributions (as for the normal-diffuse prior). The second element is a stationarity check: it consists in reporting the roots of the characteristic polynomial of the median model (that is, the model obtained by using the point estimate of each coefficient). While it is a useful indication, it should be interpreted with caution: Bayesian models rely on the whole posterior distribution, not just on the median model. Hence even if your

median model is stationary, it may still be the case that many draws from the posterior will still be explosive, possibly resulting in non-stationary results for your applications.

The fourth part of the output is related to the residuals and shocks:

Figure 5.5.: *Fourth part of Matlab output*

```
sigma (residual covariance matrix): posterior estimates
 0.224   0.031  -0.033
 0.031   0.729  -0.002
-0.033  -0.002   0.033

D (structural decomposition matrix): posterior estimates
 1.000   0.000   0.000
 0.161   1.000   0.000
-0.119   0.008   1.000

gamma (structural disturbances covariance matrix): posterior estimates
 0.215   0.000   0.000
 0.000   0.626   0.000
 0.000   0.000   0.034
```

It first reports the posterior point estimates for the residual variance-covariance matrix Σ. Then, if a structural identification scheme was selected, it also reports the posterior estimates for the structural identification matrix D, and for the variance-covariance matrix of the structural disturbances Γ.

The final part of the output provides the forecast evaluation criteria. If forecasts or forecast evaluation were deactivated, this part will of course not appear:

Figure 5.6.: *Fifth part of Matlab output*

```
Forecast evaluation:

Evaluation conducted over 3 periods (from 2015q1 to 2015q3).

Endogenous: rgdp
                  2015q1      2015q2      2015q3
RMSE:              0.002       0.094       0.176
MAE:               0.002       0.068       0.137
MAPE:              0.486      17.294      42.801
Theil's U:         0.002       0.099       0.190
CRPS:              0.111       0.146       0.163
Log score 1:      -0.209      -0.481      -0.636
Log score 2:      -0.209      -0.463      -0.715

Endogenous: inf
                  2015q1      2015q2      2015q3
RMSE:              1.301       1.258       1.188
MAE:               1.301       1.257       1.182
MAPE:            126.940     101.359     120.640
Theil's U:         1.000       0.748       0.785
CRPS:              0.199       0.209       0.209
Log score 1:      -1.923      -1.765      -1.464
Log score 2:      -1.923      -4.453      -6.200

Endogenous: un
                  2015q1      2015q2      2015q3
RMSE:              0.265       0.346       0.467
MAE:               0.265       0.338       0.440
MAPE:              2.365       3.054       4.044
Theil's U:         0.012       0.015       0.021
CRPS:              0.042       0.069       0.098
Log score 1:      -0.238      -0.634      -1.227
Log score 2:      -0.238       0.397       0.782
```

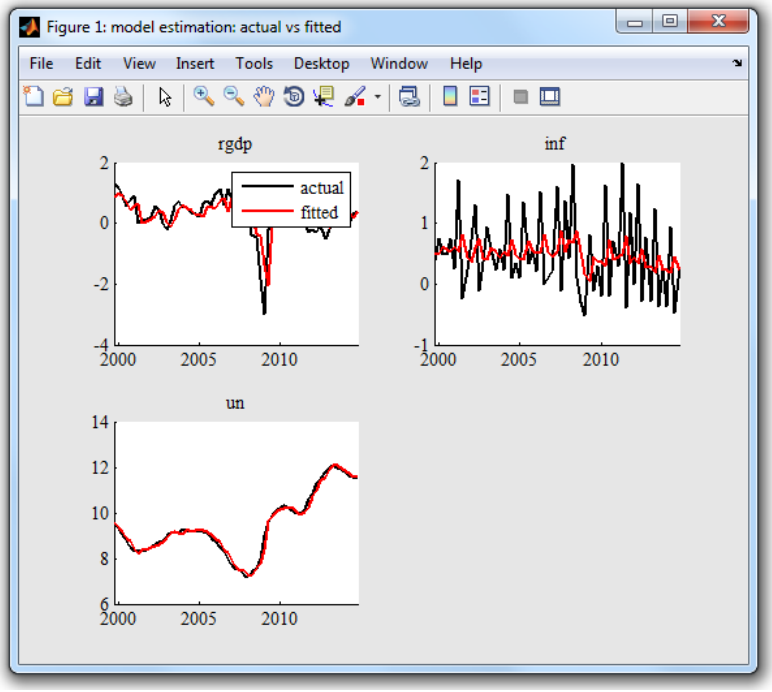
The BEAR toolbox provides 7 forecast evaluation criteria: the root mean squared error (RMSE), the mean absolute error (MAE), the mean absolute percentage error (MAPE), the Theil's U coefficient, the continuous ranked probability score (CRPS), and two different versions of the log score. Please refer to the technical guide for the definitions of the different criteria.

5.2. Matlab charts

The BEAR toolbox produces a series of charts illustrating your results. Those plots include both in-sample and application results. Any application that was de-activated will of course be missing in the plots. Assuming all the applications were selected, the set of lots produced by BEAR are the following:

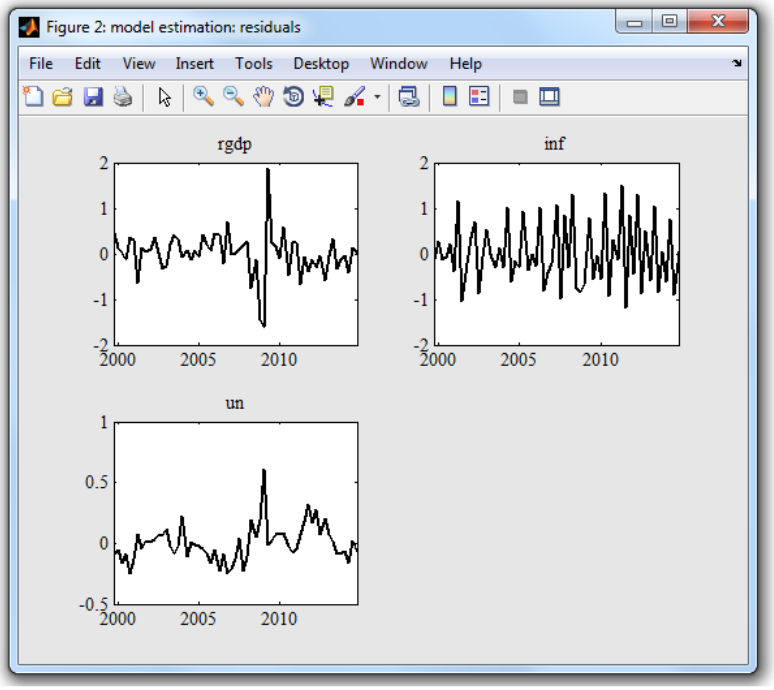
Actual VS fitted (obtained from the median model):

Figure 5.7.: *Actual VS fitted*



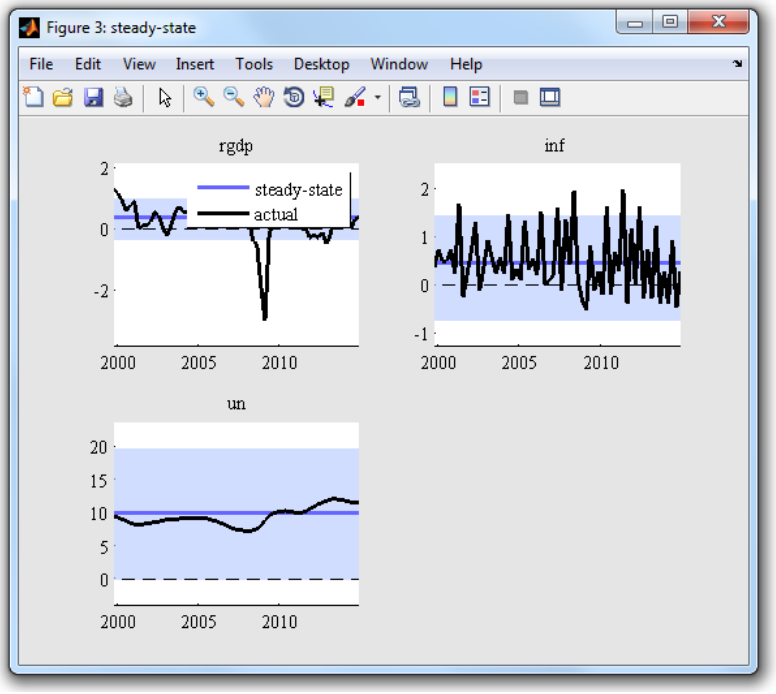
Residuals (obtained from the median model):

Figure 5.8.: *Residuals*



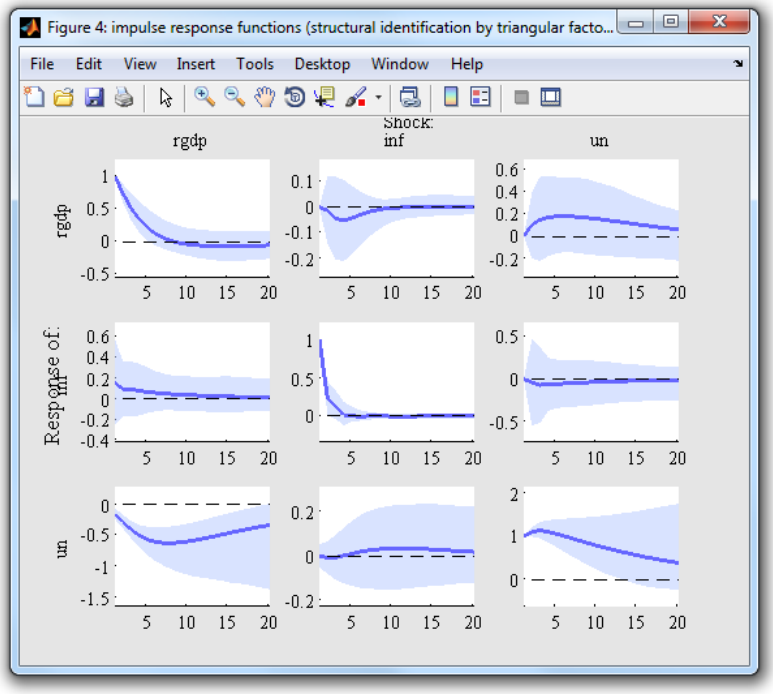
Steady-state:

Figure 5.9.: *Steady-state*



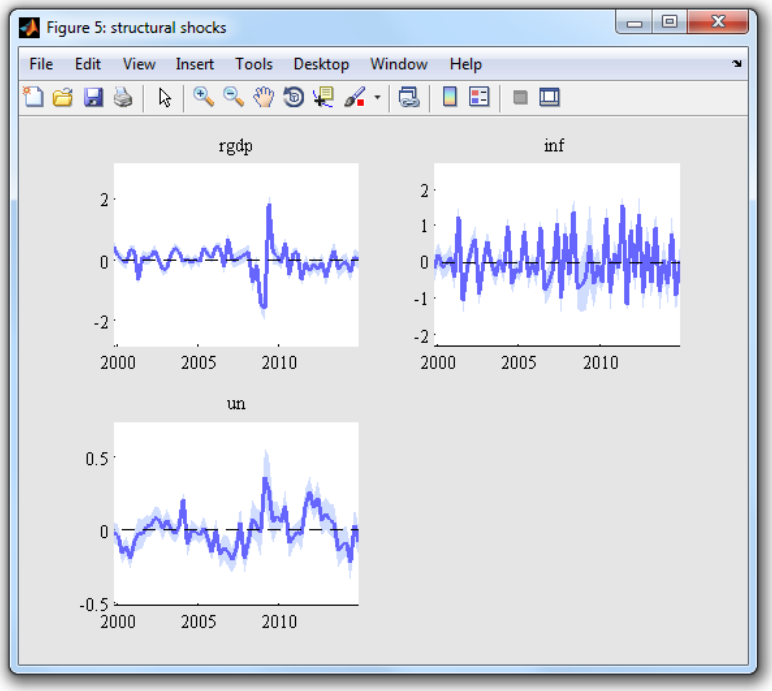
Impulse response functions:

Figure 5.10.: *Impulse response functions*



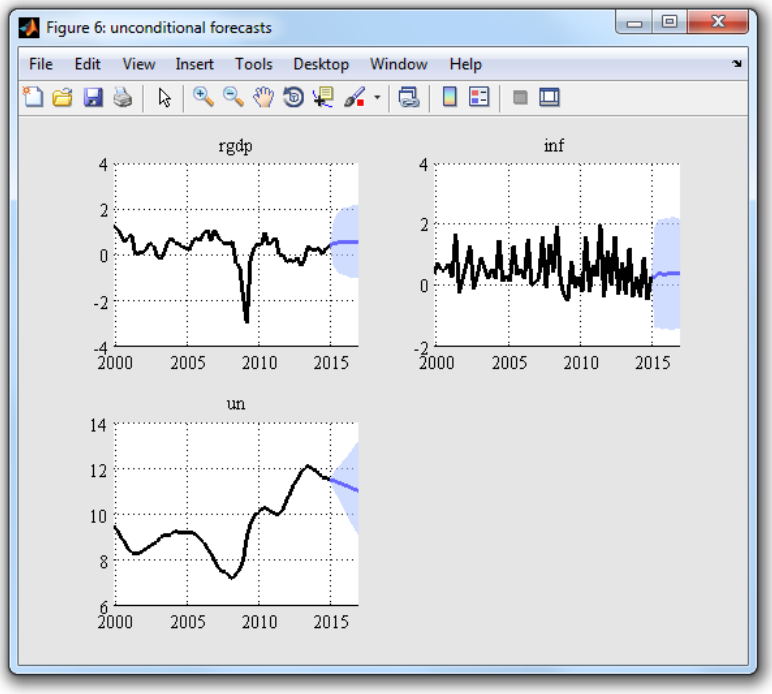
Structural shocks:

Figure 5.11.: *Structural shocks*



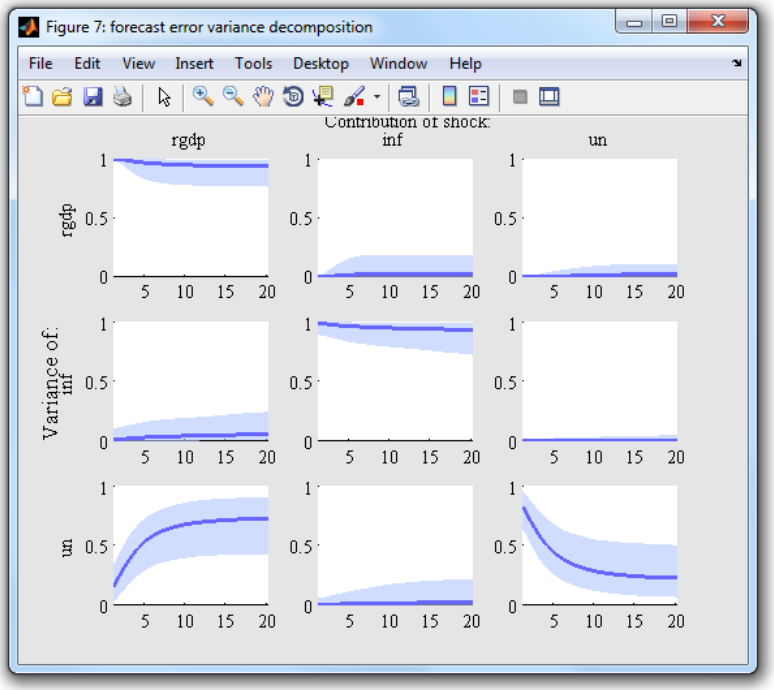
Unconditional forecasts:

Figure 5.12.: *Unconditional forecasts*



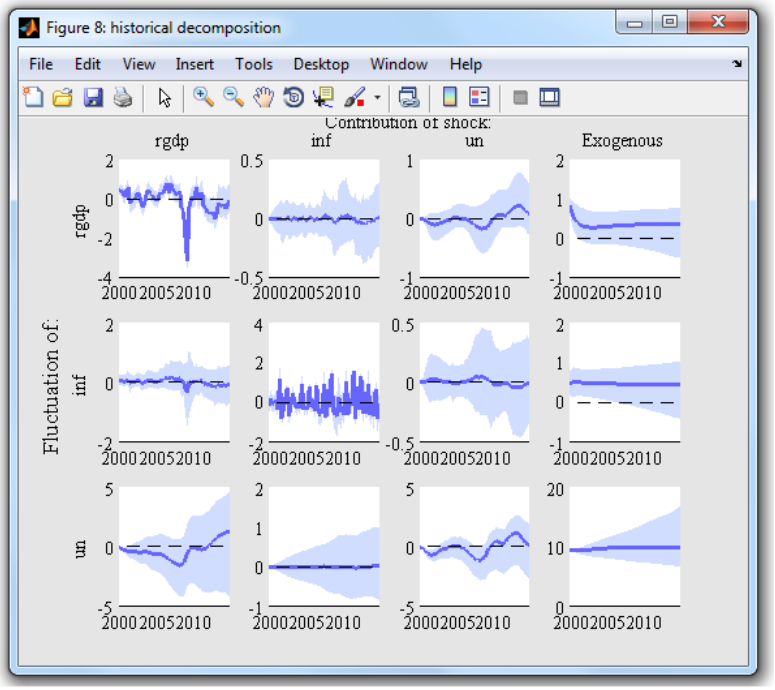
Forecast error variance decomposition:

Figure 5.13.: *Forecast error variance decomposition*



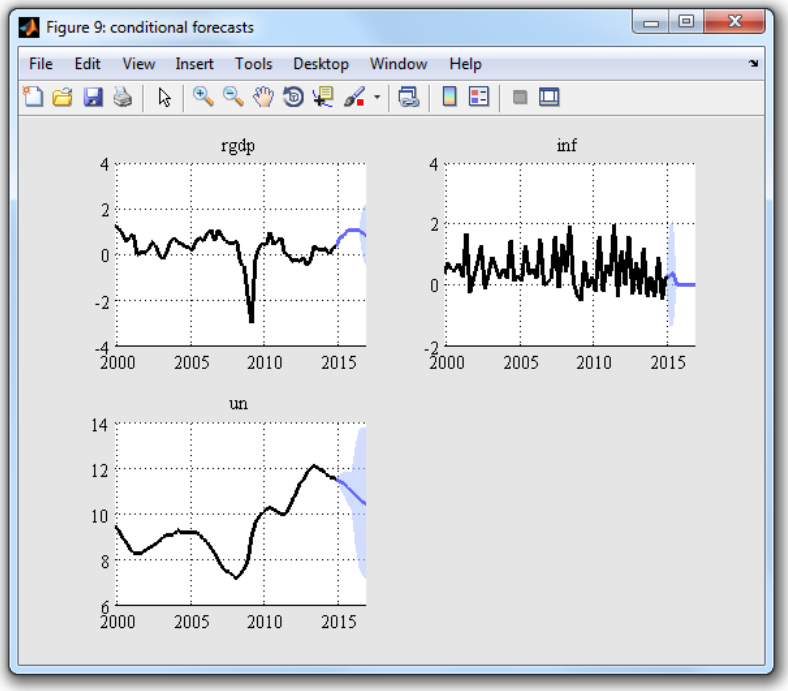
Historical decomposition:

Figure 5.14.: *Historical decomposition*



Conditional forecast:

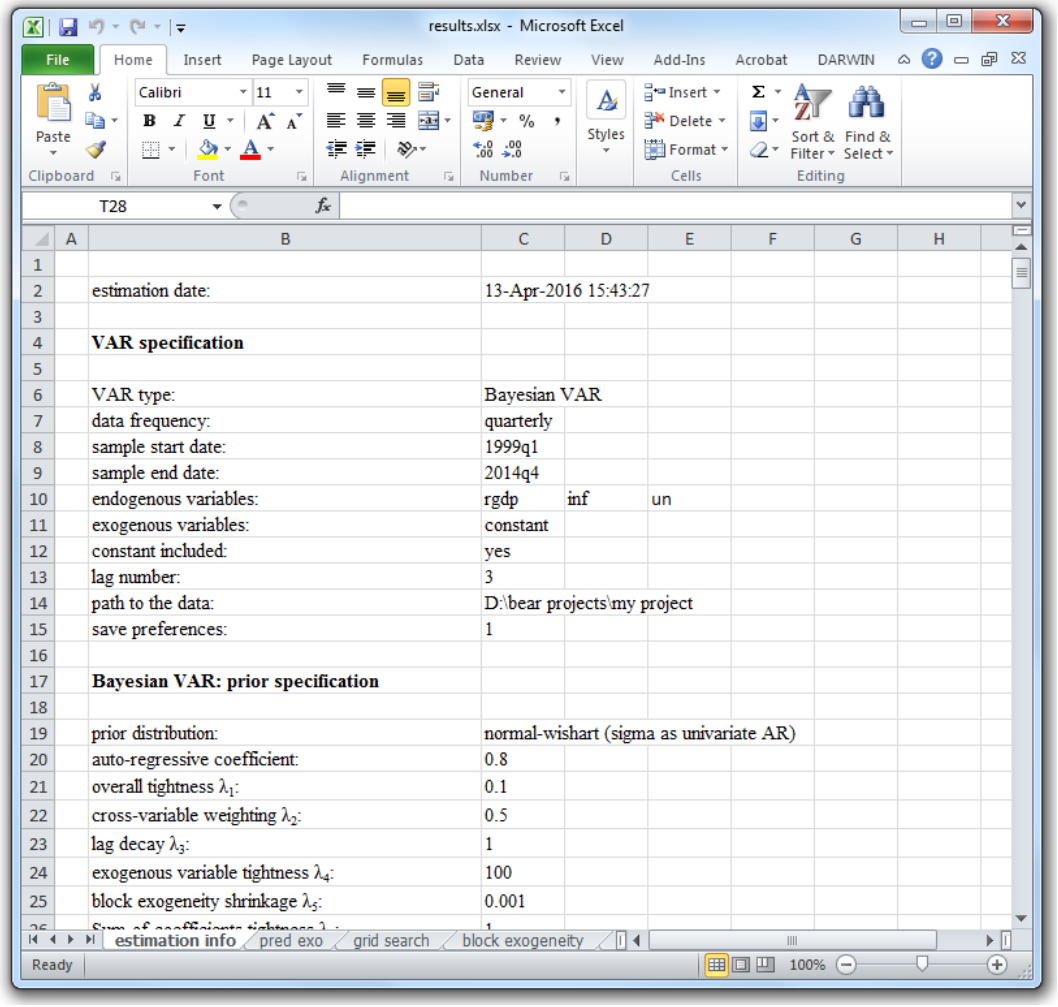
Figure 5.15.: *Conditional forecast*



5.3. Excel record

The final set of outputs produced by the BEAR toolbox is stored in the Excel file "results.xlsx". If you open this file once the estimation is over, it looks like this:

Figure 5.16.: *First page of results.xlsx file*

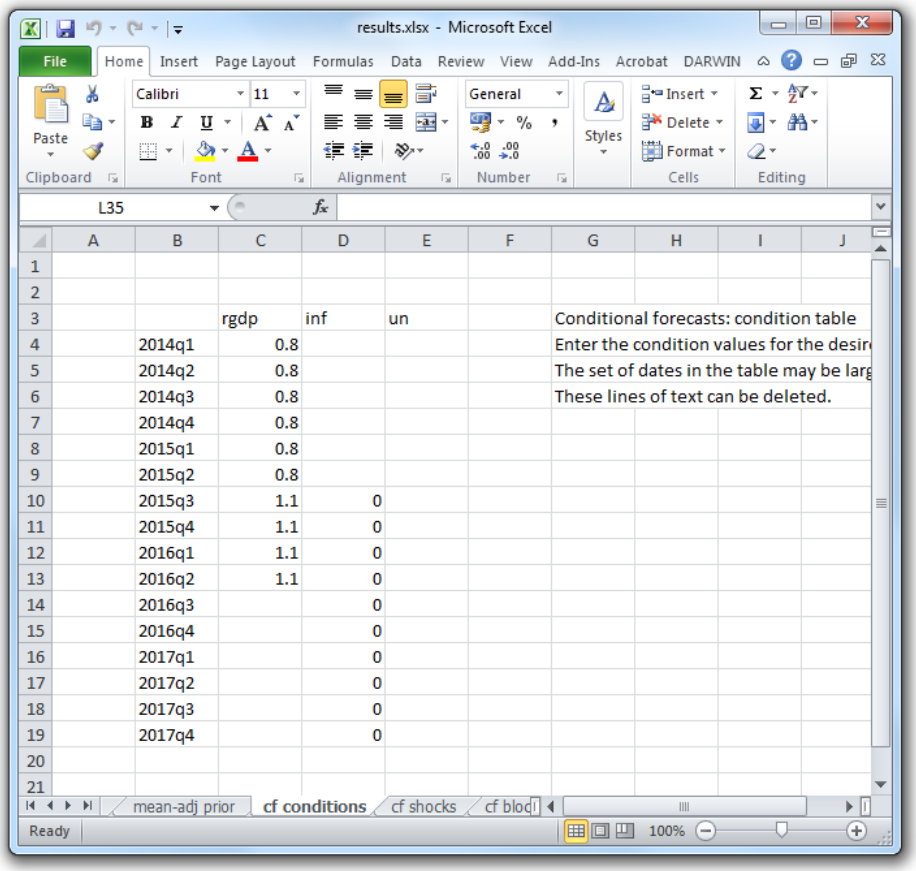


There are 3 different parts in this file. The first sheet "estimation info" (the one displayed in the figure) constitutes the first part. It exhaustively recapitulates the settings of your model. Hence, if you plan to replicate your estimation, you should be able to recover all the relevant information in this sheet. This sheet is comparable to the first part of the text output "results.txt", except that it is more complete.

The second part is dedicated to the recording of the information that you had to input by the way of Excel tables. It starts with the "pred exo" sheet, and ends up with the "cf intervals" sheet. Indeed, for many applications (predicted exogenous, grid search, block exogeneity, sign restrictions, conditional forecasts and the specification of prior values for the mean-adjusted model), your estimation information had to be written in dedicated tables in the "data.xlsx" spreadsheet. If this information is not saved somewhere in the output file, you might be unable to replicate your estimation later on if you modified your data file in the meantime. For this reason any table involved in the estimation process will be replicated in the "results.xlsx" file. For instance if you estimated

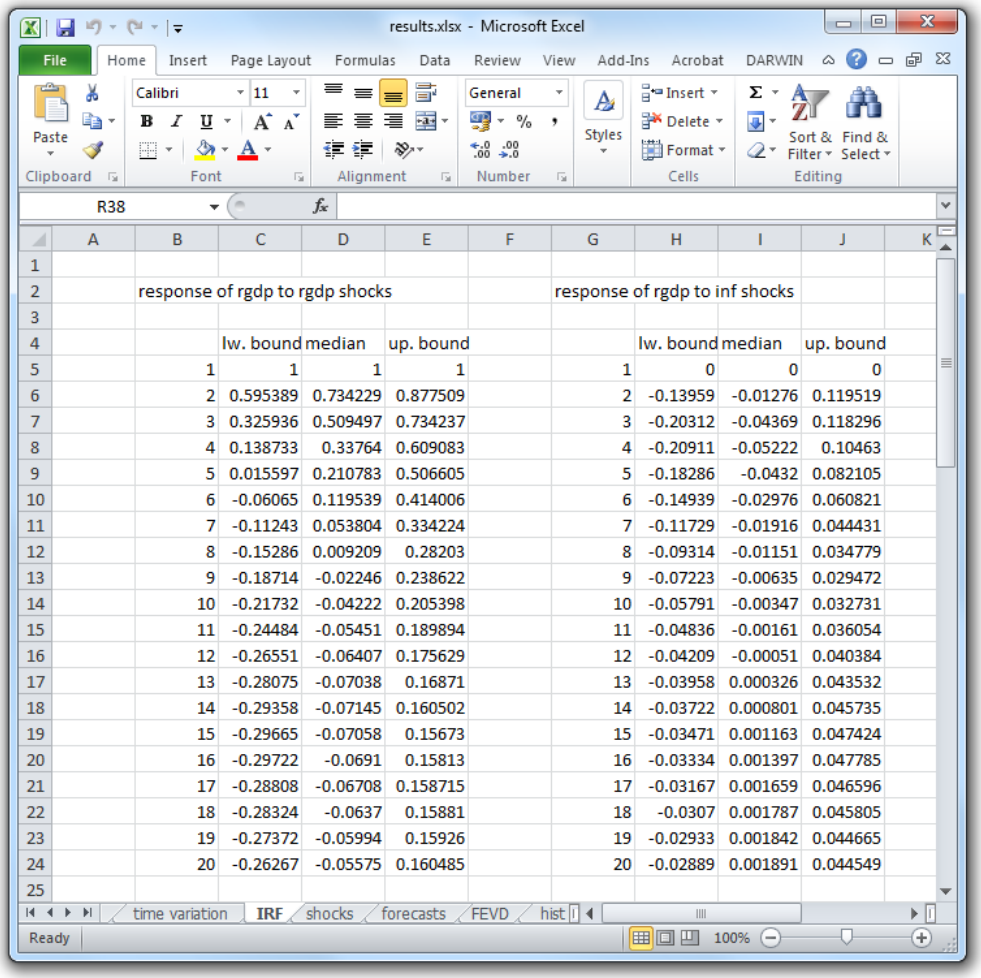
conditional forecasts, the sheet "cf conditions" will replicate your condition sheet:

Figure 5.17.: *Record of conditions in results.xlsx*



The final part of the file is constituted of the remaining worksheets. These sheets record the numerical values for all your applications. Indeed, even though Matlab produces charts for your applications, you may want to know the exact numerical values for your results, and possibly modify them, create your own plots, and so on. Therefore, for each application, you will find a record of your results in a corresponding Excel sheet. For instance if you selected the impulse response function application, your results will be saved in the sheet "IRF":

Figure 5.18.: *Record of impulse response functions in results.xlsx*



results.xlsx - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K
1											
2		response of rgdp to rgdp shocks					response of rgdp to inf shocks				
3											
4			lw. bound	median	up. bound			lw. bound	median	up. bound	
5		1	1	1	1		1	0	0	0	
6		2	0.595389	0.734229	0.877509		2	-0.13959	-0.01276	0.119519	
7		3	0.325936	0.509497	0.734237		3	-0.20312	-0.04369	0.118296	
8		4	0.138733	0.33764	0.609083		4	-0.20911	-0.05222	0.10463	
9		5	0.015597	0.210783	0.506605		5	-0.18286	-0.0432	0.082105	
10		6	-0.06065	0.119539	0.414006		6	-0.14939	-0.02976	0.060821	
11		7	-0.11243	0.053804	0.334224		7	-0.11729	-0.01916	0.044431	
12		8	-0.15286	0.009209	0.28203		8	-0.09314	-0.01151	0.034779	
13		9	-0.18714	-0.02246	0.238622		9	-0.07223	-0.00635	0.029472	
14		10	-0.21732	-0.04222	0.205398		10	-0.05791	-0.00347	0.032731	
15		11	-0.24484	-0.05451	0.189894		11	-0.04836	-0.00161	0.036054	
16		12	-0.26551	-0.06407	0.175629		12	-0.04209	-0.00051	0.040384	
17		13	-0.28075	-0.07038	0.16871		13	-0.03958	0.000326	0.043532	
18		14	-0.29358	-0.07145	0.160502		14	-0.03722	0.000801	0.045735	
19		15	-0.29665	-0.07058	0.15673		15	-0.03471	0.001163	0.047424	
20		16	-0.29722	-0.0691	0.15813		16	-0.03334	0.001397	0.047785	
21		17	-0.28808	-0.06708	0.158715		17	-0.03167	0.001659	0.046596	
22		18	-0.28324	-0.0637	0.15881		18	-0.0307	0.001787	0.045805	
23		19	-0.27372	-0.05994	0.15926		19	-0.02933	0.001842	0.044665	
24		20	-0.26267	-0.05575	0.160485		20	-0.02889	0.001891	0.044549	
25											

APPENDIX A

Appendix: description of the example datasets

For non-panel data:

yearly data: a model of small and large open economies (Luxembourg and France), comprising real GDP and inflation data. The series are "frgdp" (French GDP), "frinf" (French inflation), "luxgdp" (Luxembourg GDP) and "luxinf" (Luxembourg inflation).

quarterly data: a model of fluctuation for the US and the Euro area. The series are: "eurgdp" (growth rate of real GDP for the Euro area), "eurinf" (inflation for the Euro area), eurint (Euro area interbang rate), "usgdp", (growth rate of US real GDP), "usint" (US short term interest rate), and "ex", the Euro-dollar exchange rate.

monthly data: a model of inflation dynamics for Japan. The series are "inf" (Japanese inflation), "gdp" (growth rate of Japanese real GDP), "trade" (growth rate of trade balance), "u" (Japanese unemployment rate) and "ex" (exchange rate with the US dollar).

weekly data: a model of monetary policy for China. The series are "shibor" (Shanghai interbank rate),"exus" (exchange rate with the dollar), "exeur" (exchange rate with the euro), "vix" (financial volatility index), and "oil" (brent spot price).

daily data: a model of daily exchange rate for the yen. The series are: "dollar" (yen-dollar exchange rate), "euro" (yen-euro exchange rate), "rmb" (yen-yuan exchange rate), "rupiah" (yen-Indian rupiah exchange rate), and "baht" (yen-baht exchange rate).

undated data: the data set is just that of quarterly data, turned here into an undated set.

For panel data:

The set represents a model of fluctuation for four countries of the Euro area: Germany, France, Italy and Spain, with data respectively recorded in the sheets Ger, Fr, It and Sp. Each sheet contains four series: "rgdp" (growth rate of real GDP), "inf" (CPI inflation), "un" (harmonised unemployment rate), and "exdoll" (exchange rate with the US dollar). The first three variables are used as endogenous variables and are thus country-specific. The final one is used as an exogenous and is thus common to all countries.

Appendix A. Appendix: description of the example datasets

ï»¿

Frequently asked questions

My interfaces are not displayed correctly. In particular, the text size is wrong and bits seem to be missing.

This is a common issue which is not related to Matlab or the toolbox. Many computers (especially laptops) apply specific display settings. These settings typically magnify or reduce the case size while leaving other elements unchanged. While these display settings work fine with the Windows environment, they may result in bad display of the Matlab interfaces. To fix the problem, you need to modify your display settings. The specific options to change can vary from one computer to another, but typically you should take a look at your control panel.

My historical decomposition components don't sum to actual. Similarly, my forecast error variance decomposition components don't add up to 1. What is wrong there?

Basically nothing is wrong. To understand this you need to keep in mind that Bayesian estimation always results in a full posterior distribution, and not in a single point estimate. If you estimate a historical decomposition in a traditional (frequentist) framework, the point estimate that you obtain will by construction sum up to your actual value. But in a Bayesian framework, every component of your historical decomposition will be characterised by a full posterior distribution. Depending on the specific value you pick for each component from its posterior distribution, you will obtain different summation results, which will (most likely) not be equal to your actual value anyway. The point estimate that is adopted in BEAR is the median of each posterior distribution. Such an estimate should typically produce summations close to your actual value. However if the posterior is strongly skewed or has a large variance, you may obtain summations which depart significantly from your actual values. Common practice for historical decomposition then consists in summing the median contributions for all your shocks, hence creating a "total shock contribution" component. Then defining an "unconditional forecast, absent shock" component as the difference between actual and total shock component, one obtains again an exact identification.

The same logic applies to forecast error variance decomposition: the fact that you are dealing with a full distribution rather than a single point estimate explains why your summation may differ from 1. There is however no all-ready solution to deal with the case. You may deal with the relative weight of your respective components (i.e. mentioning for instance that a given component weights twice as much as some other component) rather than with the absolute contribution of each component. Or you may use the absolute contribution of each component anyway, simply ignoring the discrepancy with the theoretical unit total.

I have a VAR model with three variables. I use block exogeneity on variable 2 so that it has no effect on variable 1. Yet the impulse response function of variable 1 to

shock 2 is not equal to zero. How is that possible?

To understand this you need to consider precisely what block exogeneity implies. Block exogeneity effectively neutralises the response of a given variable (say 1) to past values of some other variables (say 2). In this sense block exogeneity works perfectly. But when it comes to impulse response functions, the transmission channel becomes more complex. Consider the specified VAR model (3 variables, block exogeneity on variable 2 with respect to variable 1), and the case of a shock on variable 2. At impact, block exogeneity indeed implies that the shock on variable 2 has no effect on variable 1. However, as block exogeneity is not applied to variable 3, the shock on variable 2 will have an effect on variable 3. In subsequent periods, variable 1 will respond to past values of variable 3, which are non-zero. This will result in variable 1 producing non-zero responses to shocks in variable 2, not directly as this is neutralised by block exogeneity, but indirectly through the channel created by variable 3.

The conclusion is straightforward: the problem here comes from the fact that variable 3 responds to variable 2. If you want to shut down any kind of spillover stemming from variable 2, you need not only to make it block exogenous to variable 1, but to variable 3 as well. In general if you want to make a variable or a block of variables exogenous, you need to insulate the block completely by making it exogenous to all the other variables of the model. In this sense, the name "block exogeneity" makes full sense as it reflects the idea of organising your variables into separate blocks. For instance if your VAR comprises some large economy variables (say for the US) and some other small economy variables (say for Cuba), and you implement block exogeneity to reflect the fact the Cuba should not affect the US, then the Cuban variables should be set as block exogenous to all the US variables. In this way, you effectively define a full block of Cuban variables which are exogenous to your block of US variables.

How do I discriminate among competing models? How can I decide which specification is best (number of lags, choice of the prior, hyperparameter values and so on) for my model?

The Bayesian field offers you a great evaluation criterion: the marginal likelihood. In theory, it should be sufficient to discriminate among models. And indeed, the grid search used to optimise the value of the hyperparameters relies on the marginal likelihood. In practice however, the marginal likelihood suffers from two main limitations. First, it is only available for a limited number of models. Many models don't have an associated marginal likelihood value, either because they use improper prior distributions, or because they are too complex to yield convenient formulas. Second, the marginal likelihood tends to favour certain specifications in a regular basis: for instance it will typically favour a normal-Wishart prior over a Minnesota prior, or a model with more lags over a model with fewer lags. In this respect, you may want to consider the results with some care.

Fortunately, BEAR provides you with additional evaluation criteria: in-sample criteria (sums of squared residuals, R-squared and adjusted R-squared), and forecast evaluation criteria (RMSE, MAE, MAPE, Theil's U, CRPS and log scores). In the end, your decision should be based on all these criteria jointly. How much weight you attribute to each of them ultimately depends on your objective. If your objective is to optimise the quality of your forecasts, then clearly forecast evaluation criteria

should be your sole consideration. If you want to run more qualitative study, you may consider more seriously the marginal likelihood and in-sample criteria.

The values for the R-squared and adjusted R-squared coefficients that I obtain are not comprised between 0 and 1. I even obtain negative values. Is there a problem?

No, there is no problem. There are several possible definitions for these criteria, and the ones that we use can produce values not comprised between zero and 1. This does not change anything to the interpretation however: the higher the value, the better the in-sample fit.

Bibliography

Arias, J. E., Rubio-Ramirez, J. F., and Waggoner, D. F. (2014). Inference Based on SVARs Identified with Sign and Zero Restrictions: Theory and Applications. Dynare Working Papers 30, CEPREMAP.

Canova, F. and Ciccarelli, M. (2013). Panel vector autoregressive models: a survey. Working Paper Series 1507, European Central Bank.

Jarocinski, M. (2010). Responses to monetary policy shocks in the east and the west of Europe: a comparison. *Journal of Applied Econometrics*, 25(5):833–868.

Litterman, R. (1986). Forecasting with Bayesian Vector Autoregressions – Five years of experience : Robert b. Litterman, Journal of Business and Economic Statistics 4 (1986) 25-38. *International Journal of Forecasting*, 2(4):497–498.

Pesaran, H. and Smith, R. (1995). Estimating long-run relationships from dynamic heterogeneous panels. *Journal of Econometrics*, 1:79–113.

Robertson, J. C., Tallman, E. W., and Whiteman, C. H. (2005). Forecasting Using Relative Entropy. *Journal of Money, Credit and Banking*, 37(3):383–401.

Villani, M. (2009). Steady-state priors for vector autoregressions. *Journal of Applied Econometrics*, 24(4):630–650.

Waggoner, D. F. and Zha, T. (1999). Conditional Forecasts In Dynamic Multivariate Models. *The Review of Economics and Statistics*, 81(4):639–651.

Zellner, A. and Hong, C. (1989). Forecasting international growth rates using bayesian shrinkage and other procedures. *Journal of Econometrics*, 40(1):183–202.