OxMetrics version 8

OxMetrics 8

OxMetrics is a powerful system for econometric, statistical, and financial econometric modelling and forecasting. OxMetrics 8 continues a long tradition of reliable and easy-to-use software.
The user-friendly OxMetrics front-end is shared between all OxMetrics modules, thus considerably reducing the learning curve when using the full range of econometric and statistical functionality of the OxMetrics system.
The modules have been developed by experts in their field: many of their advances have been published in learned journals, while at the same time benefiting the OxMetrics system.
We first present an overview of the components of the entire OxMetrics system, before more detailed descriptions of each module.

Contents

1 OxMetrics overview

OxMetrics is a family of software packages providing an integrated solution for the econometric analysis of time series, forecasting, financial econometric modelling, or statistical analysis of cross-section and panel data. OxMetrics consists of a front-end program called OxMetrics, and individual application modules such as Ox, CATS, PcGive, STAMP and G@RCH.
OxMetrics Enterprise is a single product that includes all the important components: OxMetrics desktop, Ox Professional, CATS, PcGive and STAMP and G@RCH

The OxMetrics front end provides facilities to manage and transform the **databases** that are used in the statistical analysis. Output from OxMetrics modules is displayed in the form of **graphics and reports**. Graphs can be edited in preparation for publication. Multiple plots within a graph are possible, with automatic adjustments for smaller sizes.
OxMetrics databases for macro-economic analysis have a **fixed frequency**. Databases for financial econometrics are usually 'dated', which allows for **daily or timed data**. Aggregation facilities are provided, e.g. to convert daily data into monthly.
Transformations can be done using a calculator, or with algebra code. A **batch language** allows for repetition of tasks, and the batch code is recorded in the background while interactive modelling proceeds.
All modelling components have been written in the **Ox language**.

Ox is an object-oriented statistical and econometric development system. At its core is a powerful matrix language, which is complemented by a comprehensive matrix and statistical library. Among the special features of Ox are its speed, well-designed syntax, and graphical facilities. Ox can read and write many data formats, including spreadsheets and OxMetrics files.
Ox is at the core of OxMetrics: most of the interactive modules of OxMetrics (such as PcGive, STAMP, G@RCH) are implemented with the Ox language.

PcGive aims to give an operational and structured approach to **econometric modelling** and **forecasting** using the most sophisticated yet user-friendly software. The accompanying books transcend the old ideas of 'textbooks' and 'computer manuals' by linking the learning of econometric methods and concepts to the outcomes achieved when they are applied. The econometric techniques of PcGive include: VAR, cointegration, simultaneous equations models, Markov Switching, ARFIMA, logit, probit, GARCH modelling, static and dynamic panel data models, X12ARIMA, and more.
PcGive uses **Autometrics** for automatic model selection.
PcGive incorporates **PcNaive** to interactively design and run Monte Carlo experiments.

CATS is dedicated to the cointegrated vector autoregression. Both **I(1)** and **I(2)** models can be estimated using the most sophisticated algorithms. **Restrictions** can be imposed on the cointegrating vectors and their loadings. Both Bartlett corrections and **bootstrap** versions of tests are available. Models can be estimated recursively. **Random samples** can be drawn from the estimated models, making it easier to implement simulation experiments. **CATSmining** helps with identification of the cointegrating space.

G@RCH is dedicated to the estimation and forecasting of **univariate and multivariate ARCH-type models**. It also allows the estimation of univariate and multivariate non-parametric estimators of **quadratic variation** and **integrated volatility**. G@RCH provides a menu-driven easy-to-use interface, as well as graphical features. For repeated tasks, the models can be estimated via the Batch Editor of OxMetrics or using the Ox language together with the 'Garch', 'MGarch' and 'Realized' classes.

STAMP stands for Structural Time series Analyser, Modeller and Predictor. The models are set up in terms of unobserved components such as trends, seasonals and cycles, which all have a direct interpretation. STAMP gives access to sophisticated algorithms through an easy-to-use interface.
Univariate decompositions can be specified with higher-order trends and cycles to force these components to be more smooth; the signal-to-noise ratio can be fixed but also estimated. This option in STAMP is regularly used for extracting business and financial cycles from macroeconomic time series, even when portions of the time series are missing.
Realized-variance time series can be decomposed into multiple stochastic components such as long-term variance (random walk or smooth trend), trading-day effects (seasonal component), stationary volatility (autoregressive) and noise (irregular component). The multiple components can be extracted from the data for analysis and forecasting.
Multivariate decompositions of macroeconomic time series are relevant for economic policy making when it is recognized that the variables are endogenous. This is fully supported in STAMP.

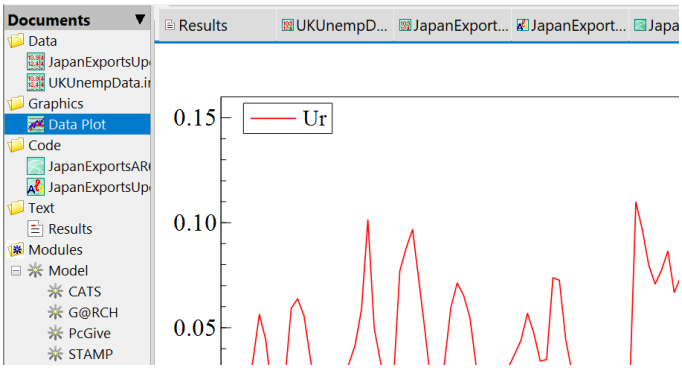
SsfPack is a library to carry out computations involving the statistical analysis of univariate and multivariate models in state space form. SsfPack routines can be called from Ox, while the library itself is written in C.
SsfPack is not a member of OxMetrics Enterprise Edition.

Versions

- **OxMetrics Enterprise Edition** is a single product that includes and integrates all the important components for theoretical and empirical research in econometrics, time series analysis and forecasting, applied economics and financial time series: OxMetrics front-end, Ox Professional, PcGive, G@RCH, CATS, and STAMP.
- **OxMetrics Modules** are available individually, always including the OxMetrics front-end and Ox Professional.
- **SsfPack** is a separate program.

Recent editions of macOS, Windows and Linux are supported.

2 OxMetrics



Data formats

- **OxMetrics** data files (.in7/.bn7) — this format is designed to make reading and writing of data very efficient;
- **Excel** spreadsheet files (.xlsx);
- **Spreadsheet** Comma-separated files (.csv);
- **Other** Stata 13 and 14 .dta files; formatted text files.

Graphics

- **Actual series** with optional transformations
  - Create separate Plots
  - Style
    - Lines
    - Symbols
    - Lines and symbols
    - Index line: plot first series as index
    - Index line and symbols: plot first series as index
    - Bars: plot all series as bars
    - Shading: use shading where this variable is 1, no shading otherwise
    - First as bar: plot only the first as bar chart
  - Transformation:
    - Logarithms: natural logs of the series: $\log(y_t)$
    - Growth rates: $\log(y_t - y_{t-1})$
    - First differences: $\Delta y_t \square y_t - y_{t-1}$
    - Seasonal growth rates: $\log(y_t - y_{t-s})$, $s \square 4$ for quarterly data, $s \square 12$ for monthly data,
    - Seasonal differences: $\Delta_s y_t \square y_t - y_{t-s}$
  - Use log scale.
- **Multiple series** with optional transformations
  - Match series by
    - None: no matching is done
    - Mean & range matched to first series
    - Second series on right scale
    - Start = 100
  - Style: as above
  - Transformation: as above
  - Use log scale
- **Scatter plots**
  - Y against X
  - Y against X, labels along the axes
  - Scatter plot with regression line
  - With cubic spline smooth, automatic bandwidth
  - All scatter plots
- **Distribution plots**
  - Estimated density and histogram, optionally with normal reference
  - Estimated distribution against normal: a QQ plot
  - Frequencies and/or cumulative frequencies
  - QQ plot against Uniform, normal, t, F, or χ^2 distribution
  - Box plot
- **Time-series plots**
  - Autocorrelation function
  - Partial autocorrelation function
  - Cross-correlation function
  - Periodogram
  - Spectral density
  - Seasonal sub-plot
- **QQ plots**
  - Quantile plot: against uniform
  - QQ plot against normal (same mean, variance)
  - QQ plot with choice of distribution
- **Two series by a third**
  - Error bars
  - Error bands
  - Error fans
  - High-low
  - Show Z values
  - As symbol size: Bubble chart
- **3-dimensional plots**
  - Surface from scatter: X,Y,Z are vectors
  - Triangulation from scatter: X,Y,Z are vectors
  - Surface from table: Z,Y columns match
  - Surface from table: Z,X columns match
  - 3D points
  - 2D contour from 3D scatter surface

Graph editing and saving

Each graph consists of a collection of objects, which in most cases can be manipulated, moved or deleted.
Graphs can be saved as:

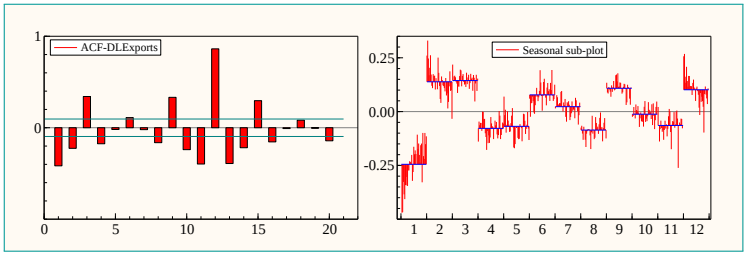
extension	format
`.eps`	Encapsulated PostScript;
`.gwg`	*OxMetrics* graphics file;
`.pdf`	Portable document format;
`.png`	a bitmap format;
`.ps`	PostScript;
`.svg`	SVG, supported by most browsers.

Transformations

The **Algebra language** enables you to transform database variables by writing mathematical formulae. Algebra code can be written interactively in the Calculator, or directly in the Algebra editor. Such algebra code can be saved, reloaded, and edited.
The **Calculator** writes its operations as algebra code to the Results window, from where it can be cut and pasted into the algebra editor. Algebra can also be run directly from the results window, by highlighting the block of algebra code, and then pressing Ctrl+A.

Modules

An increasing number of modules interacts with OxMetrics: the front-end is the 'server', while the modules (G@RCH, PcGive, STAMP, etc.) are the 'clients'. While it is possible to write clients that interface directly with the server (such as OxPack, and OxRun), it is much easier to develop Ox packages to do this. This requires the use of the Modelbase class, which provides the necessary functionality.



3 PcGive

The **special features** of PcGive are:

1 *Ease of use* – all modelling can be done interactively.
2 *Flexible data handling in OxMetrics*.
3 *Efficient modelling* – fast and reliable algorithms written in Ox.
4 *Automatic model selection* – **Autometrics** is available for many model types to provide a sophisticated model selection tool.
5 *Advanced graphics* – graphic analysis of the estimated model, recursive graphics, forecast graphics and others.
6 *Powerful evaluation* – statistical tests to evaluate model adequacy.
7 *Extensive batch language* together with generation of Ox code for more advanced uses.
8 *Well-presented output*.

Capabilities

PcGive supports the following model categories (book volume in parentheses):

- **Models for cross-section data**
  - Cross-section Regression (I)
- **Models for discrete data**
  - Binary Discrete Choice (III): Logit and Probit
  - Multinomial Discrete Choice (III): Multinomial Logit
  - Count data (III): Poisson and Negative Binomial
- **Models for financial data**
  - GARCH Models (III): GARCH in mean, GARCH with Student-t, EGARCH, Estimation with Nelson&Cao restrictions
- **Models for panel data**
  - Static Panel Methods (III): within groups, between groups
  - Dynamic Panel Methods (III): Arellano-Bond GMM estimators
- **Models for time-series data**
  - Single-equation Dynamic Modelling (I), optionally using *Autometrics*
  - Multiple-equation Dynamic Modelling (II): VAR, cointegration, simultaneous equations analysis, optionally using *Autometrics*
  - Regime Switching (V): Markov-switching models
  - ARFIMA Models (III): exact maximum likelihood, modified-profile likelihood or non-linear least squares
  - Seasonal adjustment using X12ARIMA (III): regARIMA modelling, Automatic model selection, Census X-11 seasonal adjustment.
- **Monte Carlo using PcNaive**
  - AR(1) Experiment using PcNaive (IV)
  - Static Experiment using PcNaive (IV)
  - Advanced Experiment using PcNaive & Ox Professional (IV)
- **Other models**
  - Nonlinear Modelling (I)
  - Descriptive Statistics (I):
    - Means, standard deviations and correlations
    - Normality tests and descriptive statistics
    - Autocorrelations (ACF) and Portmanteau statistic
    - Unit-root tests
    - Principal component analysis

Documentation

PcGive documentation consists of 5 books. These explain the econometric methods, the modelling approach, and the techniques used, as well as bridge the gap between econometric theory and empirical practice.

I Single-equation modelling
II Multiple-equation modelling
III Other methods
IV PcNaive
V Markov-switching models

Autometrics

Autometrics is our realization of **automatic model selection**, designed to handle these challenges:

- **Automatic**: computer is a powerful modelling aid,
- **General to specific**: can maintain desirable econometric properties,
- **Extensive search**: to handle correlated data,
- **Efficient search**: need to estimate many models,
- **Statistical congruence**: maintained as a search constraint,
- **Statistical properties**: extensively researched,
- **Not maximizing goodness-of-fit**: avoids overfitting,
- **Controlled through gauge**: expected number of falsely selected variables,
- **Flexible**: more variables than observations, logit models, ...
- **Robustness**: to outliers using impulse-indicator saturation (**IIS**) and breaks using step-indicator saturation (**SIS**).

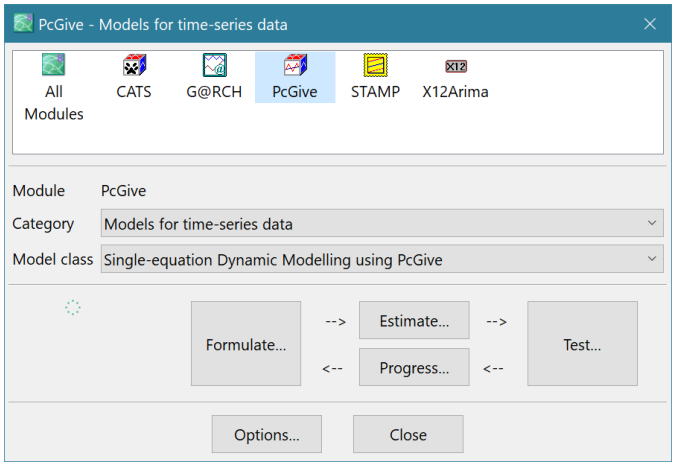
Also see Hendry and Doornik (2014), *Empirical Model Discovery and Theory Evaluation*, MIT Press.

PcNaive

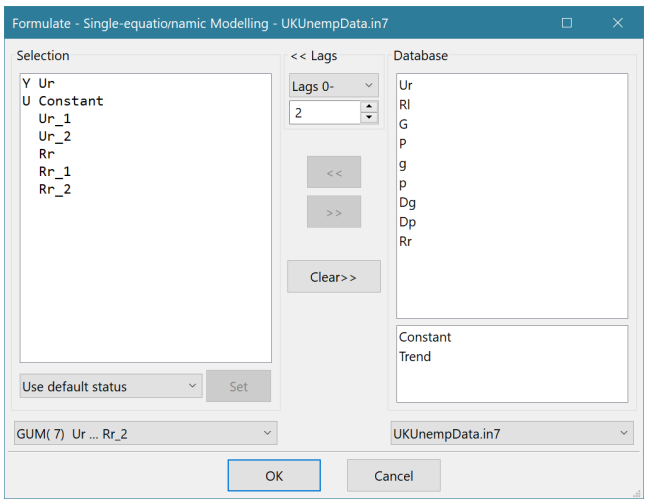
PcNaive is an **interactive** program for **Monte Carlo** study of econometric methods. Experiments allow the finite-sample properties of dynamic econometric methods to be evaluated in relevant settings.
Experiments based on the **simple AR(1) DGP** and **static DGP** can be formulated in PcGive, and run directly in OxMetrics.
Advanced experiments, on the other hand, are formulated interactively. PcNaive then writes the **Ox** code which is executed in OxMetrics by **OxRun**.

Example

This example uses the UK unemployment rate (*Ur*) and the real interest rate minus the real growth rate (*Rr*, see Clements & Hendry (2006) Ch.12 in *Handbook of Economic Forecasting*, Vol.1).
After loading the database in OxMetrics, start modelling using PcGive:



Formulate the model with two lags:



using IIS and SIS, and keeping back 8 forecasts:

Model Settings - Single-equation Dynamic Modelling

Choose the Autometrics options:

Automatic model selection	☑
Target size	Tiny: 0.001
Pre-search lag reduction	☑
Outlier and break detection	
None	○
Large residuals	○
Saturation estimation	◉
Impulse indicator saturation (IIS)	☑
Step indicator saturation (SIS)	☑
Differenced IIS (DIIS)	☐
Trend saturation (TIS)	☐
Advanced Autometrics settings	☐

Estimate - Single-equation Dynamic Modelling

Choose the estimation sample:

Selection sample	1863 - 2014
Estimation starts at	1863
Estimation ends at	2014
Less forecasts	8

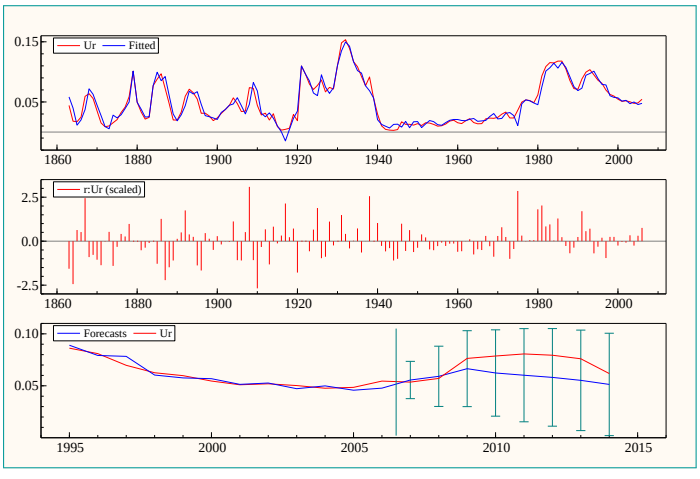
Choose the estimation method:

Estimation method:	Ordinary Least Squares
Standard errors	Standard

Estimation with *Autometrics* finds 4 impulses and 5 steps. The steps, however, combine to effectively make dummies, so we rerun with IIS only. This leaves 7 impulse dummies, three before 1890, two in the 1920's, and finally 1930 and 1939:

```
Modelling Ur by OLS
The dataset is: UKUnempData.in7
The estimation sample is: 1863 - 2006

             Coefficient  Std.Error  t-value  t-prob Part.R^2
Ur_1             1.26941    0.06397     19.8  0.0000   0.7489
Ur_2            -0.371891   0.05731    -6.49  0.0000   0.2418
Rr               0.160550   0.02089     7.69  0.0000   0.3092
Rr_1            -0.0929132  0.02214    -4.20  0.0000   0.1178
I:1879           0.0312359  0.009103    3.43  0.0008   0.0819
I:1880          -0.0484728  0.009531   -5.09  0.0000   0.1639
I:1884           0.0453289  0.009084    4.99  0.0000   0.1587
I:1921           0.0527939  0.01042     5.06  0.0000   0.1627
I:1922          -0.0487495  0.01040    -4.69  0.0000   0.1426
I:1930           0.0363700  0.009069    4.01  0.0001   0.1086
I:1939          -0.0349234  0.009160   -3.81  0.0002   0.0992
Constant  U    0.00475903  0.001469    3.24  0.0015   0.0736

sigma               0.00897131  RSS                 0.0106239429
R^2                   0.939578  F(11,132) =   186.6 [0.000]**
Adj.R^2               0.934543  log-likelihood          480.714
no. of observations         144  no. of parameters            12
mean(Ur)             0.0489454  se(Ur)                0.0350653

AR 1-2 test:      F(2,130)  =   2.8363 [0.0623]
ARCH 1-1 test:    F(1,142)  =0.0099229 [0.9208]
Normality test:   Chi^2(2)  =   9.0941 [0.0106]*
Hetero test:      F(8,128)  =   1.7521 [0.0926]
Hetero-X test:    F(14,122) =   1.1535 [0.3198]
RESET23 test:     F(2,130)  =   1.8745 [0.1576]

1-step (ex post) forecast analysis 2007 - 2014
Parameter constancy forecast tests:
Forecast  Chi^2(8)  =   2.8385 [0.9441]
Chow      F(8,132)  =  0.34906 [0.9448]
CUSUM     t(7)      =   0.4219 [0.6858]
```

The model and forecasts are:



While the forecasts are dynamic in the unemployment rate, they are conditional on the growth adjusted interest rate. So they are not genuinely out of sample.

One approach is to reformulate the model as a vector autoregression. Starting with two lags as before, it is possible to select *Autometrics* with IIS and SIS again, this time using 1%:

Choose a model type:

Unrestricted system	◉
Cointegrated VAR	○
Simultaneous equations model	○
Constrained simultaneous equations model	○

Choose the Autometrics options:

Automatic model selection	☑
Target size	Small: 0.01
Pre-search lag reduction	☑
Outlier and break detection	
None	○
Large residuals	○
Saturation estimation	◉
Impulse indicator saturation (IIS)	☑
Step indicator saturation (SIS)	☑
Differenced IIS (DIIS)	☐
Trend saturation (TIS)	☐
Advanced Autometrics settings	☐

The selected model has the same variables in each equation. To obtain a more parsimonious representation, rerun *Autometrics* at 0.1% on the simultaneous equations model with both equations in their unrestricted reduced form. Estimation is by FIML:

Choose the estimation sample:

Selection sample	1862 - 2014
Estimation starts at	1863
Estimation ends at	2014
Less forecasts	8

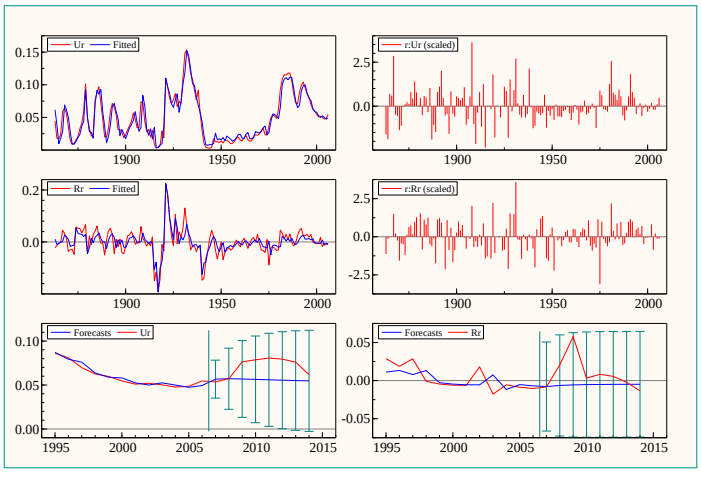
Choose the estimation method:

Full Information Maximum Likelihood (FIML)	◉
Three-stage Least Squares (3SLS)	○
Two-stage Least Squares (2SLS)	○
Equation by Equation OLS (1SLS)	○
Automatic maximization	☑
Standard errors	Standard

The first equation (using LaTeX code written by PcGive):

$$\begin{aligned}\mathrm{Ur} \;=\;& \underset{(0.055)}{1.26}\,\mathrm{Ur}_{t-1} - \underset{(0.052)}{0.374}\,\mathrm{Ur}_{t-2} + \underset{(0.0087)}{0.0306}\,\mathrm{I{:}1879}_t \\ & - \underset{(0.011)}{0.0597}\,\mathrm{I{:}1880}_t + \underset{(0.0086)}{0.0455}\,\mathrm{I{:}1884}_t - \underset{(0.016)}{0.132}\,\mathrm{I{:}1922}_t \\ & + \underset{(0.0086)}{0.0325}\,\mathrm{I{:}1930}_t - \underset{(0.0087)}{0.0354}\,\mathrm{I{:}1939}_t - \underset{(0.011)}{0.0935}\,\mathrm{S1{:}1920}_t \\ & + \underset{(0.011)}{0.091}\,\mathrm{S1{:}1922}_t + \underset{(0.0017)}{0.00597}\end{aligned}$$

The final model does have some non-normality left, but the forecasts are genuinely ex ante now:



4 CATS

The basic model of CATS is the p-dimensional vector autoregressive (**VAR**) model with Gaussian errors and k lags

$$X_t = A_1 X_{t-1} + \cdots + A_k X_{t-k} + \Phi D_t + \epsilon_t, \quad t = 1, \ldots, T,$$

where $X_0, \ldots, X_{-k+1}$ are fixed, $\epsilon_1, \ldots, \epsilon_T$ are *iid* $N_p(0, \Omega)$ and D_t is a vector of deterministic variables such as a constant, linear trend, and seasonal or intervention dummies.
The VAR(p) model is reformulated as (taking $k = 2$):

$$\Delta X_t = \Pi X_{t-1} + \Gamma_1 \Delta X_{t-1} + \Phi D_t + \epsilon_t, \quad t = 1, \ldots, T.$$

The hypothesis of cointegration is formulated as a reduced rank condition on the Π matrix,

$$\mathcal{H}(r) : \Pi = \alpha\beta',$$

where α and β are $p \times r$ matrices of full column rank. This is the **I(1) model**.
The VAR in second order differences with $\mathcal{H}(r)$ imposed is:

$$\Delta^2 X_t = \alpha\beta' X_{t-1} - \Gamma \Delta X_{t-1} + \Phi D_t + \epsilon_t, \; t = 1, \ldots, T.$$

The **I(2) model** imposes an additional rank reduction on the model:

$$\alpha_\perp' \Gamma \beta_\perp = \xi\eta',$$

where ξ and η both are $(p - r \times s_1)$-dimensional matrices.

CATS features

Here is a brief summary of new features in the I(1) part of CATS

1. Much more efficient computations (can be several orders of magnitude faster than previous implementations) in Bartlett correction and recursive estimation;
2. Bartlett correction always included when valid;
3. Improved beta-switching algorithm;
4. New alpha-beta-switching algorithm allowing linear restrictions on alpha and not requiring identification;
5. Bootstrap of rank test;
6. Bootstrap of restrictions;
7. More Monte Carlo facilities: draw from estimated model, either with estimated or with specified coefficients;
8. General-to-specific CATSmining;
9. Automatic generation of Ox code;
10. New convenient way to express restrictions;
11. Most algorithms are QR based.

And for the I(2) part of CATS:

1. Improved tau-switching algorithm;
2. New delta-switching algorithm;
3. New triangular-switching algorithm allowing linear restrictions on alpha, beta, tau and not requiring identification;
4. Estimation with $\delta = 0$;
5. Bootstrap of rank test;
6. Simulation of asymptotic distribution of rank test;
7. Bootstrap of restrictions;
8. More Monte Carlo facilities: draw from estimated model, either with estimated or with specified coefficients;
9. Automatic tests of unit vectors and variables;
10. CATSmining;
11. Improved computation of standard errors;
12. Automatic generation of Ox code;
13. All algorithms are QR based.

Documentation

The CATS documentation, written by Katarina Juselius and Jurgen Doornik, presents the models together with tutorials. The contents are

- The Multivariate Cointegration Model
- An I(1) Analysis
- CATSmining
- The Cointegrated I(2) model
- An I(2) Analysis

Formulating linear restrictions

One of the most important features of CATS is that it allows you to test or impose restrictions on the parameters α and β. Linear restrictions on the cointegrating vectors can include the desired normalization.
There are two ways, in general, to express the linear restrictions on a $p_1 \times 1$ vector. First as

$$\beta = H\varphi,$$

where H is a known $p_1 \times q$ matrix, with $q < p_1$ and φ is a column vector of length q. A second way to express restrictions is in the form $R'\beta = 0$, where R is $(p_1 - q) \times p_1$. $R = H_\perp$ moves between the representations.

CATS presents a more intuitive way to express restrictions that reflects how we report our research. For example, homogeneity between X_2, X_3 and X_4 is formulated as

```
* a  b  -a-b  *
```

This leaves the first and last elements free, but forces the penultimate to be the negative sum of the previous two so that the sum of the tree middle coefficients is zero.
When α is restricted, we collect the joint restrictions in sections, e.g. for $r = 1, p = 3, p_1 = 4$

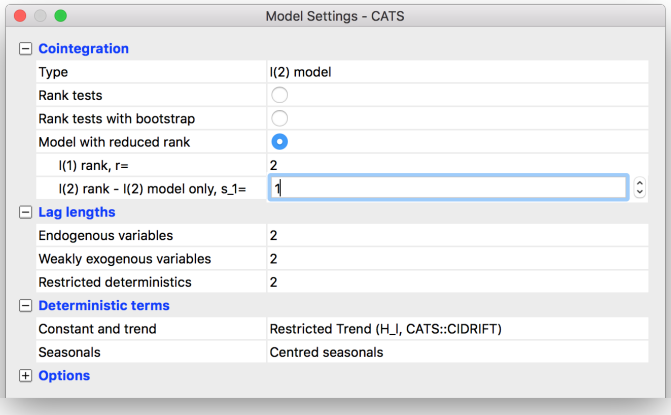
```
[alpha]
 * 0 *
[beta]
 1 0 * *
```

which corresponds to:

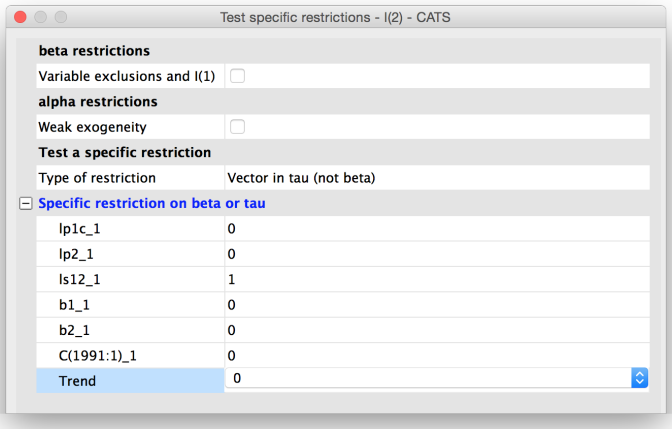
$$\alpha\beta' = \begin{bmatrix} a_0 \\ 0 \\ a_1 \end{bmatrix} \begin{bmatrix} 1 & 0 & b_0 & b_1 \end{bmatrix}.$$

Example dialogs

After selecting the variables in the VAR, the settings are used to specify the lag length and choose between I(1) and I(2) modelling. Here an I(2) model is specified:



Several types of tests are pre-programmed:



5 G@RCH

G@RCH is an OxMetrics application dedicated to the estimation and forecasting of **univariate and multivariate ARCH-type models**. G@RCH provides a user-friendly interface with rolling menus as well as many graphical features. For repeated tasks, the models can be estimated via the 'Batch Editor' of OxMetrics or the Ox programming language (several example files using the G@RCH class are provided).

G@RCH capabilities

- **Conditional mean**: ARMA, ARFIMA, ARCH-in-Mean, Explanatory Variables;
- **Univariate conditional variance**: GARCH, EGARCH, GJR, APARCH, IGARCH, RiskMetrics, FIGARCH, FIEGARCH, FIAPARCH, HYGARCH, GAS; Explanatory Variables;
- **Multivariate conditional variance**: MG@RCH: scalar BEKK, diagonal BEKK, full BEKK, DCC, cDCC, CCC, DECO, OGARCH, GO-GARCH, Principal Components, RiskMetrics, Variance Targeting;
- (Quasi-)Maximum Likelihood: Normal, Student, GED or skewed-Student distribution;
- Constrained Maximum Likelihood, Simulated Annealing;
- **Value-at-Risk**, Expected shortfall, Backtesting (Kupiec LRT, Dynamic Quantile test);
- **Forecasting**, Realized volatility;
- Tests for **jumps**;
- RE@LIZED non-parametric estimators of **quadratic variation**, **integrated volatility** and jumps using intraday data.
- h-steps-ahead **forecasts** of both equations;
- **Univariate and multivariate misspecification tests** (Nyblom, Sign Bias Tests, Pearson goodness-of-fit, Box-Pierce, Residual-Based Diagnostic for conditional heteroscedasticity, Hosking's portmanteau test, Li and McLead test, constant correlation test, and more).
- Three univariate models of the class of **Generalized Autoregressive Score (GAS) models**, i.e. the GAS, Exponential GAS, and Asymmetric Exponential GAS models with a Normal, Student-t, GED and Skewed-Student distribution.
- **Spline-GARCH** and Spline-GJR models.

Multivariate GARCH (MGARCH)

From a financial-econometric perspective, MGARCH models enable better decision tools in many areas, e.g. asset pricing, portfolio selection, option pricing, hedging, and risk management. However, implementation of such models is not easy, making G@RCH a potentially valuable tool for financial institutions.

MGARCH models for N stochastic processes can be described as

$$y_t \sim \mu_t(\theta) + \epsilon_t$$

where θ is a finite vector of parameters, $\mu_t(\theta)$ is the conditional mean vector and,

$$\epsilon_t \sim H_t^{1/2}(\theta) z_t,$$

with $H_t^{1/2}$ a positive definite matrix.

Several MGARCH models are available in G@RCH since version 5.0. The most popular one is perhaps the dynamic conditional correlation (DCC) model of Rob Engle. DCC defines:

$$H_t \sim D_t R_t D_t$$

where $D_t \sim \text{diag}(h_{11,t}^{1/2}, \ldots, h_{NN,t}^{1/2})$ and $h_{ii,t}^{1/2}$ can be taken as any univariate GARCH model. R_t is a correlation matrix derived from a covariance matrix $Q_t \sim (q_{ij,t})$:

$$R_t \sim \text{diag}(q_{11,t}^{-1/2}, \ldots, q_{NN,t}^{-1/2}) Q_t \text{diag}(q_{11,t}^{-1/2}, \ldots, q_{NN,t}^{-1/2}),$$

specified as

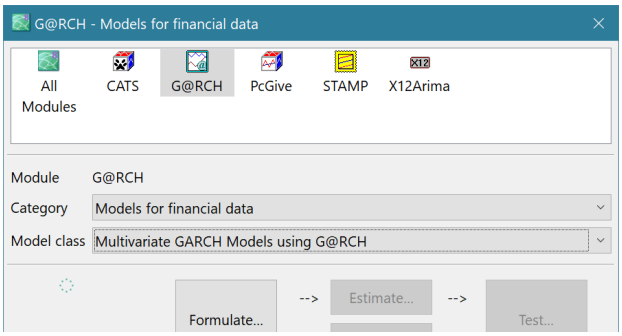
$$Q_t \sim (1 - \alpha - \beta)\overline{Q} + \alpha u_t u_t' + \beta Q_{t-1}.$$

A convenient feature of DCC models is that the parameters governing the variance and correlation dynamics can be estimated separately.

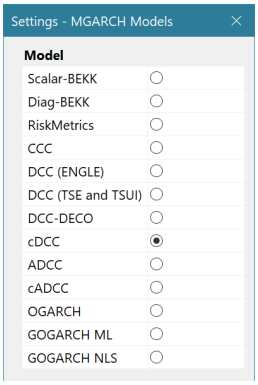
Aielli (2013, *Journal of Business & Economic Statistics*) presents a corrected DCC model, which addresses the inconsistency in the estimation of the unconditional variance matrix. This is available in G@RCH as cDCC.

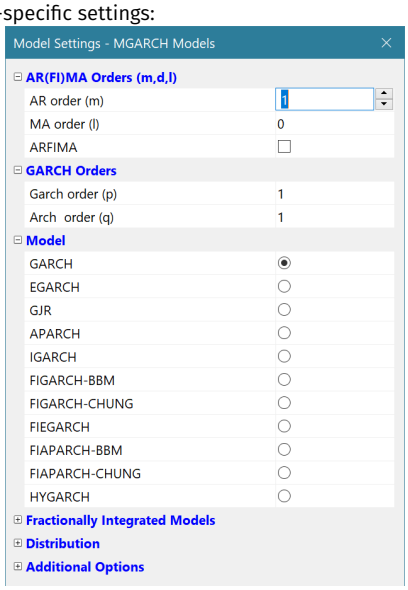
MGARCH Example

The bivariate GARCH(1,1)-cDCC example is based on the Dow Jones and Nasdaq indices. After loading the `DJNQ.xls` data set in OxMetrics, activate the model dialog selecting MGARCH models with G@RCH:



The two variables for the model are DJ and NQ. The next two steps then specify the model, and model-specific settings:

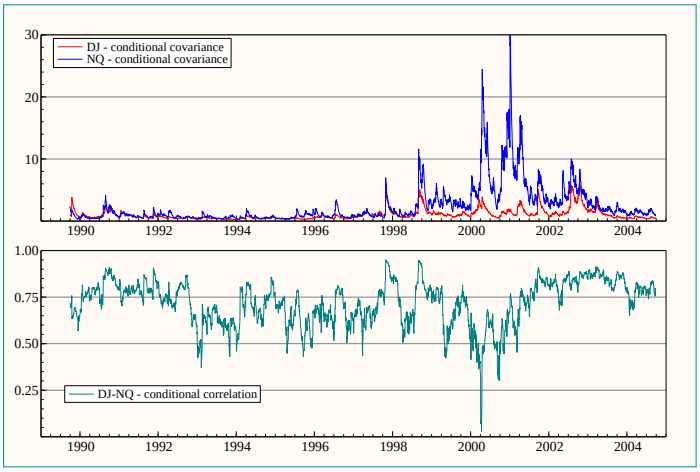




Selecting the default sample gives the univariate models in the Results wiondow, followed by the cDCC:

```
  *************
 ** SERIES  **
*************
#1: DJ
#2: NQ

The dataset is: C:\...\Documents\OxMetrics8\data\DJNQ.xls
The estimation sample is:  1989-09-28 - 2004-09-27

 ******************************
 ** MG@RCH(2) SPECIFICATIONS **
 ******************************
Conditional Variance: Corrected Dynamic Correlation Model (Aielli)
Multivariate Normal distribution.

Strong convergence using numerical derivatives
Log-likelihood = -9835.07
Please wait : Computing the Std Errors ...

 Robust Standard Errors (Sandwich formula)
                Coefficient  Std.Error  t-value  t-prob
rho_21            0.718938   0.066626    10.79  0.0000
alpha             0.040522  0.0061874    6.549  0.0000
beta              0.951995  0.0095593    99.59  0.0000
No. Observations :     3913  No. Parameters  :        13
No. Series       :        2  Log Likelihood  : -9835.071
Elapsed Time : 0.149 seconds (or 0.00248333 minutes).
```

The next graph shows the estimated conditional variances, with the conditional correlations in the bottom graph:



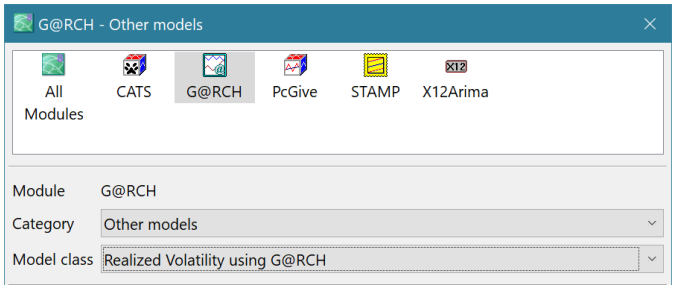
Ox code for the model just estimated is generated automatically by OxMetrics using Model/Ox Batch Code. The slightly shortened version is:

```
#include <oxstd.oxh>
#include <oxdraw.h>
#import <packages/MGarch/mgarch>

main()
{
    decl model = new MGarch();

    model.Load("DJNQ.xls");
    model.Deterministic(-1);

    model.Select("Y", {"DJ", 0, 0});
    model.Select("Y", {"NQ", 0, 0});
    model.CSTS(1,1);
    model.DISTRI(MGarch::NORMAL);
    model.ARMA_ORDERS(1,0);
    model.GARCH_ORDERS(1,1);
    model.MODEL(MGarch::cDCC);
    model.SetSelSampleByDates(
      dayofcalendar(1989,9,28), dayofcalendar(2004,9,27));
    model.Initialization(<>);
    model.PrintOutput(1);
    model.DoEstimation();

    model.PLOT_VAR(1);
    model.PLOT_CORR(1);
    model.Graphs_in_sample(-1);

    delete model;
}
```

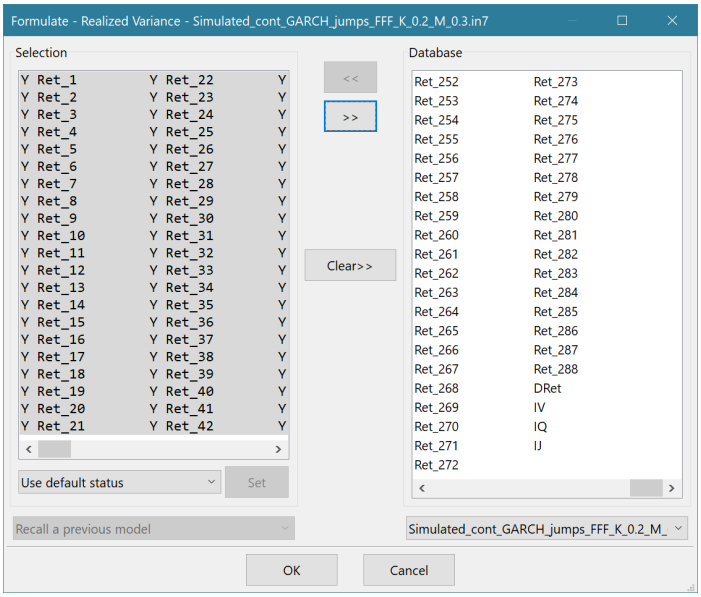
The Ox code can be used to rerun the estimation. This is particularly useful in a more production-oriented environment.

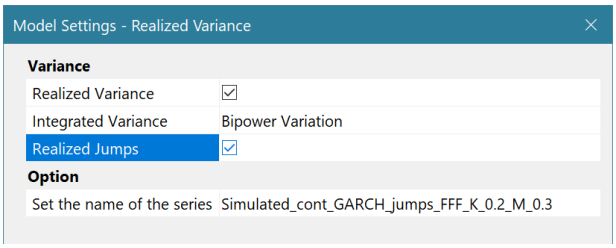
Realized volatility Example

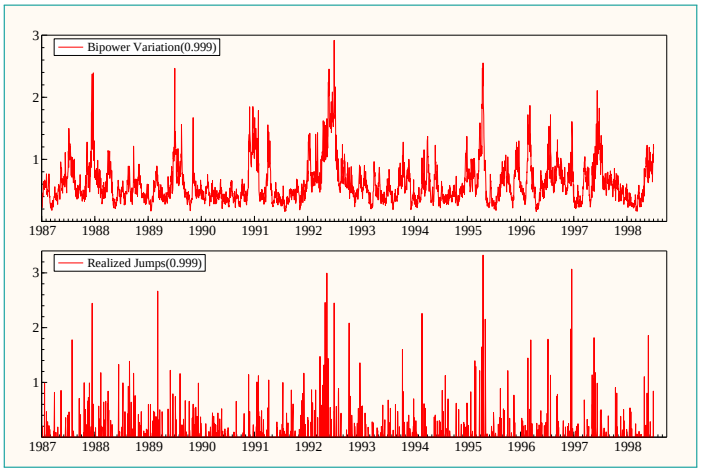
RE@LIZED provides a full set of procedures to compute non-parametric estimates of the quadratic variation, integrated volatility and jumps from intraday data. This includes univariate and multivariate versions of the realized volatility, bi-power-variation and realized outlyingness weighted variance. Daily and intradaily tests for jumps are also implemented. These facilities are accessible through the rolling menus of G@RCH, as well as from Ox.
After loading the data file from the Ox folder (`OxMetrics8/ox/packages/Realized/samples/data`), activate RE@LIZED:



All 288 variables Ret_1 to Ret_288, one for each 5 minute interval, are added to the model:








```
#include <oxstd.oxh>
#import <packages/Realized/Realized>

main()
{
    decl model = new Realized();
    model.Load("C:\\Program␣Files\\OxMetrics8\\ox\\packages"
      "\\Realized\\samples\\data\\"
      "Simulated_cont_GARCH_jumps_FFF_K_0.2_M_0.3.in7");

    for (decl i = 1; i <= 288; ++i)
        model.Select("Y", {sprint("Ret_", i)});

    model.SetModelClass(Realized::MC_RV);
    model.RV(1);
    model.IV(1);
    model.OPTIONS_JUMPS_TEST_BV(1,0,0,0.999);

    model.SetSelSampleByDates(
      dayofcalendar(1987, 1, 5), dayofcalendar(1998, 7, 3));
    model.Estimate();

    delete model;
}
```

6 STAMP

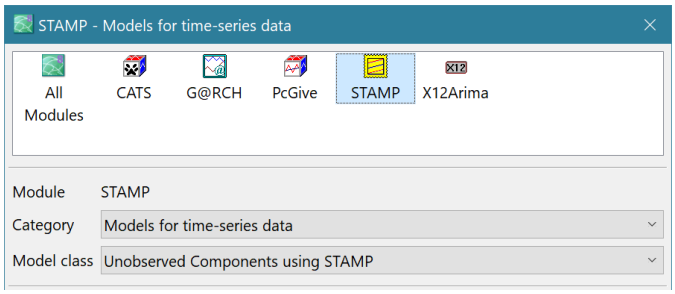
STAMP is designed to model and forecast time series, based on structural time series models. These models use advanced estimation techniques, such as **Kalman filtering**, but are formulated in STAMP using convenient dialogs — at the most basic level all that is required is some appreciation of the concepts of **trend**, **seasonal** and **irregular**. The hard work is done by the program, leaving the user free to concentrate on formulating models, then using them to make **forecasts**.
Structural time series modelling can be applied to a variety of problems in time series. Macro-economic time series like gross national production, inflation and consumption can be handled effectively, but also financial time series, like interest rates and stock market volatility, can be modelled using STAMP. Further, STAMP is used for modelling and forecasting time series in medicine, biology, engineering, marketing and in many other areas.

STAMP features

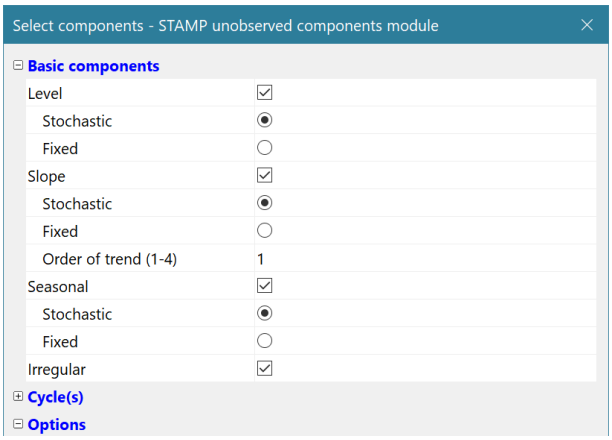
- Univariate State Space Models
- Multivariate State Space Models
- Time-varying regression coefficients
- Autoregressive processes of order 1 and 2
- High order Smooth Cycles
- Automatic outlier and break detection
- A battery of equation mis-specification tests
- Post-sample and within sample predictive testing
- Easy to use, menu driven interface
- Advanced graphics capabilities

Example

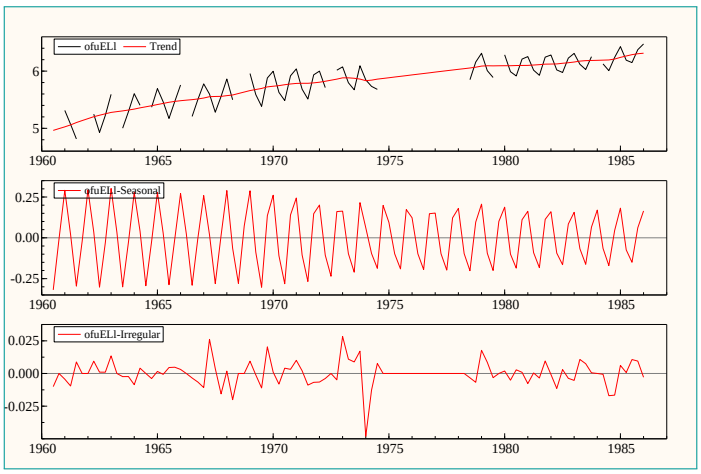
We will consider the basic structural time-series model (BSM) with unobserved components (UC) trend, seasonal and irregular and use it for the analysis of a time series with missing entries. After loading `ENERGYmiss.in7` in OxMetrics, select the *ofuELl* variable. This is quarterly UK electricity consumption between 1960 and 1986 (millions of useful therms for 'Other final users').



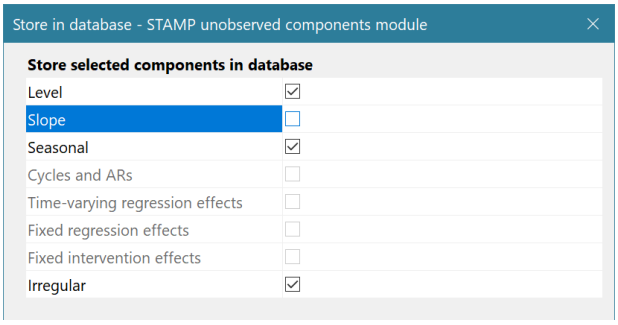
Accept the model settings as they are by default:



The model graph shows the decomposition in trend, seasonal, and irregular:



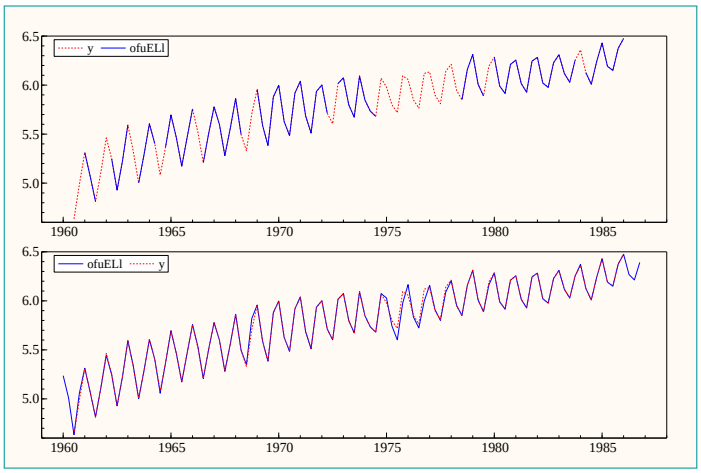
We can see how STAMP recreated the missing values. First use Store in Database from the Test menu



To store the trend, seasonal and irregular under their default names of *Level, Seasonal, Irregular* in the database. Next, use Algebra code to construct:

```
y = Level+Seasonal+Irregular;
```

The top of the following graph shows *ofuELl* as a solid blue line, and the reconstructed variable as a dotted line.



In this particular example, we actually know the full series. This is shown in the bottom part of the graph above. It shows that the reconstruction worked remarkably well.

Generating batch code

After STAMP has estimated the parameters of the BSM, the standard output is sent to the Text/Results and the Graphics/Model windows. The graphical output is as presented as the first figure. The batch option is activated from the Model/Batch option (Alt-B). The Batch code as given by

```
module("STAMP");
package("UCstamp");
usedata("ENERGYmiss.in7");
system
{
    Y = ofuELl;
```

```
    }
    setcmp("level", 1, 0.000387097, 0, 0);
    setcmp("slope", 1, 1.87978e-06, 0, 0);
    setcmp("seasonal", 4, 0.00017307, 0, 0);
    setcmp("irregular", 0, 0.000524284, 0, 0);
    setmodel();
    estimate("MAXSTAMP", 1960, 3, 1986, 1);
```

The code represents the model in STAMP after maximum likelihood estimation. The commands `module` and `package` are required to start the STAMP module in OxMetrics. The command `usedata` loads the data file "ENERGYmiss.in7" while the command `system` assigns the variable *ofuELl* as the dependent variable that we want to analyze. The unobserved components model is constructed by the commands `setcmp` which introduces the components level with slope (trend), seasonal and irregular. The third arguments are the estimated variances for the components. The model formulation for our unobserved components model is completed by the command `setmodel`. The variances in the `setcmp` commands are the ones estimated by STAMP (using the method of maximum likelihood). To run the Batch code, click on the Run button (Alt-R) and the STAMP program will re-estimate the UC model.
In case we prefer a deterministic seasonal component, we can fix the seasonal variance by setting the variance to zero. The modified line is:

```
    setcmp("seasonal", 4, 0, 0, 0);
```

When the variance is set to zero in the `setcmp` command, STAMP will treat the component as deterministic (the component is fixed over time). A part of the STAMP output in the Results window reports the maximum likelihood estimates of the variances:

```
    Variances of disturbances:
                          Value    (q-ratio)
    Level              0.000000  (   0.0000)
    Slope           1.32900e-06  (0.0002261)
    Seasonal           0.000000  (   0.0000)
    Irregular       0.00587911   (    1.000)
```

It follows that the Level component is also estimated as zero which leads to a smooth trend component in STAMP. Other useful Batch commands are `intervention` for including interventions in the model, `forecast` for generating forecasts from the model, `store` for storing residuals and estimated components from STAMP and `teststate` for printing the estimated state vector and related output. These commands are documented in the STAMP manual.

Generating Ox code

An alternative but a more flexible method of running STAMP in batch is by means of the Ox code generator facility. It is activated by pressing Alt-O when STAMP is activated. In case a model is formulated in STAMP and parameters are estimated, the option Alt-O opens the menu window `Generate Ox code` in which the user has two options. The default choice is `Most recent model` and can be accepted. STAMP then outputs the following Ox code:

```
#include <oxstd.oxh>
#import <packages/stamp/stamp_ox_uc>

main()
{
    decl model = new UCstamp();

    model.Load("C:\\Users\\...\\Documents\\OxMetrics8\\"
      "data\\ENERGYmiss.in7");
    model.Deterministic(-1);

    model.Select("Y", {"ofuELl", 0, 0});

    model.SetSelSample(1960, 3, 1986, 1);
    model.SetMethod("MAXSTAMP");
    // Specify components
    model.StartStamp();
    model.AddComponent(COM_LEVEL, 1, 0.000387097);
    model.AddComponent(COM_SLOPE, 1, 1.87978e-06);
    model.AddComponent(COM_SEASONAL, 4, 0.00017307);
    model.AddComponent(COM_IRREG, 0, 0.000524284);

    model.Estimate();

    delete model;
}
```

When STAMP is installed on the computer, the `stamp_ox_uc` library offers many Ox functions that are developed for STAMP. These functions are collected in the class `UCstamp` which is activated by the `new` command in Ox.
The command `Load` reads in the data file `ENERGYmiss.in7` and `Select` takes the variable ofuELl as the dependent variable to analyze. In the Ox code, we can first select the sample and the estimation method using the commands `SetSelSample` and `SetMethod`, respectively. The command `StartStamp` initializes the settings for formulating an UC model. The next part of the Ox code is similar to the Batch code. The inclusion of a component is established by the command `AddComponent`. In case of the seasonal component, the constant `COM_SEASONAL` indicates that the seasonal component is included in the model. The second constant indicates that we work with quarterly data (seasonal length is 4) and the value 0.00017307 is the value of the seasonal variance, as estimated by the STAMP program. The `Estimate` command (re-)estimates the variances (and, possibly, other parameters). Other commands in the `stamp_ox_uc` library are `AddIntervention` for including interventions in the model, `GetForecast` for generating forecasts from the model, `Store` for storing residuals and estimated components from STAMP and `PrintState` for printing the estimated state vector.

7 Ox

Ox is an **object-oriented matrix language** with a comprehensive mathematical and statistical function library. Matrices can be used directly in expressions, for example to multiply two matrices, or to invert a matrix. The **basic syntax elements** of Ox are similar to the C++ and Java languages (however, knowledge if these languages is not a prerequisite for using Ox). This similarity is most clear in syntax items such as loops, functions, arrays and classes. A major difference is that Ox variables have **no explicit type**, and that special **support for matrices** is available.

```
#include <oxstd.oxh>  // include Ox standard library header

main()                // function main is the starting point
{
    decl m1, m2;        // declare two variables, m1 and m2

    m1 = unit(3);   // assign to m1 a 3 x 3 identity matrix
    m1[0][0] = 2;               // set top-left element to 2
    m2 = <0,0,0;1,1,1>;  // m2 is a 2 x 3 matrix, the first
               // row consists of zeros, the second of ones

    println("two matrices", m1, m2);  // print the matrices
}
```

Object oriented

The advantages of object-oriented programming are that it potentially improves the clarity and maintainability of the code, and reduces coding effort through inheritance. Several useful classes are provided with Ox. The following example uses the **Database class**:

```
#include <oxstd.h>
#import <database>
main()
{
    decl db = new Database();
    db.Load("data.in7");
    db.Info();
    delete db;
}
```

A dynamic regression model can be estimated using the **PcFiml class**:

```
#include <oxstd.h>
#import <pcfiml>

main()
{
    decl mod = new PcFiml();

    mod.Load("data/data.in7");
    // create deterministic vars in the database
    mod.Deterministic(FALSE);
    // formulate the model, lag 0&1 for CONS,INC
    mod.Select("Y", { "CONS", 0, 1 } );
    mod.Select("X", { "INC", 0, 1 } );
    mod.Select("X", { "Constant", 0, 0 } );

    mod.SetSelSample(-1, 1, -1, 1);// max sample
    mod.Estimate();        // estimate the model

    delete mod;
}
```

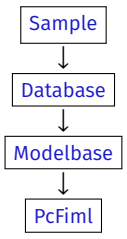
This estimates a model by ordinary least squares:

$$y_t \sim \beta_0 + \beta_1 y_{t-1} + \beta_2 x_t + \beta_3 x_{t-1} + \epsilon_t,$$

where y_t is CONS, x_t is INC (income).

- `Deterministic()` creates a constant term, trend, and seasonal dummies (a Database function).
- `Select` formulates the model: "Y" for dependent and lagged dependent variables, "X" for the other regressors. The second argument is an array with three elements: variable name, start lag and end lag.
- `SetSelSample` sets the maximum sample, but could also be used to select a subsample.
- `Estimate` estimates and prints the results. How much work would this have been starting from scratch?
- Finally, when done, we `delete` the object. This calls the destructor function, and then clears the object from memory.

The inheritance structure for the PcFiml class is:



Stores sample frequency and period.

Holds a data matrix with variables names. Handles a database selection for modelling.

Model formulation, extract data for modelling, initialize parameters, estimate, print a report.

Single or multiple equations dynamic regressions. Mis-specification testing and cointegration analysis.

Ox programming

- Ox is structured as proper programming language:
  - Somewhat harder for very small programs;
  - Much better for larger projects.
- Can be used to teach programming to economics/statistics students
- while being able to write relevant applications
- C/C++ like structure integrates well with business environment
- Facilities to easily create interactive (possibly commercial) applications for OxMetrics.

Mathematical, Statistical and Graphical library

- Many matrix and statistical functions.
- Maximization functions (Unconstrained: BFGS, Newton, QP, SQP, FSQP, NLE), numerical differentiation.
- Random number generations/quantiles/probabilities of many statistical distributions.
- Time-series functions.
- Many graph types.
- Load and save data files.
- Date and time functions.

Further features

- It is possible to mix high and low level code:
  - add C or Fortran procedure libraries to Ox, e.g. time critical sections, or use available libraries (e.g. SsfPack);
  - use Ox procedures from C.
- Call Ox from other languages.
- Write interface for Ox program in another languages.
- Web hosted environment (under development).

OxPack

The following structure

Database class ⟶ Modelbase class ⟶ your class

makes it easy to create an interactive version that can be used via OxPack. Examples are: Arfima, PcNaive, DPD, G@RCH, etc.
The benefits are:

- One code base for all versions: simulation, estimation, incorporation, as well as GUI version:
- Dialogs written in Ox using code that is easy to write and maintain:
  - Simple code,
  - Convenient run/development cycle,
  - No need to be professional programmer.

Documentation

- *Introduction to Ox*
- *Ox: An Object-oriented Matrix Programming Language*
- *Developer's manual for Ox*

8 SsfPack

SsfPack is a suite of C routines for carrying out computations involving the statistical analysis of **univariate and multivariate models in state space form** with easy-to-use functions for **Ox**. SsfPack allows for a full range of different state space forms: from a simple time-invariant model to a complicated multivariate time-varying model.
Functions are provided to put standard models such as SARIMA, unobserved components, time-varying regressions and cubic spline models into state space form. Basic functions are available for Kalman filtering, moment smoothing and simulation smoothing. Ready-to-use functions are provided for standard tasks such as likelihood evaluation, forecasting and signal extraction.
SsfPack can be easily used for implementing, fitting and analysing Gaussian models relevant to many areas of econometrics and statistics. Furthermore it provides all relevant tools for the treatment of non-Gaussian and nonlinear state space models. In particular, tools are available to implement **simulation based estimation methods** such as importance sampling and Markov chain Monte Carlo (MCMC) methods.

Examples of SsfPack functions

Models in state space form

GetSsfSarima()	puts Seasonal ARIMA model in state space
GetSsfStsm()	puts UC model with higher order trends and cycles in state space

General state space algorithms

These use the new more robust filtering algorithms with exact diffuse initialisations.

KalmanFilInit()	returns initial output the exact diffuse Kalman filter
KalmanFilEx()	returns output of the Kalman filter
KalmanSmoEx()	returns output of the basic smoothing algorithm
KalmanFilMeanEx()	returns Kalman filter output to implement diffuse initialisation based on augmentation method
KalmanSmoMeanEx()	returns Kalman smoother output to implement diffuse initialisation on augmentation method
KalmanFilSmoMeanEx()	alternative Kalman smoother output to implement diffuse initialisation on augmentation method

Likelihood functions

These use the new more robust filtering algorithms with exact diffuse initialisations.

SsfLikEx()	returns log-likelihood function
SsfLikConcEx()	returns profile log-likelihood function
SsfLikScoEx()	returns score vector

Ready-to-use functions

These use the new more robust filtering algorithms with exact diffuse initialisations.

SsfMomentEstEx()	returns output from prediction, forecasting and smoothing
SsfCondDensEx()	returns mean or a draw from the conditional density
SsfForecast()	forecasting function with simplified syntax
SsfWeightsEx()	returns observation weights of state and signal estimates
SsfBootstrap()	returns bootstrap simulation draws