Description of outputs for SpecEval add-in

This document describes output objects for specification evaluation add-in. Below is the table of contents. First section includes list of all outputs with brief descriptions. Following sections each corresponds to main components of the add-in. Each section will describe the informational content of the corresponding objects that are included, as well as provide information on their name and location in the spool.[1] Last section contains list of additional workfile objects and potentially stored by the add-in. Appendix explains in detail the procedures used to produce the various forecasts and equations on which the outputs are based.

Contents

[1] The names correspond to names in situation when single specification is being evaluated; if multiple specifications are being evaluated then workfile names include additional alias at the end of name, while the objects in spool are named ‘spec_{id}’.

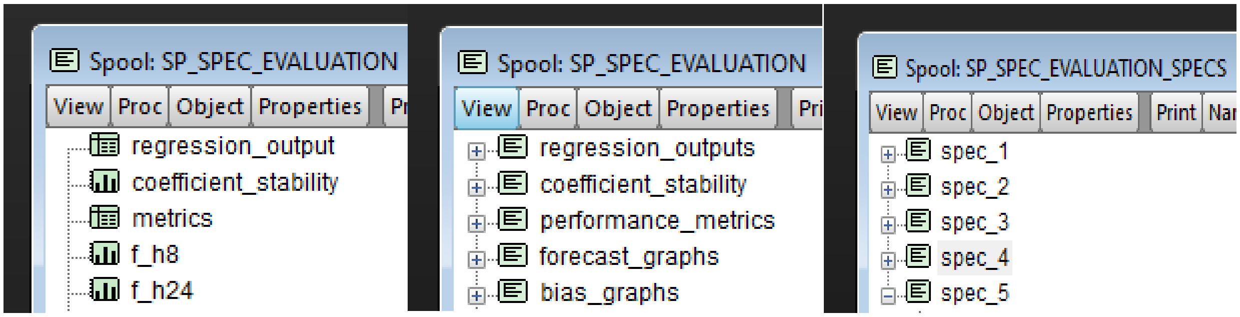
List of add-in outputs

Table below provides terms used for, and brief description of, all the output objects. The names are linked with the illustrations in this document that directly follow the discussion of the objects in detail. The order follows

Object name	Description
Regression output	Adjusted regression output table
Coefficient stability graph	Graph with recursive equation coefficients
: Model stability graph	Graph with recursive lag orders
Performance metrics table	Table with values of forecast performance metrics
Performance metrics table (multiple specifications)	Table with values of forecast performance metrics for given metric for all specifications
Forecast summary graph	Graph with all recursive forecasts with given horizons
Sub-sample forecast graph	Graph with forecast for given sub-sample
Subsample forecast decomposition graph	Graph with decomposition of sub-sample forecast
Forecast bias graph	Scatter plot of forecast and actual values for given forecast horizon (Minzer-Zarnowitz plot)
Individual conditional scenario forecast graph (level)	Graph with forecast for single scenario and specification
Individual conditional scenario forecast graph (transformation)	Graph with transformation of forecast for single scenario and specification
All conditional scenario forecast graph	Graph with forecasts for all scenarios for single speciication
Multiple specification conditional scenario forecast graph	Graph with forecasts for single scenario for multiple specifications
Shock response graphs	Graphs with response to shock to individual independent variable/regressor

Spools with all results

All the individual outputs are stored in spools. When the program is executed for single specification then the spool ‘sp_spec_performance’ contains outputs for given specification. This spool will be referred to in the rest of this document simply as “the spool”. When the program is executed for multiple specifications then the spool ‘sp_spec_performance’ contains results for all specifications, organized by output. In addition there will be spool ‘sp_spec_performance_specs’ that will contain results organized by specification, each in one sub-spool. See illustrations below:

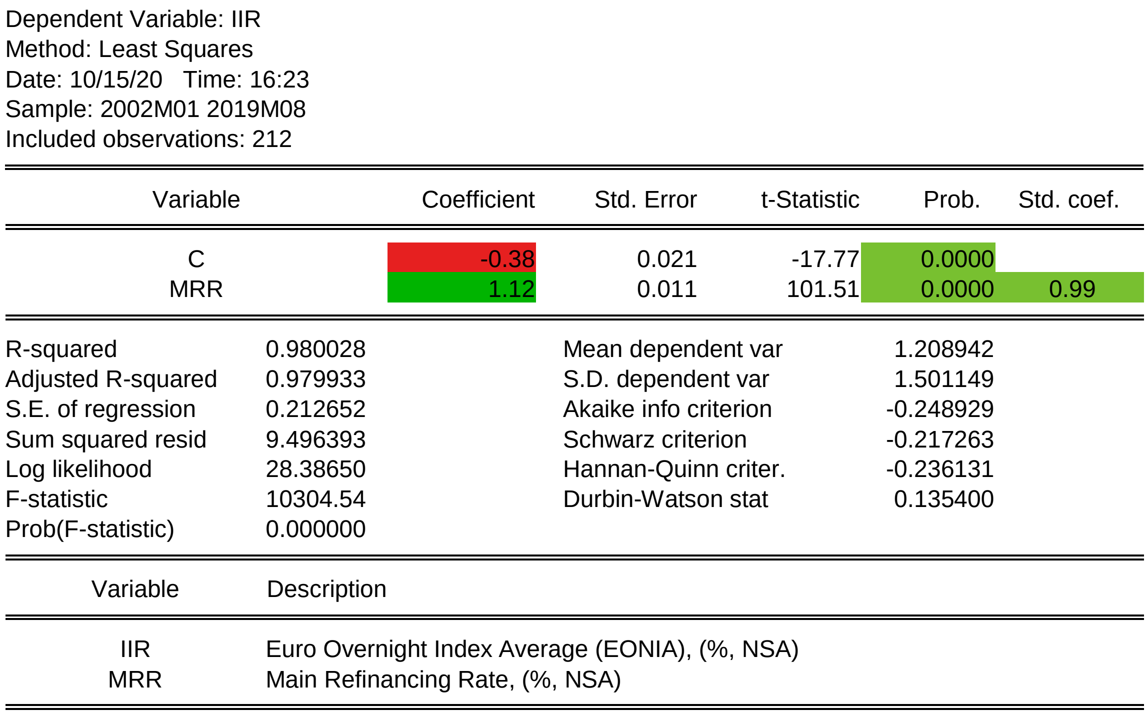

Spool: SP_SPEC_EVALUATION
View Proc Object Properties Pr
regression_output
coefficient_stability
metrics
f_h8
f_h24
Spool: SP_SPEC_EVALUATION
View Proc Object Properties Pri
regression_outputs
coefficient_stability
performance_metrics
forecast_graphs
bias_graphs
Spool: SP_SPEC_EVALUATION_SPECS
View Proc Object Properties Print Nar
spec_1
spec_2
spec_3
spec_4
spec_5

Regression information

There are up to three outputs that provide information about the regression. First, there is the basic regression output based on the equation in its current form in the workfile, with several adjustments, see Table 1. The output is stored as 'tb_reg_output' in the workfile[2], and as 'regression_output' in the spool. The output contains following adjustments:

- All values associated with estimated coefficients are formatted to either 2 decimals or 3 significant digits.
- Coefficients are color coded, dark red for negative values and dark green for positive values
- T-statistics and p-values are color coded according to significance (light green for significant, yellow for marginally significant, orange for insignificant)[3]
- Standardized coefficients are included; they are color coded, with light green for values above 0.5, orange for values below 0.1, and yellow for values in between.
- Below the regression output there is section with list of independent variables and associated descriptions[4]

Table 1: Regression output



Dependent Variable: IIR
Method: Least Squares
Date: 10/15/20   Time: 16:23
Sample: 2002M01 2019M08
Included observations: 212

Variable	Coefficient	Std. Error	t-Statistic	Prob.	Std. coef.
C	-0.38	0.021	-17.77	0.0000	
MRR	1.12	0.011	101.51	0.0000	0.99

R-squared	0.980028	Mean dependent var	1.208942
Adjusted R-squared	0.979933	S.D. dependent var	1.501149
S.E. of regression	0.212652	Akaike info criterion	-0.248929
Sum squared resid	9.496393	Schwarz criterion	-0.217263
Log likelihood	28.38650	Hannan-Quinn criter.	-0.236131
F-statistic	10304.54	Durbin-Watson stat	0.135400
Prob(F-statistic)	0.000000		

Variable	Description
IIR	Euro Overnight Index Average (EONIA), (%, NSA)
MRR	Main Refinancing Rate, (%, NSA)

[2] This assumes that user specified to that he wishes to keep intermediate objects. Same applies for discussion below. See list of intermediate objects in dedicated section.

[3] The two significance levels are 0.1 and 0.25.

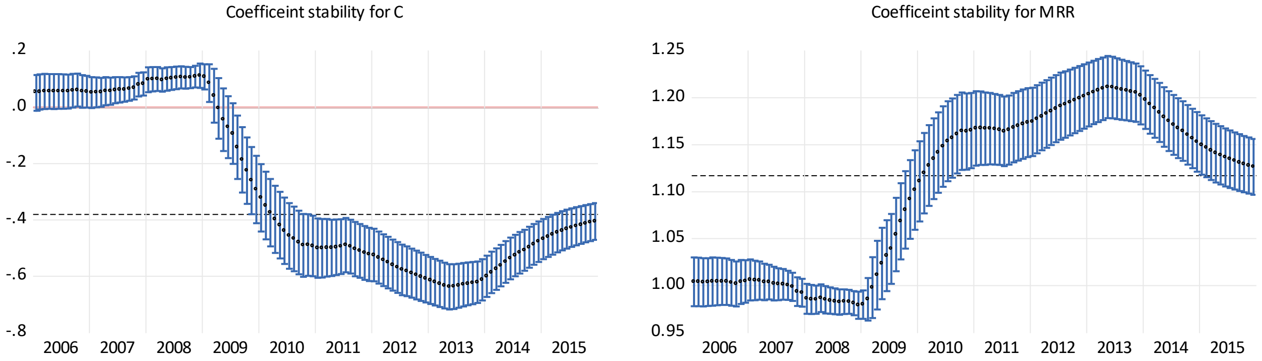
[4] The descriptions are by default equal to the descriptions in given series. User can overwrite this behavior by creating table 'tb_variable_descriptions' which will include variable mnemonic in first column and description to be used in second column.

Note that the standard regression table is adjusted to accommodate efficient inclusion of long-string regressors and variables with long descriptions.

The other two outputs related to the regression itself are applicable when the forecasts are made out-of-sample so that the equation is re-estimated on different estimation samples, which results in different coefficient estimates and possibly model structure; see appendix for discussion of the re-estimation and forecasting process.

Correspondingly, the second output is the coefficient stability graph while the third output is model stability graph. The coefficient stability graph captures the time series of recursive coefficient estimates, together with 2-standard deviations error bands and line indicating full/original estimation sample coefficient, see Figure 1. The graph is stored as 'gp_coef_stability' in the workfile and as 'coefficient_stability' in the spool.

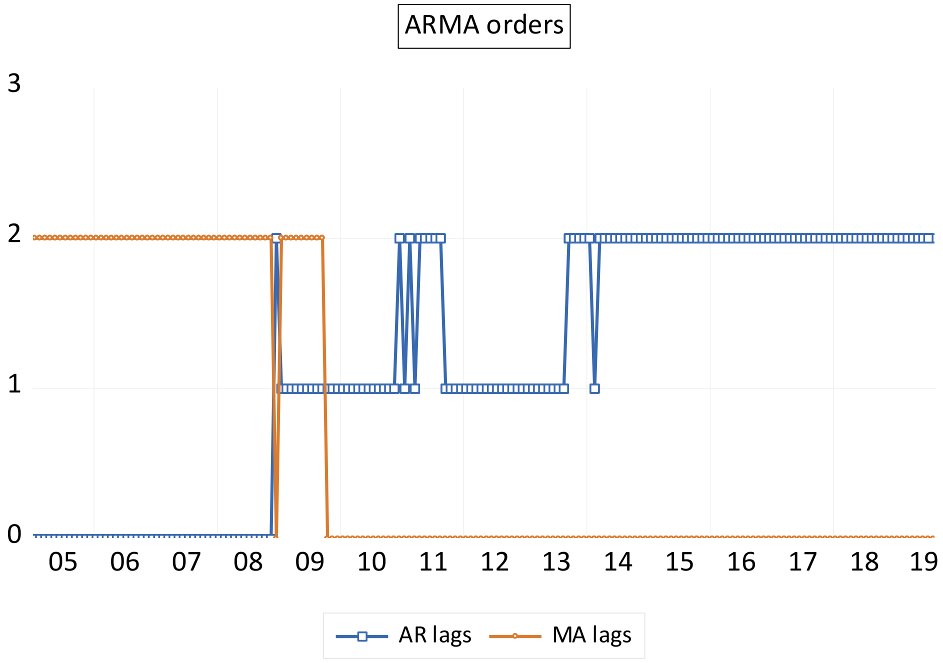
Figure 1: Coefficient stability graph



Circles: Recursive coefficient estimates
Dahsed line: Full sample coefficient
Blue lines: Confidence interval

The third output is graph of automatically selected lag orders, see Figure 2. As such it is provided only when automatic model selection is used. The graph is called ‘gp_lag_orders’ in workfile and ‘lag_orders’ in the spool. It varies between ARMA, ARDL and VAR models. Below is example for case of ARMA model.

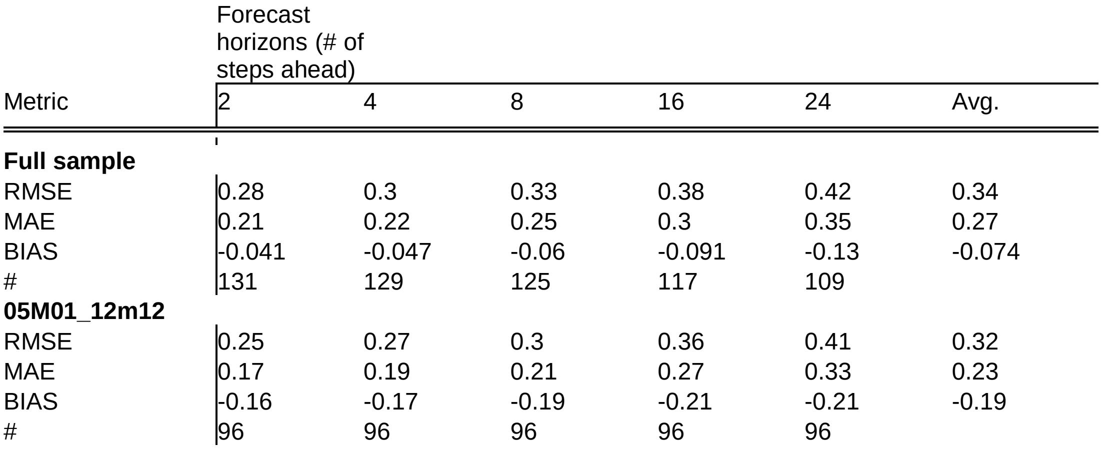
Figure 2: Model stability graph



Performance metrics

All performance metrics are stored in single table. The table is called ‘tb_performance_metrics’ in the workfile, and ‘metrics’ in the spool. The table contains the value for each performance metric specified by user, for all the horizons specified by user. For example, in Table 2, the value in row 6 column 3 indicates that the mean of absolute values of all available forecasts errors for 4-steps-ahead forecasts is 0.22. In addition to these values corresponding to individual forecast metric-forecast horizon combinations, the table also includes average value across all forecast horizons, stored in last column; and number of actual forecasts that have been used in computing the overall metric (in row 8 in illustration below). Finally, if sub-samples are specified then the table will contain one section for full sample, and one for each sub-sample.[5]

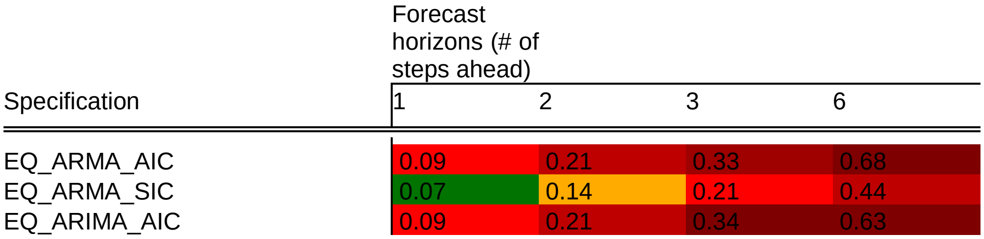
Table 2: Performance metrics table



	Forecast horizons (# of steps ahead)					
Metric	2	4	8	16	24	Avg.
Full sample						
RMSE	0.28	0.3	0.33	0.38	0.42	0.34
MAE	0.21	0.22	0.25	0.3	0.35	0.27
BIAS	-0.041	-0.047	-0.06	-0.091	-0.13	-0.074
#	131	129	125	117	109	
05M01_12m12						
RMSE	0.25	0.27	0.3	0.36	0.41	0.32
MAE	0.17	0.19	0.21	0.27	0.33	0.23
BIAS	-0.16	-0.17	-0.19	-0.21	-0.21	-0.19
#	96	96	96	96	96	

When the program is executed for multiple specifications than the one table for all metrics is broken in to individual tables for each metric. For example, there will be one table to full-sample RMSE, one table for full-sample MAE, and one table for each RMSE and MAE for each sub-sample. The tables will contain results for all alternative specifications in successive rows, with first column indicating equation name or alias, see Table 3. Finally, the table will be color-coded, with lowest values dark green, followed by light green, followed by yellow, followed by orange, followed by red, and finally by dark red.[6] See illustration below.

Table 3: Performance metrics table (multiple specifications)



	Forecast horizons (# of steps ahead)			
Specification	1	2	3	6
EQ_ARMA_AIC	0.09	0.21	0.33	0.68
EQ_ARMA_SIC	0.07	0.14	0.21	0.44
EQ_ARIMA_AIC	0.09	0.21	0.34	0.63

[5] To count as subsample-relevant forecast error, given forecast has to completely lie in the given sample.

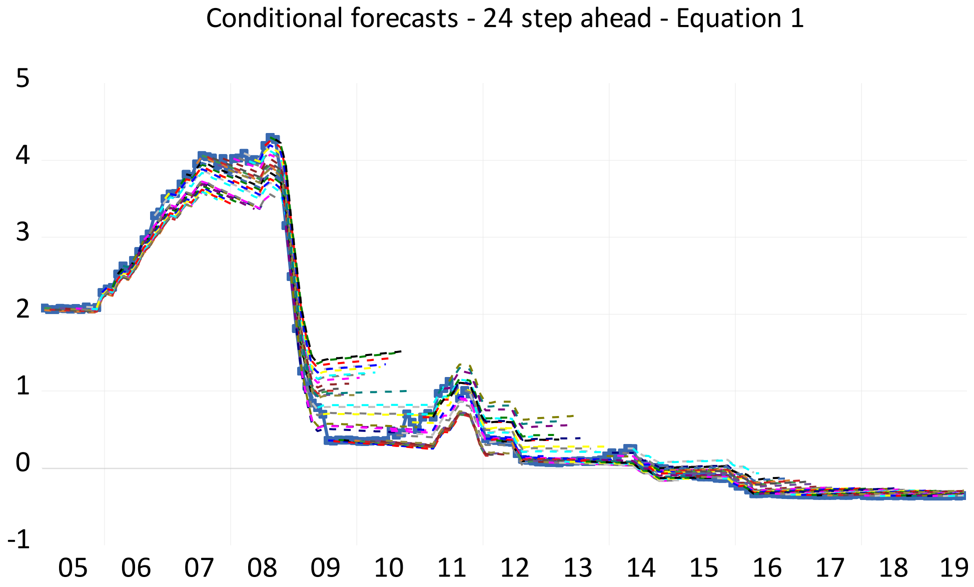
[6] The number of color scales increases with number of specifications, with 4 specifications for one color-scale.

EQ_ARIMA_SIC	0.05	0.09	0.13	0.24

Forecast graphs

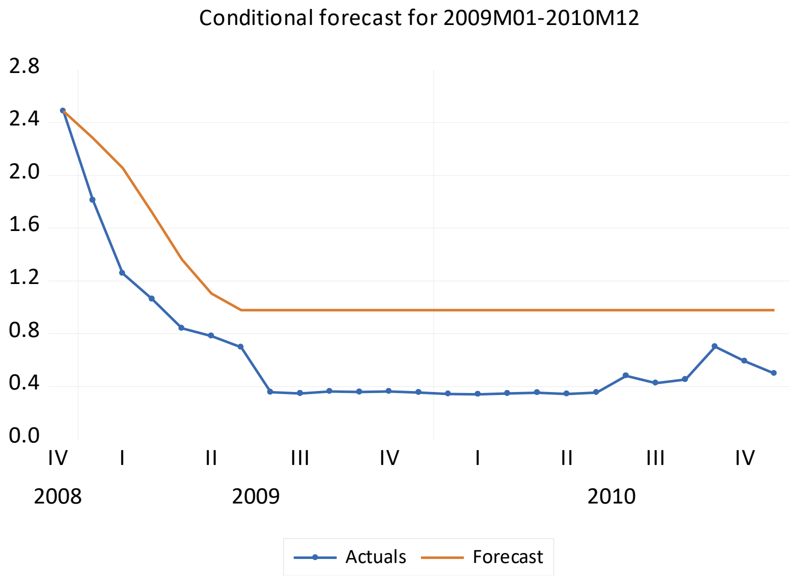
There two types of forecast graphs. First type contains all forecasts for given horizon. The graph is stored as ‘gp_forecasts_all_h{horizon}’ in the workfile, and as ‘f_h{horizon}’ in the spool. These graphs include the actual historical series with filled squares symbol, together with all the individual forecasts (dashed without symbol; the colors follow default coloring scheme). See Figure 1 for illustration. The graphs also contain any additional series specified by user, again including symbols. Note that the legend is included only when additional series are included and refers only to the historical series and additional series, not to the individual forecasts. The graph sample is full back-testing sample, with potential adjustments specified by user. See illustration below.

Figure 3: Forecast summary graph



Second type contains single forecast covering the specified sub-sample in addition to the historical series, as in Figure 4. The graphs are stored as ‘gp_forecast_subsample{subsemaple}’ in the workfile and as ‘f_{start}_{end}’ in the spool (e.g. f_08M01_10M12’).

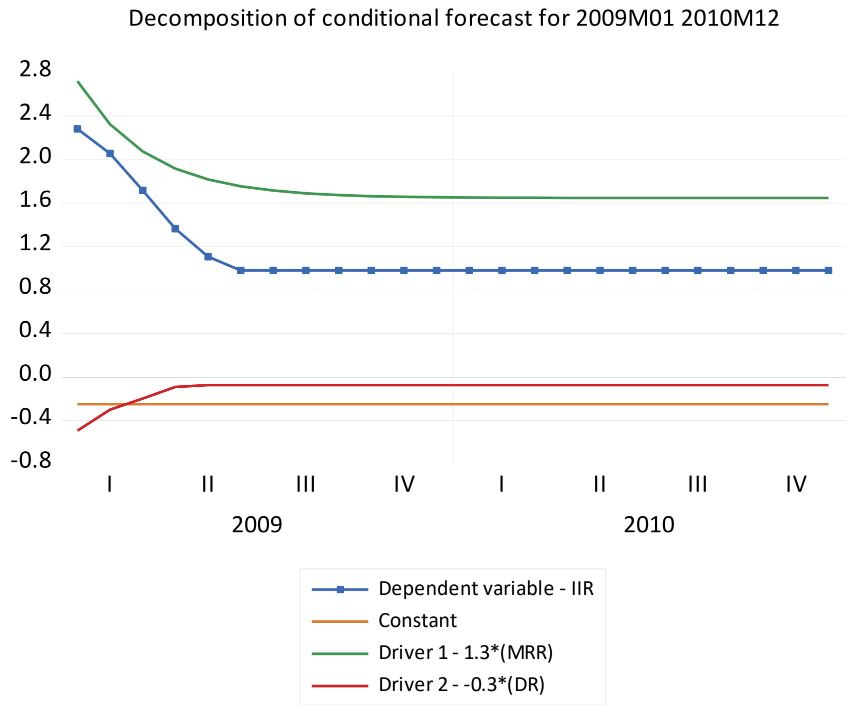
Figure 4: Sub-sample forecast graph



Both above graphs will obey the transformation settings.

Finally, for each sub-sample there is also forecast decomposition graph, called 'gp_fd_ss{subsample}' in workfile and 'f_{start}_{end}_decomposition' in the spool. See document dedicated to forecast decomposition add-in output for explanation of this graph.

Figure 5: Subsample forecast decomposition graph

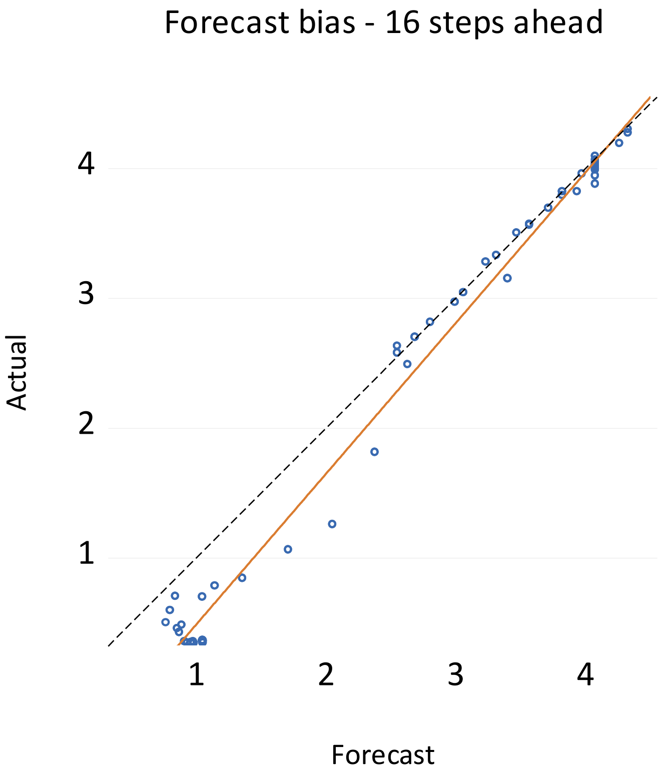


When the program is executed for multiple specification then the graphs are stored in in sub-spool 'forecast_graphs' in individual sub-spools corresponding to forecast horizon or to sub-sample named as indicated above.

Bias graphs

Bias graphs are the Minzer-Zarnowitz graphs showing the scatter plot of actual values against the forecast values, see Figure 6. In addition they display the regression line from regression of actual values on forecast values as well as 45-degree line which indicates unbiased forecasts.[7] The graphs are stored under name 'gp_forecast_bias_h{horizon}' in the workfile, and 'bias_h{horizon}' in the spool.

Figure 6: Forecast bias graph



When the program is executed for multiple specification then the graphs are stored in in sub-spool 'bias_graphs' in individual sub-spools corresponding to forecast horizon.

[7] Line with slope less than 45-degrees means that the actuals increase slower than the forecasts. Meanwhile, values above 45-degree line correspond to observations with forecasts lower than actuals (i.e. negative bias) and vice versa.

Scenario graphs

When list of scenarios is specified then the add-in creates conditional scenario forecasts. These are forecasts for the base variable corresponding to specified values of right-hand side variables which correspond to different scenarios.

There are multiple types of graphs for conditional[8] scenario forecast graphs. They can be divided in three subcategories: graphs of forecasts for single specification; graphs of forecasts for multiple specifications; and forecast decomposition graphs. The first two categories include two separate graphs, one in level and one in transformation; the last category includes two separate graphs, one in level and in scenario difference.

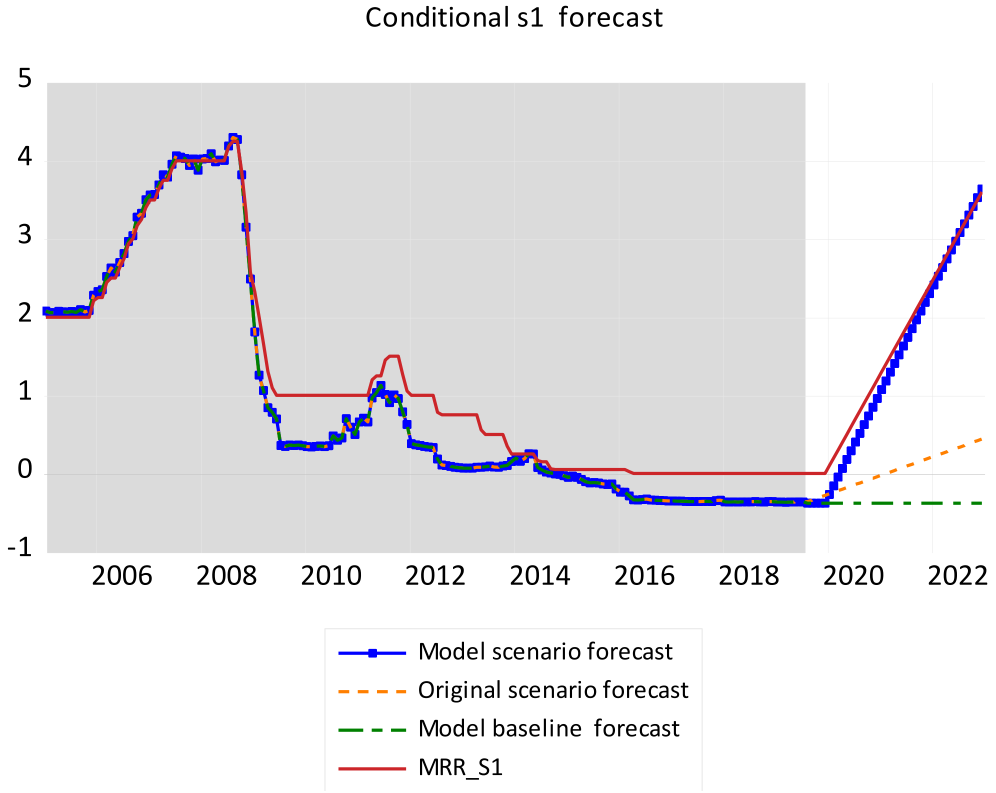
Single specification forecast graphs

The single specification conditional scenario forecast graphs are stored in subspool 'conditional_scenarios'. This subspool is then subdivided into additional spools, one for each scenario and one for all scenarios (named 'all').

There are three types of graphs for conditional scenario forecasts. First type is the individual scenario forecast graphs in level, stored as 'gp_csf_level_{scenario}' in workfile and '{scenario}_level' in the spool. These primarily capture the conditional scenario forecast (blue with squares); they can also include original scenario forecast (i.e. scenario forecast located in workfile prior to the execution of the add-in; orange with dashed line) and/or baseline scenario forecast (i.e. conditional scenario forecast for baseline scenario; green with dash-dots), as well additional series specified by user (solid lines). See Figure 7 for illustration. The actual conditional scenario forecast is labeled "Model scenario forecast", the original scenario forecast is labeled "Original scenario forecast", while the baseline scenario forecast is labeled "Model baseline forecast"; finally, the additional series are labeled with their full name. See illustration below.

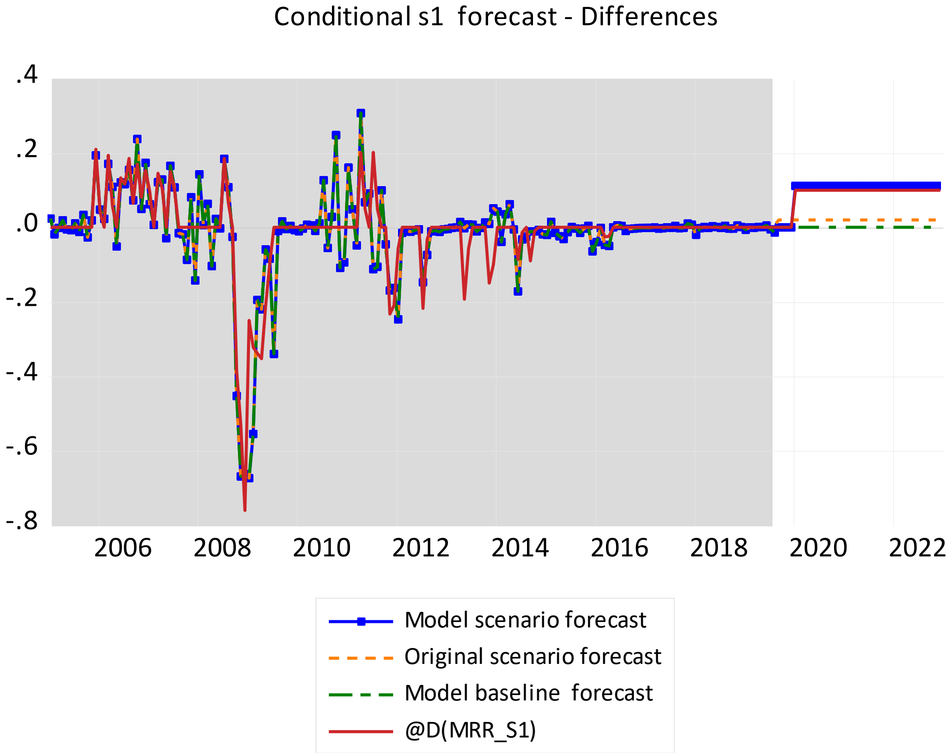
[8] Given that the add-in operates mostly in terms of single equation approach, the scenario forecasts are made *conditional* on actual scenario values of independent variables. See appendix for more detailed discussion.

Figure 7: Individual conditional scenario forecast graph (level)



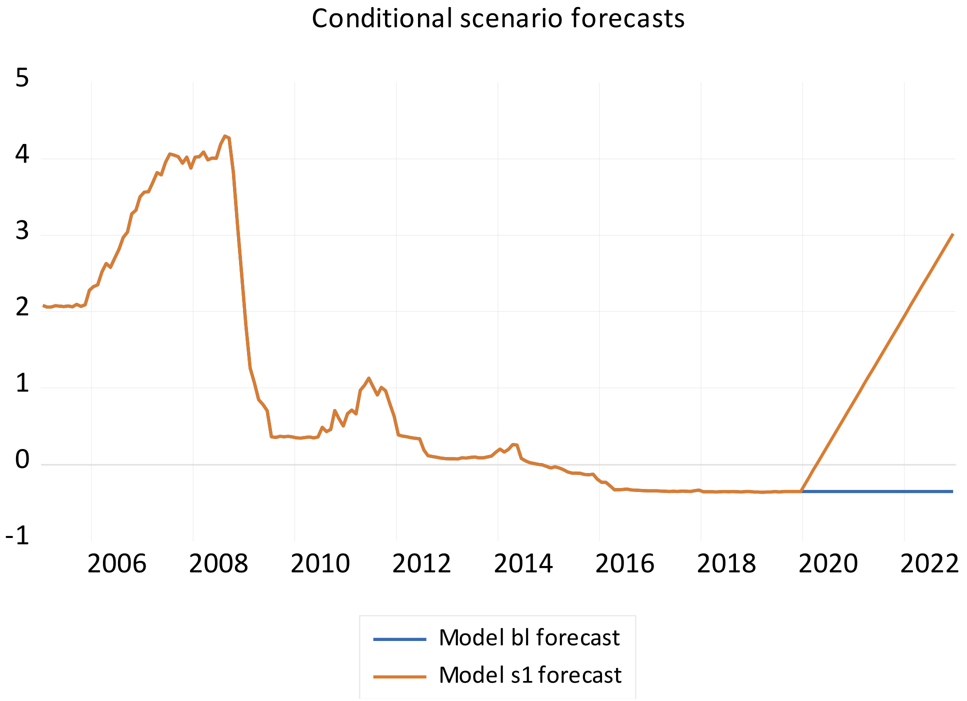
Second type is conditional scenario forecasts in transformation, stored as 'gp_csf_trans_{scenario}' in workfile and '{scenario}_trans' in the spool. The graph will include the same series as the level graph. Apart from the standard transformations (difference/growth rate/spread/index/ration/logarithm) user can also specify deviation transformation, in which case the graph will display difference/deviation between given scenario and baseline scenario conditional forecast. See illustration in Figure 8.

Figure 8: Individual conditional scenario forecast graph (transformation)



Final graph type is graph of all scenarios together, stored as 'gp_csf_level_all' and 'gp_csf_trans_all' in workfile and 'all_transformation' and 'all_transformation' in the spool. This graph displays the conditional scenario forecasts for all user-specified scenarios, and does not include any other series, see Figure 9.

Figure 9: All conditional scenario forecast graph



When the program is executed for multiple specifications then all the above graphs are stored in spool ‘conditional_scenarios’, in individual sub-spools dedicated to each scenario named with the scenario alias (e.g. “s1”) and one sub-spool for all-scenario graphs. Each scenario sub-spool is then divided into one sub-spool for level graphs and one for transformation, named ‘level’ and ‘transformation’, respectively.

Decomposition graphs

Conditional scenario forecast decomposition graphs are located in subspool called ‘scenario_decomposition’. There are two types of scenario decomposition graphs. First, there is level decomposition graph for each scenario, stored as ‘gp_csf_fd_{scenario}’ in workfile and ‘{scenario}_decomposition’ in the spool. Second, there is scenario-difference decomposition graph for each scenario except for baseline scenario, stored as ‘gp_csf_fdd_{scenario}’ in workfile and ‘{scenario}_decomposition_diff’ in the spool. See document dedicated to forecast decomposition add-in outputs for explanation of these graphs and illustration.

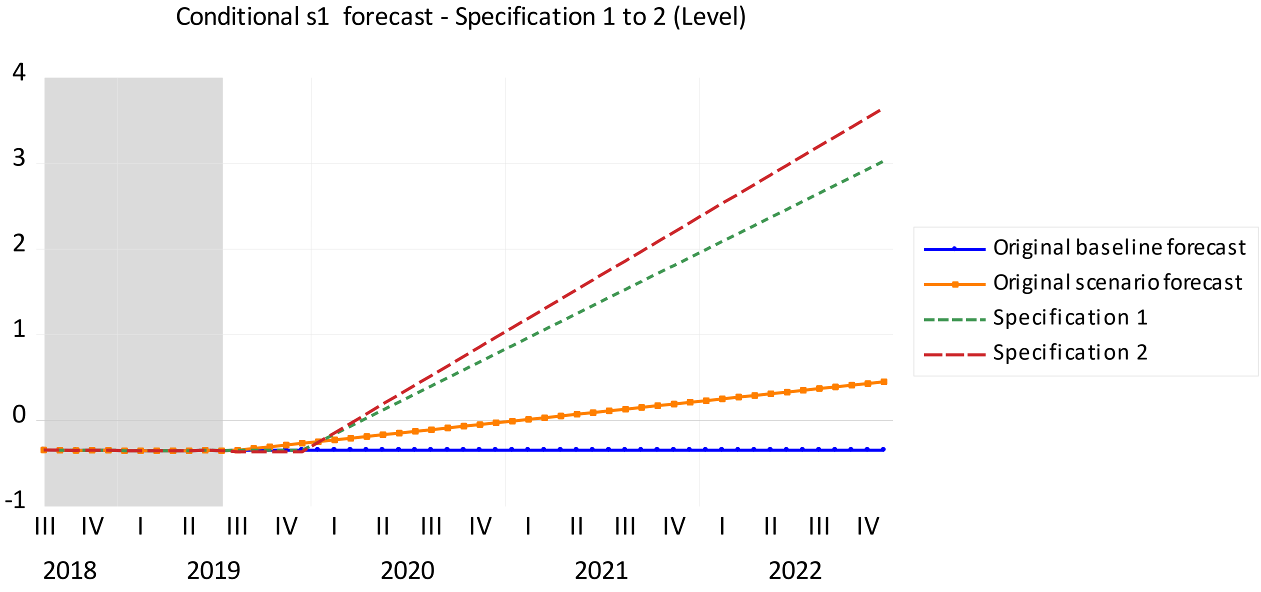
When the program is executed for multiple specifications then all the above graphs are stored in spool ‘scenario_decomposition, in individual sub-spools dedicated to each scenario named with the scenario alias (e.g. “s1”). Each scenario sub-spool is then divided into one sub-spool for level decomposition graphs and one for scenario difference decomposition graphs, named ‘level’ and ‘difference, respectively.

Multiple specification forecast graphs

Final category of conditional scenario forecasts graphs are multiple specification conditional scenarios forecast graphs, located in subspoool ‘conditional_scenarios_ms’. As before, there is one spool for each scenario, and there are two types of graphs: one graph in terms of level and one in terms of transformation. Each graph is named according to specifications it contains, e.g. ‘specs_1to5’ respectively ‘specs_1to5_trans’ in the spool, and ‘gp_csf_mslevel_bl_1to5’, respectively ‘gp_csf_mstrans_bl_1to5’ in the workfile.

The graphs contain forecast for given scenario and transformation for multiple (up to 5) specifications at a time, see Figure 10. This allows for direct comparison of conditional scenario forecasts from alternative specifications. Each specification forecast uses distinct line pattern and color, but no symbol. In addition to these conditional forecasts the graphs can also contain original baseline and given scenario forecasts, if specified and supplied by user. These forecasts use solid pattern and specific symbol.

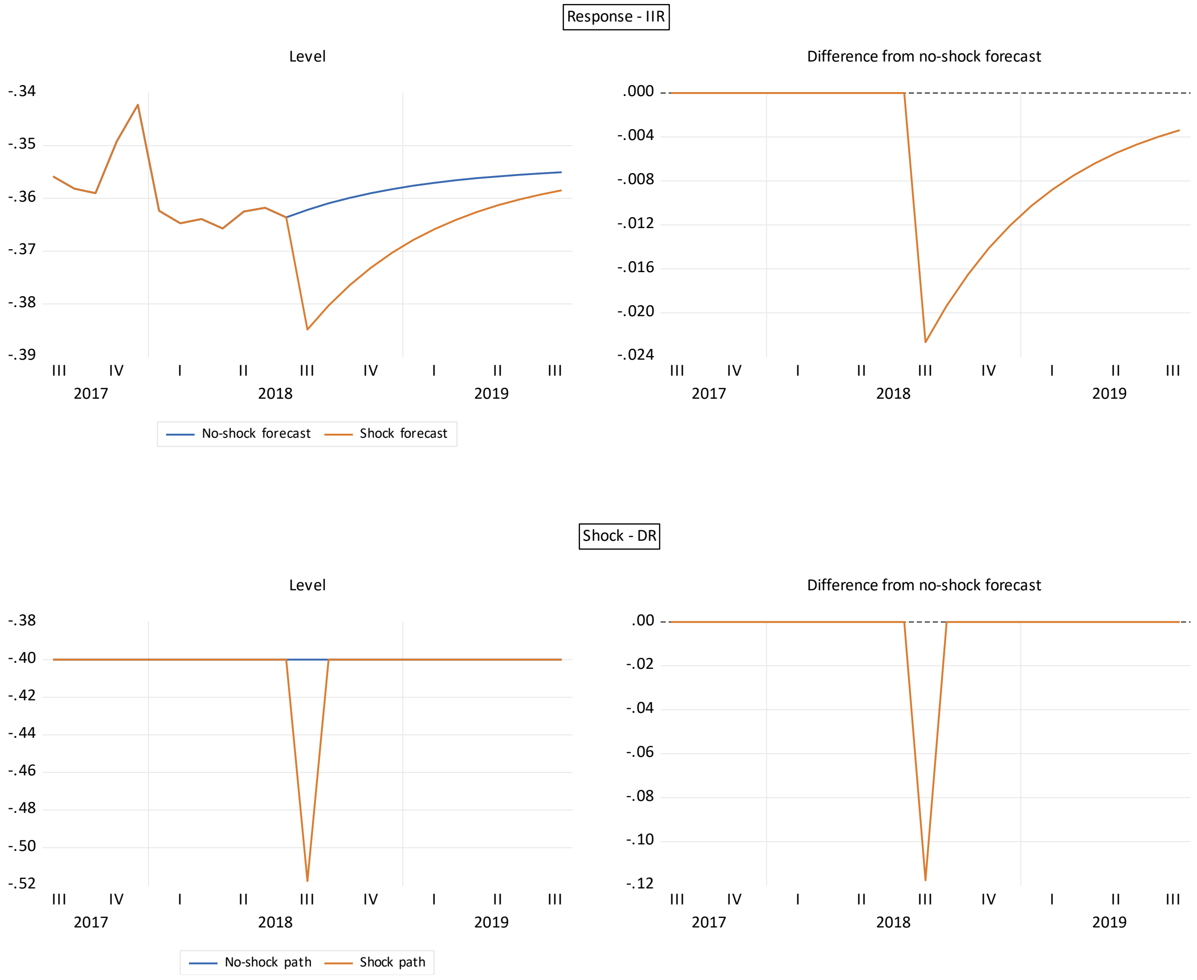
Figure 10: Multiple specification conditional scenario forecast graph



Shock response graphs

Shock response graphs display the response of the base variable to shocks in independent variables and/or regressors.[9] See Figure …. In both cases the graphs show the forecast for base variable in absence of shock and forecast for base variable in presence of shock to single independent variable/regressor (top panels), and the path for independent variable/regressor in absence of shock and with shock. The graphs are stored as 'gp_sr_{shock_varaible}', respectively 'sr_{shock_varaible}', in case of independent variable shocks; and as 'gp_sr_reg{regressor_number}', respectively 'sr_reg{ regressor_number }', in case of regressor shocks.

Figure 11: Shock response graphs



When the program is executed for multiple specification then the graphs are stored in in sub-spool 'shock_responses', in sub-spools 'varaibles' and 'regressors' for shocks to independent variables, respectively regressors.

[9] The difference between independent variable and regressor is that regressor can be transformation of multiple independent varaibles, e.g. in equation y=a+b*x/z, there are two independent variables x and z, and one rergressor x/z. Of course, in many situations independent variables and regressors coincide, in which case the graphs will be indentical.

Additional workfile objects

Apart from the graphs, tables and spool objects discussed in previous sections, the program can store also additional objects in the workfile. This section describes these objects. Each object

Forecast objects

User can indicate that the forecasts created by the add-in should be stored, by setting 'keep_forecasts="t"'. If this setting is used then the workfile will include individual backtesting forecasts and individual scenario forecasts.

Individual bactesting forecasts are stored with following naming convention: '{VAR}_{TSTART}'. Therefore, forecast for variable 'IIR' starting in 2005M02 (i.e. 2005M02 is the first month when forecast differs from observed history) is called 'IIR_F2005M02'.

Individual conditional scenario forecasts are stored with following naming convention: {VAR}_CSF{S}. Therefore, forecast for variable IIR for scenario SU will be stored as 'IIR_CSFSU'.

The naming conventions are followed irrespective of whether the "IIR" is a stand-alone dependent variable, or if it is used in transformation.

Equations

User can indicate that the re-estimated equations created by the add-in should be stored by setting 'keep_equations="t"'.[10] If this setting is used then the workfile will include individual equations used in the creating backtesting forecasts.

The individual equations are stored with following naming convention: {equation_name}_reest{fstart}. Therefore, if original equation name is 'EQ_IIR' and the forecast stars in 2005M02 (i.e. 2005M02 is the first month when forecast differs from observed history) then the equation is called 'EQ_IIR_F2005M02'.

Settings objects

User can indicate that the settings objects should be stored by setting 'keep_settings="t"'. If this setting is used then the workfile will include multiple objects with settings specified by the user. For objects that directly correspond to settings options the naming convention is in most cases following: 'st_{settings_option}', e.g. settings option 'horizons_bias' is stored in object with name 'st_horizions_bias'. The full list of objects in alphabetical order, together with setting option to which they correspond and/or further explanation, is below:

- sc_add_eq_count = number of additional equations specified in eq_list_add
- sc_horizons_bias_n = number of bias horizons
- sc_horizons_graph_n = number of graph horizons
- sc_horizons_metrics_n = number of metrics horizons
- sc_spec_count = number of specifications

[10] This is applicable only forecasts are created out of sample; otherwise only the original equation is used for creating forecasts.

- sc_subsample{N}_length = number of periods in subsample {N}
- sc_subsample{N}_start = id of forecast corresponding to subsample {N}
- sc_subsample_count = number of subsamples
- st_add_scenarios = add_scenarios
- st_auto_selection = auto_select
- st_base_var = base_var; if not specified then it is the base variable determined by the add-in.
- st_custom_reestiamtion = custom_reest
- st_eq_list_add = eq_list_add
- st_exec_list = final processed list of execution arguments
- st_exec_list_user = exec_list
- st_forecast_depvar = forecast_depvar
- st_forecasted_ivariables = forecasted_ivarables
- st_graph_add_backtest = graph_add_backtest
- st_graph_add_scenarios = graph_add_scenarios
- st_graph_benchmark = graph_bench
- st_horizons_bias = horizons_bias
- st_horizons_graph = horizons_graph
- st_horizons_metrics = horizons_metrics
- st_ignore_errors = ignore_errors
- st_include_baseline = include_baseline
- st_include_bias = indicator if bias is included among performance metrics
- st_include_descriptions = indicator if descriptions should be included in the exported PDF
- st_include_mae = indicator if MAE is included among performance metrics
- st_include_original = include_original
- st_include_rmse = indicator if RMSE is included among performance metrics
- st_index_period = index_period
- st_keep_equations = keep_eqs
- st_keep_forecasts = keep_forecasts
- st_keep_information = keep_info
- st_keep_objects = keep_objects
- st_keep_settings = keep_settings
- st_model_name_add = model_name_add
- st_outofsample = OOS
- st_percentage_error = perc
- st_performance_metrics = metrics
- st_save_output = indicator whether final spool should be exported as PDF
- st_scenario_dataload = scenario_dataload
- st_scenarios = scenarios
- st_spec_alias_list = alias_list
- st_specification_list = spec_list

- st_subsample{N} = string of sub-sample N
- st_subsample{N}_end = ending date of sub-sample N
- st_subsample{N}_start = starting date of sub-sample N
- st_subsamples = subsamples
- st_tfirst_backtest_user = tfirst_test
- st_tfirst_graph_user = tfirst_graph
- st_tfirst_scenarios =tfirst_scenarios; if not specified then equal to value determined by add-in
- st_tfirst_sgraph = tfirst_sgraph; if not specified then equal to value determined by add-in
- st_tlast_backtest_user = tlast_test
- st_tlast_graph_user = tlast_graph
- st_tlast_scenarios = tlast_scenarios
- st_transformation = trans
- st_use_names – use_names

Information objects

User can indicate that the information objects created by the add-in should be stored by setting ‘keep_info=”t”’. If this setting is used then the workfile will include individual objects containing information about the specification evaluation process. The full list of objects in alphabetical order and their information content is below:

- st_auto_type = type of automatic model selection (ARMA, ARDL or VAR)
- st_estimation_sample = estimation sample used in the original equation
- st_exog_variables = list of variables that are treated as exogenous in conditional scenario forecasting
- tb_forecast_numbers = table containing mapping between forecast numbers and starting periods of forecasts
- st_missing_variables_{scenario} = list of variables with missing scenario values for given scenario
- tb_sb_{equation_name} = sample boundaries for regressors used in given equation
- st_tfirst_backtest = first period when backtesting sample (i.e. period in which first forecast is created)
- st_tfirst_estimation = first period of estimation sample in original equation; acts as lower bound for tfirst_backtest
- st_tlast_backtest = last period when backtesting sample (i.e. period in which first forecast is created)
- st_tlast_estimation = last period of estimation sample in original equation

Intermediate objects

User can indicate that the intermediate objects created by the add-in should be stored by setting ‘keep_objects=”t”’. If this setting is used then the workfile will include individual objects containing intermediate outputs from the specification evaluation process.

The intermediate objects can be separated into several groups. First, there are following graph objects discussed above:

- gp_forecasts_all_h{horizon} = forecast summary graphs for given horizon
- gp_forecast_subsample{ID} = subsample forecast graph for given subsample
- gp_forecast_bias_h{horizon} = forecast bias graphs for given horizon
- gp_csf_level_{scenario} = conditional scenario forecast level graph for given scenario
- gp_csf_trans_{scenario} = conditional scenario forecast transformation graph for given scenario
- gp_csf_fd_{scenario} = conditional scenario forecast decomposition graph for given scenario
- gp_csf_fd_{scenario} = conditional scenario forecast difference decomposition graph for given scenario
- gp_csf_mslevel_{scenario}_{specifications} = multiple specification conditional scenario forecast level graph for given scenario
- gp_csf_mslevel_{scenario}_{specifications} = multiple specification conditional scenario forecast transformation graph for given scenario
- gp_coef_stability = coefficient stability graph
- gp_lag_orders = recursive lag orders graph

Second, there are vector and matrix objects which contain values used for calculation of forecast performance metrics or creating forecast bias graphs:

- m_fb_h{horizon} = matrix with forecast and actual values for given variable (used for forecast bias graphs)
- v_{var}_fe_h{horizon} = vector with forecast errors for given variable and given horizon
- v_{var}_fe_ss{ID}_h{horizon} = vector with forecast errors for given variable and given horizon and subsample with given ID
- v_{var}_rmse = vectors with RMSE for each of the metrics horizons
- v_{var}_{metric}_ss{ID} = vectors with forecast performance metrics for each of the metrics horizons for subsample with given ID

Third, there is table with performance metrics (‘tb_performance_metrics’) and table with adjusted regression output (‘tb_reg_output’).

Fourth, there are series with recursive coefficients (‘s_coef_{ID}’) and standard errors (‘s_serrros{ID}’).

Finally, there is model object which contains the model used for creating all forecasts (‘m_speceval’).

Appendix – Documentation of forecasting procedures

This appendix describes the estimation and forecasting procedures as performed by the add-in. It is separated into section for backtest forecasting procedure and conditional scenario forecast procedure.

Backtest forecasting procedure

The backtest forecasting procedure needs to be divided into two steps: (possible) re-estimation step, and forecasting step. The re-estimation step applies to situation when the forecasts are made out of sample, but not when they are made in-sample. Correspondingly, I will first discuss the forecasting step, and only later the re-estimation step. Throughout I will consider the situation of following general equation:

$$f(y_t) = \beta_0 + \beta_1 g(y_{t-1}) + \beta_2 x_t + \ \beta_3 \, h(z_t) + \epsilon_t$$

The equation captures the idea that the *dependent variable* - $f(y_t)$ - is not the same as the *base variable* - y_t; that the base variable can be present in lags on the right-hand side; and that the right-hand side can include additional *independent variables*, possible in transformation.

When creating (possibly) multistep forecasts for left-hand side variable one needs to specify three things:

1. The past values of base variable, if (implicitly or explicitly) present on right hand side
2. Current and past values of independent variables, if present
3. Values of current and future shocks

The backtest forecasting procedure is (a-priori) focused on dynamic conditional forecasts. Conditional in the sense that the forecasts are made *conditional on actual historical values* of all independent variables. Dynamic in that any lagged values of base variable are replaced with forecasts, when relevant. And the shocks are set to zero. Finally, note that irrespective of whether the depend variable is directly the base variable or not, the add-in constructs and performs all following procedures on the base variable itself. Out of the three aspects currently only[11] the conditionality is something the user can influence by including equations/identities for additional independent variables through option ‘eq_list_add’. Hence if the list of added equations includes also equations for x_t, but not for z_t, then the forecasts will be conditional on z_t, but unconditional with respect to x_t.[12]

The forecasts are made for made for each available start and end date, referred to as the backtest start and end period. Unless specified by the user (‘tfirst_test’ and ‘tlast_test’) these are determined by data availability. Specifically, the backtest start date is equal to first period in which all regressors have values, or period in which the estimation of the original equation started. Meanwhile, the backtest end date is

[11] Of course, in most but not all cases the user can deal with this limitation on the forecasting of base variable by changing the dependent variable from explicit to implicit, i.e. replacing the transformed base variable with new variable equivalent to the transformed variable.

[12] Of course, if equation for x_t includes additional independent variables not present in equation for y_t, then the system of forecasts will be also conditional on these additional variables.

equal to last period in which all regressors have value.[13] All the dates between backtest start and backtest end periods then constitute a starting period for a forecast, and correspondingly one forecast is then created for each forecast start period. Irrespective of forecast start period (or other parameters) the forecasts end date is the backtest end date.

In case out of sample forecast one cannot use the coefficient estimates based on full estimating sample, since in such situation the forecasts would use information available only after start of the forecast. Therefore, in addition to the above forecasting procedure one needs to also perform re-estimation procedure that ensures that the forecast is created by model that was based only on information available before the start of the forecast. In the above equation this amounts to using different values for coefficients $\beta_0, \beta_1, \beta_2$ and β_4. The obtain these estimates the above equation is re-estimated on adjusted estimation sample. Specifically, consider equation that is estimated on sample starting in backtest start period t=0, and ending in backtest end period t=T, and consider forecasts that starts in period $\underline{t}$. Then the equation is re-estimated on sample going from period t=0 until period t=$\underline{t}$-1. The re-estimated equation is then used for the creating the dynamic conditional forecasts starting in period $\underline{t}$ and ending in period T using procedure as outlined above. [14] Note that the need to re-estimate the equation means that the backtest start period is adjusted relative to what was outlined for in-sample forecasts above. Specifically, to avoid the analysis being unduly influenced by early forecasts made based on equations with too short estimation sample – and hence likely noisy coefficients – the backtest start period is shifted so that there are 2 observations for each estimated coefficient.

The re-estimation can face important challenges. Most importantly, some equations cannot be estimated on shorter samples because their structure leads to perfect multicollinearity problems. These can arise in two different reasons:

- The equation might include constant together with variable that has zero variance on initial part of the original estimation sample (but not on the whole original estimation sample, otherwise it would not be possible to estimate even the original equation).[15] If it is attempted to re-estimate the equation on the sample where the variable does zero variance the estimation will fail due to perfect collinearity between the constant and the zero-variance variable.
- The equation might include two regressors that do vary in the initial part of the original estimation sample, but they are perfectly collinear on such sample.

[13] The sample boundaries of regressors are stored in 'tb_sb_{equation_name}'. Note that when the equation is estimated on full available sample then the backtest start and backtest end will correspond to starting and ending date of estimation. The reason why the estimation start is treated as lower bound is because presumably by adjusting the estimation start date the modeler is indicating that the model does not apply to the period before. Since this argument does not apply to estimation end, it is not treated the same.

[14] Such re-estimation and resulting forecasts are often called *recursive*, to distinguish them from *rolling* forecast. In rolling re-estimation/forecasting the estimation window is being kept at fixed size specified by the user.

[15] A typical example is when one of the variables is a dummy variable. Since dummy variable is zero before the first occurrence of the event it represents, it can often happen that the re-estimation sample includes period when it has zero variance.

This raises an operations problem: such issues will cause estimation to fail, which could prevent the operation of the remaining parts of the add-in. The add-in addresses this issue in two ways. First, by default[16] the add-in checks specifically for the two issues outlined above and addresses them. In the first case it eliminates the constant unless the zero-variance variable has values equal to zero, in which case it eliminates that variable.[17] In the second case it eliminates the variable that is second in the specification list. However, this approach does not cover all possible reasons for estimation crashes.[18] Hence, the add-in offers second way to avoid such crashes preventing the remainder of the execution: user can specify that estimation errors should be ignored ('ignore_errors="t"') in which case the add-in will use last re-estimated equation when equation was not successfully estimated on current sample.[19]

Conditional scenario forecast procedure

To create conditional scenario forecasts needs to specify the same three things as in case of backtest forecasting procedure:

1. The past values of base variable, if (implicitly or explicitly) present on right hand side
2. Current and past values of independent variables, if present
3. Values of current and future shocks

The treatment of the first and third aspect is the same as in backtest forecasting procedure: the forecasts are dynamic (i.e. forecasted values are used for future past values of base variables) and made under assumption of zero future shocks. Similarly, the forecasts are made in terms of the underlying base variable, rather than the actual dependent variable.

In terms of current and past values of independent variables the forecasting procedure cannot rely on using actual observed historical values. Instead, the procedure uses scenario values for all independent variables, if avaiable.[20] Therefore, the scenario forecasts for the base variable are conditional in that they are made based on scenario values of independent variables. Again, the user can control the degree of conditionality by including equations/identities for additional variables.

The sample for which the forecasts are made again follows the data availability, if not adjusted by the user. The starting date is the first period after last date in which all regressors have historical values. The

[16] This can be switched off by setting option 'eliminate_multicol="f"'.

[17] The logic for this rule is following. Since the modeler included the variable in the regression it is presumable linked to the dependent variable. This link should be present even when the independent variable has zero variance. A typical example will be link between different interest rates. However, this rule cannot be followed when all the values are zero – a typical situation for dummy variable - since in such case the equation cannot be estimated.

[18] For example, the add-in check for perfect correlation only between all pairs of regressors. A perfect multicollinearity can also arise in situations when no two regressors are perfectly correlated, for example in case of set of several dummies covering all cases or in situation of multiple lags.

[19] Note that this allows user to ignore only errors that can be ignored by Eviews itself. For example, in BREAKLS estimation method Eviews stops whenever perfrect collinearity was encountered irrespective of error handling.

[20] If the scenario forecasts are not available for given variable and scenario then the no-scenario forecast is used instead. To see which values have been used for each variable either consult the scenario in the model object or check the string 'st_missing_variables_{scenario}'.

ending data is either the end date of the workfile, or last date for which values for all variables are available.