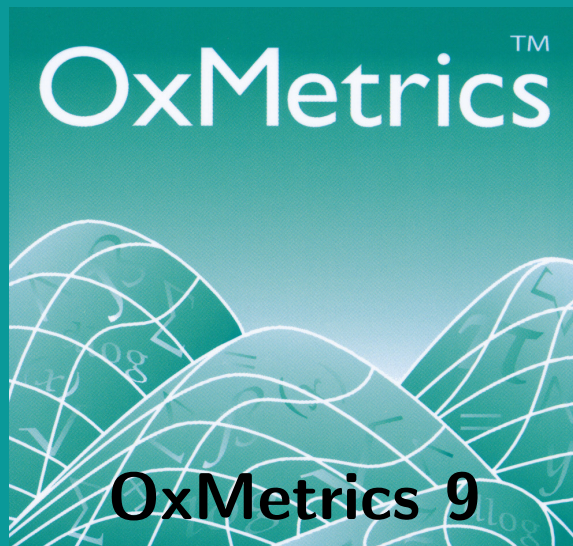


Jurgen A. Doornik

**An Introduction to
OxMetrics™ 9**



Published by Timberlake Consultants Ltd
www.timberlake.co.uk
www.timberlake-consultancy.com
www.oxmetrics.net, www.doornik.com

An Introduction to OxMetrics™ 9
A Software System for Data Analysis and Forecasting

Copyright ©2021 Jurgen A Doornik

First published by Timberlake Consultants in 2005

Revised in 2007, 2009, 2013, 2018, 2022

All rights reserved. No part of this work, which is copyrighted, may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopy, record taping, or information storage and retrieval systems – without the written consent of the Publisher, except in accordance with the provisions of the Copyright Designs and Patents Act 1988.

Whilst the Publisher has taken all reasonable care in the preparation of this book, the Publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions from the book, or from the software, or the consequences thereof.

British Library Cataloguing-in-Publication Data

A catalogue record of this book is available from the British Library

Library of Congress Cataloguing-in-Publication Data

A catalogue record of this book is available from the Library of Congress

Jurgen A Doornik

p. cm. – (An Introduction to OxMetrics™ 9 A Software System for Data Analysis and Forecasting)

ISBN

Published by

Timberlake Consultants Ltd
The Loft, 2C Blake Mews Kew Gardens
Richmond, TW9 3GA, UK
<http://www.timberlake.co.uk>

Timberlake Analytics, Inc

www.timberlake-analytics.com

Trademark notice

All Companies and products referred to in this book are either trademarks or registered trademarks of their associated Companies.

Contents

Front matter	iii
Contents	v
List of Figures	xiii
List of Tables	xv
Preface	xvii
1 Introduction and Overview	3
1.1 Supported platforms	3
1.1.1 Folder structure	4
1.2 What is new?	4
1.3 Availability	5
1.4 Citation	5
1.5 Help	5
1.6 Modular application structure	6
1.7 Ox language	6
1.8 Registration	6
1.9 Upgrades	7
1.10 Algebra	7
1.11 Batch	7
1.12 Data storage	8
1.13 Filenames and extensions	8
1.14 Sample periods	8
1.15 Status bar	9
1.16 Tool bars	9
1.17 Documentation conventions	10
2 Getting Started	11
2.1 Registering OxMetrics	12
2.2 Loading and viewing the tutorial data set	13

2.3	OxMetrics graphics	15
2.3.1	A first graph	15
2.3.2	Multiple graphs	18
2.3.3	Graph saving and printing	20
2.3.4	Using the clipboard for graph pasting	20
2.4	Calculator	21
2.5	Algebra	23
2.6	The workspace	25
2.7	Preferences	25
3	OxMetrics Applications	27
3.1	OxMetrics Modules	27
3.2	Financial data	28
3.3	Weekly and daily data	29
3.4	PcGive	31
3.4.1	Formulate a model	32
4	Tutorial on Graphics	39
4.1	Descriptive graphics	39
4.2	Actual series with optional transformations	42
4.3	Multiple series with optional transformations	44
4.4	Scatter plots	45
4.5	Distribution	47
4.5.1	Density estimation: Histogram and density	48
4.5.2	Distribution	48
4.5.3	Frequencies	49
4.5.4	Box plot	50
4.6	Time-series: ACF etc.	50
4.6.1	Autocorrelation function or correlogram	50
4.6.2	Partial autocorrelation function (PACF)	51
4.6.3	Cross-correlation function	51
4.6.4	Spectrum and periodogram	52
4.7	QQ plots	53
4.8	Two series by a third	54
4.9	3-dimensional plots	57
4.9.1	Surface from scatter	57
4.9.2	Surface from table	59
4.10	Conclusions	60
5	Tutorial on Graph Editing	61
5.1	Multiple graphs	61
5.2	Graphics paper: areas and coordinates	61
5.3	Graphics view	63
5.4	New Data Plot Window	63

5.5	Copy and paste	63
5.6	About line colour and style	64
5.7	Editing graphs: Graphics properties	64
5.8	Graphics setup	68
5.9	Adding to and removing from a graph	69
5.10	Drawing	70
5.11	Adding text and variables	71
5.12	Legends	72
5.13	Scaling variables	72
6	Tutorial on Data Input and Output	75
6.1	Entering data directly into OxMetrics	76
6.1.1	Directly into the database	76
6.1.2	Using the clipboard; using OxMetrics as editor	78
6.2	Saving data	80
6.3	Loading data into OxMetrics	81
6.3.1	Loading OxMetrics files	81
6.3.2	Loading spreadsheet files	81
6.3.3	Loading a text (human-readable) file	82
6.4	Adding variables using the clipboard	82
6.5	Changing the sample period	83
6.5.1	Extending the sample period	83
6.5.2	Reset starting date	83
6.6	Appending data	83
6.7	Working with daily and weekly data	84
6.7.1	Using Change Sample to create a daily database	85
6.7.2	Using Algebra to create a daily database	86
6.8	Working with choice variables	87
7	Tutorial on Data Transformation	90
7.1	Calculator	90
7.2	Advanced algebra	92
7.2.1	Introduction	92
7.2.2	Database for advanced algebra	92
7.2.3	Statistical distributions	92
7.2.4	Random number generators	95
7.2.5	Generating data	97
7.2.6	Smoothing data	99
8	OxMetrics Statistics	103
8.1	Actual series and scatter plots	103
8.2	Mean, standard deviation and variance	103
8.3	Autocorrelation function (ACF) or covariogram	104
8.4	Partial autocorrelation function (PACF)	104

8.5	Correlogram	104
8.6	Cross-correlation function (CCF)	105
8.7	Periodogram	105
8.8	Spectral density	106
8.9	Histogram, estimated density and distribution	106
8.10	Regression lines and smooths	107
	8.10.1 Kernel smooth	108
	8.10.2 Spline smooth	109
8.11	QQ plot	110
8.12	Box plot	111
8.13	Exponentially-weighted moving average (EWMA)	111
8.14	Exponentially-weighted moving correlation	111
9	OxMetrics file formats	113
9.1	OxMetrics data files (.oxdata)	113
9.2	OxMetrics data files (.in7/.bn7)	113
	9.2.1 The .in7 file format	113
	9.2.2 The .bn7 file format	115
	9.2.3 Limitations	116
	9.2.4 The information and ASCII data files (.in7/.dat)	116
9.3	Spreadsheet files (.xlsx,.csv)	116
	9.3.1 .xlsx files	117
	9.3.2 .csv files	119
	9.3.3 Zipped .csv files	120
	9.3.4 Interpreted and uninterpreted .xlsx and .csv files	120
9.4	Data by observation (.dat)	120
9.5	Data with load info (.dat)	121
9.6	Stata data file (.dta)	121
9.7	Results file (.out)	122
9.8	Batch file (.f1)	122
9.9	Algebra file (.alg)	122
9.10	Ox file (.ox)	122
9.11	Matrix file (.mat)	122
9.12	OxMetrics graphics file (.gwg)	122
9.13	PDF file (.pdf)	124
9.14	SVG file (.svg)	124
9.15	PostScript file (.eps)	124
9.16	PostScript file (.ps)	127
9.17	Enhanced meta file (.emf)	127

10	Algebra Language	128
10.1	Introduction	128
10.2	Executing Algebra code	128
10.2.1	Calculator (Alt+c)	128
10.2.2	Algebra Editor (Alt+a)	129
10.2.3	Algebra from Results windows (Ctrl+a)	129
10.2.4	Algebra from a Batch file	129
10.3	Syntax of Algebra language	129
10.3.1	Variables and variable names	129
10.3.2	Comment	129
10.3.3	Constants	130
10.3.4	Algebra operators	130
10.3.4.1	Arithmetic operators	130
10.3.4.2	Relational and logical operators	130
10.3.4.3	Algebra operator precedence	130
10.3.5	Assignment statements	130
10.3.6	Conditional assignment statements	131
10.3.7	Indexing	132
10.3.8	Keywords	132
10.4	Algebra implementation	133
10.5	Algebra Functions	134
10.5.1	Differencing and lag functions	134
10.5.2	Implicit functions	134
10.5.3	ACF and periodogram functions	135
10.5.4	Sorting functions	135
10.5.5	Smoothing functions	136
10.5.5.1	Hodrick–Prescott filter	136
10.5.5.2	Kernel and spline smoothing	137
10.5.5.3	Exponentially-weighted moving average and correlation	137
10.5.5.4	Date and time functions	137
10.6	Algebra function summary	138
11	Batch Language	143
11.1	Introduction	143
11.2	Executing Batch commands	143
11.2.1	Batch Editor (Alt+b)	143
11.2.2	Batch from Results windows (Ctrl+b)	143
11.2.3	Batch from the File/Open command	144
11.2.4	Batch from the Windows Explorer	144
11.2.5	Batch from a Batch file	144
11.3	Batch files and default folders	144
11.4	General Batch command summary	144
11.4.1	Comment	144

11.4.2	Command types	144
11.4.3	Default arguments	145
11.5	Batch commands	146
11.5.1	algebra { ... }	146
11.5.2	appenddata("filename", "group=");	146
11.5.3	appresults("filename");	146
11.5.4	break;	146
11.5.5	chdir("path");	146
11.5.6	closedata("databasename");	147
11.5.7	closedrawwindow();	147
11.5.8	command("command_line");	147
11.5.9	database("name", year1, period1, year2, period2, freq);	147
11.5.10	draw(area, "y", "mode=");	147
11.5.11	drawf(area, "y", "function", d1=0, d2=0);	147
11.5.12	drawptext(area, "text", "x", "y", size=300, rot=0);	148
11.5.13	drawtext(area, "text", "x", "y", size=300);	148
11.5.14	drawtitle(area, "text", "pos", size=300);	148
11.5.15	drawx(area, "y", "x", "mode=");	148
11.5.16	drawz(area, "y", "x", "mode=");	148
11.5.17	exit;	149
11.5.18	loadalgebra("filename");	149
11.5.19	loadbatch("filename");	149
11.5.20	loadcommand("filename");	149
11.5.21	loaddata("filename");	149
11.5.22	loadgraph("filename");	150
11.5.23	module("name");	150
11.5.24	package("packagename", "modeltype=");	150
11.5.25	print("text");	150
11.5.26	println("text");	150
11.5.27	printdate;	150
11.5.28	savadata("filename");	150
11.5.29	savedrawwindow("filename", "window=");	150
11.5.30	saveresults("filename");	150
11.5.31	setdraw("option", i1=0, i2=0, i3=0, i4=0, i5=0);	151
11.5.32	setdrawwindow("name");	152
11.5.33	show;	152
11.5.34	usedata("databasename", i1=0);	152
11.6	Examples	153
12	OxMetrics Graphics	155
12.1	Graphics paper	155
12.2	Creating graphs	156
12.2.1	Actual series with optional transformations	156
12.2.2	Multiple series with optional transformations	157

12.2.3	Scatter plots	157
12.2.4	Distribution	157
12.2.5	Time-series: ACF etc.	158
12.2.6	QQ plots	158
12.2.7	Two series by a third	159
12.2.8	3-dimensional plots	159
12.3	Printing graphs	160
12.4	Graphics formats	160
12.5	Saving and loading graphs	161
12.6	Saving Ox code of a graph	161
12.7	Editing graphs	162
12.7.1	Graph layout	162
12.7.2	Area layout	163
12.7.3	Variable against time, scatter, or 3D	163
	12.7.3.1 Error bars	164
	12.7.3.2 Regression, Scale	164
12.7.4	Axes	165
	12.7.4.1 Settings for non-default labelling	165
	12.7.4.2 Location and transformation	165
	12.7.4.3 Style	166
	12.7.4.4 Label style	166
12.7.5	Legend style	166
12.7.6	Histogram	167
12.7.7	Copy properties to other areas	167
12.7.8	Text	167
12.7.9	Lines and symbols	167
12.7.10	Adding, moving and deleting objects	168
12.7.11	Pointing	168
12.7.12	Graphics setup	168
12.8	Copy and paste	168
12.9	Graphs and sample selection	168
12.10	Text formatting	169
13	OxMetrics Data Management	172
13.1	Creating data	172
13.2	Database font	172
13.3	Database description	172
13.4	Printing data	172
13.5	Data formats	173
13.6	Summary statistics	173
13.7	Saving data	173
13.8	Navigation and editing	173
13.9	Renaming variables	174
13.10	Deleting variables	174

13.11	Reordering variables	174
13.12	Adding variables	174
13.13	Extending or reducing the sample period	174
13.13.1	Changing the sample period	175
13.14	Copy and paste	175
13.15	Appending data	175
13.16	Daily, weekly and timed data	175
References		177
Author Index		179
Subject Index		181

Figures

2.1	Time plot of CONS and INC	17
2.2	Multiple graphs	19
3.1	Dow Jones returns, ACF of returns, ACF of squared returns . . .	30
3.2	GARCH(1,1) model for Dow Jones returns	36
4.1	Descriptive graphics of CONS and DCONS	42
4.2	Time-series plot of CONS and INC with ‘recession’ shading	44
4.3	Time-series plot of INFLAT and OUTPUT: maximizing visual correla- tion	44
4.4	Sophisticated scatter plotting graphs	46
4.5	Further descriptive graphics of CONS and DCONS	49
4.6	Box plot of INFLAT	50
4.7	Comparison of correlogram and ACF for CONS and DCONS	51
4.8	Cross-correlogram of CONS and DCONS	51
4.9	Frequency domain graphics of CONS and DCONS	52
4.10	Densities and distributions of normal and t	53
4.11	QQ plots	54
4.12	Cross plotting consumption against income by seasons (DHSY) .	55
4.13	Bubble chart and error bars	56
4.14	3-dimensional graphs: bivariate normal and saddles	59
4.15	3-dimensional graphs from tabular data	60
5.1	Descriptive graphics of CONS, INC, DCONS and DINC	62
5.2	Edited and amended descriptive graph	68
5.3	Drawing on an INFLAT/OUTPUT scatter plot	71
5.4	CONS and OUTPUT with matching mean and range	73
6.1	Check of Cons and Inflat	78
7.1	Some important densities	94
7.2	Cumulative distribution functions from 600 points	95
7.3	Cumulative distribution functions	96

7.4	Check on CDFs	96
7.5	Time series graphs of 6 processes	97
7.6	Correlograms and spectral densities of six processes	98
7.7	AR(1) with various parameter values	99
7.8	Splines with various bandwidth settings	100
9.1	A simple graph	124

Tables

6.1	OxMetrics data input/output options	75
9.1	<code>data.in7</code> : information file of artificial data set	114
10.1	Algebra operator precedence	131
10.2	Reserved Algebra keywords	133
10.3	Algebra functions	140
11.1	Batch command summary	145
12.1	Greek symbols	169
12.2	Arrows	169
12.3	Mathematics accents	169
12.4	Foreign and accented characters	170
12.5	Mathematical symbols	170
12.6	Log-like symbols	170
12.7	Miscellaneous symbols	170
12.8	Fonts and sizes	171

Preface

The principle of OxMetrics is to have a complete separation of the front-end (for data manipulation and visualization) and the econometric and statistical applications (Ox, PcGive, G@RCH etc.). While it appears as an integrated system to the user, it maintains the benefits of being able to develop the many components separately.

OxMetrics uses wxWidgets to implement its interface. This makes it possible for all the interactive software to be available on recent versions of Windows, macOS, as well as Linux. OxMetrics also uses Unicode, thus supporting most languages.

Ox, an object-oriented matrix programming language, is one of the modules which can interface with the OxMetrics front-end. Indeed, all modules now do their computations largely in Ox, rather than in lower level C or C++. In many cases, OxMetrics also offers to write the Ox code of the estimated model, which provides a more flexible alternative to Batch code.

Although OxMetrics 9 represents a new generation, it builds on earlier versions of GiveWin and PcGive, and we wish to reiterate our thanks to all those who have helped in its development.

We wish you enjoyable and productive use of
OxMetrics

Getting Started with OxMetrics

Chapter 1

Introduction and Overview

The **OxMetricstm** system presents an interactive menu-driven graphics-oriented system for econometric, statistic, and financial analysis. Here we describe the core OxMetrics program, which acts as the ‘front-end’ to a series of integrated software applications. These applications obtain their data from OxMetrics and return output and graphics to it. OxMetrics is the component that allows you to load, edit, and save data; transform that data using the calculator or algebra; create a wide variety of graphs, which can be edited, amended and saved in various formats; provide the data for other applications to analyze; receive their text output, results, and graphics; and lets you edit, amend, and save any or all of these as desired. As such, the OxMetrics front-end can be seen as the desktop for econometric and statistical modelling.

OxMetrics databases can have a fixed frequency, or are ‘dated’, which allows for daily or timed observations. Aggregation facilities are provided, e.g. to convert daily data into monthly. OxMetrics is multilingual, in that names of database variables and text in reports and graphics can be from a wide range of languages, including Chinese and Japanese.

This introductory chapter notes the OxMetrics languages (Algebra and Batch), describes the Help system, notes the data storage format and how results (both text and graphics) are saved, including a brief explanation of the various filenames and extensions used. Finally it explains the conventions of the documentation. The Ox language is treated in separate books.

1.1 Supported platforms

The OxMetrics suite of programs runs on the following platforms:

- Windows
Installation is to `\Program Files\OxMetrics9\` by default, but this can be changed in the installation program.
- macOS (OS X)
Installation is to `/Applications/OxMetrics9/`.

- Linux

Installation is to `/usr/share/OxMetrics9/`.

A separate installation document describes the supported platforms in more detail, and shows how to install the software.

OxMetrics 9 operates independently from OxMetrics 8 and previous versions.

1.1.1 Folder structure

The OxMetrics folder structure is as follows:

```
OxMetrics9/
  oxmetrics.exe      OxMetrics executable
  oxdedit.exe        OxEdit executable
  algebra/           Algebra files
  apps/              OxMetrics application runtime files
  batch/             Batch files
  data/              Data files
  doc/               Help system
  ox/                Root of Ox installation: Ox executable files
    doc/             help files for Ox
    include/         Ox header files for Ox code
    lib/             Ox additional code library
    packages/        Ox packages
    samples/         Ox sample code
    ...
  OxEdit_resources/  Additional resources for OxEdit
  OxMetrics_resources/ Additional resources for OxMetrics
```

1.2 What is new?

This documentation refers to OxMetrics 9. OxMetrics 8 users will find much that is familiar. There are a number of small fixes in this version, which are documented in the online help.

The most important new features are:

- Support for dark mode. This is detected automatically under macOS, but can be set in Model/Preferences/Options. Under Windows and Linux not all dialogs will be dark. Graphics windows are never dark.
- New default data format: *.oxdata (this is the .in7/.bn7 files together in a zip file).
- Improved csv reading and writing, and support for zipped csv files.
- Dropped support for obsolete spreadsheet files (.xlsx, .wks), and .dht data files.
- Improved support for high resolution screens (HiDPI) and improved dialogs.
- Native support for Apple silicon (M1).
- Dropped support for 32-bit versions.
- New full precision double point to string conversion avoids printing (e.g.) 0.46000000000000002.

- Graphs can be saved in the SVG vector format which can be incorporated in an html document. SVG files can also be inserted into recent versions of Microsoft Word and Excel.
- The Ox code to regenerate the graph can be created by right-clicking on a graph in the document listing and selecting Save Ox code. This can be useful if, after manual adjustments, the graph is to be used as a template for other graphs.
- PcGive and others: made Test menu into popup menu with shortcuts
- PcGive, STAMP, G@RCH dialogs now centred on OxMetrics Model Class dialog
- New Algebra functions: `seasonal(LAG)`, `cseasonal(LAG)`, `DI(YEAR, PERIOD)`, `II(YEAR, PERIOD)`, `SI(YEAR, PERIOD)`, `TI(YEAR, PERIOD)`, `lag0`
- New Batch functions: `drawtext`, `drawptext`, `drawtitle`.
- Special variables (`Seasonal`, `CSeasonal`, `II#1980(1)`, `SI#1980(1)`, `TI#1980(1)`, etc.) can be used on the right-hand side in Algebra expressions.
- More concise way to write system Batch command (old way still works), e.g.:

```
system
{
    Y = "log(LY)" 0:3;
    Z = Constant, "log(X)" 1:2 4:5 3, LC 0:5;
}
```

1.3 Availability

For availability contact Timberlake Consultants: www.timberlake-consultancy.com or www.timberlake.co.uk.

Consult www.doornik.com or www.oxmetrics.net for pointers to additional information relevant to the current and future versions of OxMetrics. A demonstration version is also available from these web sites.

The Help/Support menu lists some contact addresses for OxMetrics, as well as web sources of information.

1.4 Citation

To facilitate replication and validation of empirical findings, the application used to generate the results should be cited in all reports and publications involving its application.

1.5 Help

In addition to this document, OxMetrics incorporates a cross-referenced help system based on HTML files. These will open in your default browser. OxMetrics help is available from the Help menu or the Help pane in the workspace. The Help pane lists

the contents of all the applications that have been installed within OxMetrics. Double clicking on an entry will start your default browser with the Help contents.

Context-dependent help, where available, can be obtained either by pressing the F1 key, or from the Help menu. For example, when writing an Ox program, put the caret on 'println', and press F1. OxMetrics will search the html index files for a reference to that word. If there is only one, it will jump there immediately. Otherwise it will offer a list of choices in a dialog.

1.6 Modular application structure

Several applications interact with OxMetrics. In that case, the OxMetrics front-end is the 'server', while the applications (G@RCH, PcGive, STAMP, etc.) are the 'clients'. The communication is implemented via a local socket.

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 which do this. This requires the use of the Modelbase class, which provides the necessary functionality. Examples of this are Arfima, PcGive, G@RCH, etc.

1.7 Ox language

Ox is a powerful object-oriented matrix programming language with an extensive statistical library. Ox allows you to write your own programs using high level matrix operations, and provides easy facilities to read the same data files as OxMetrics can load. When the preprogrammed options in other applications do not provide a required estimator or test, and you have some basic programming skills, Ox could be used instead. Ox also has preprogrammed classes (a class is a term in object-oriented programming) to facilitate writing Monte Carlo experiments. Ox tends to be faster than other popular matrix languages. Please consult the separate Ox documentation for further information. The **OxRun** dialog application allows running Ox programs with OxMetrics as the destination of text and graphical output. An interesting example which shows densities and QQ plots while a Monte Carlo experiment is in progress is provided in `ox\samples\simulation\simnor.ox`. The **OxPack** application provides an interactive front-end to several Ox packages.

1.8 Registration

A licensing code is required for the OxMetrics software to work properly. This code will have been supplied with your copy, and, under normal circumstances, the licence that is entered during installation is automatically used by the software.

Without code, OxMetrics will enter evaluation mode, allowing use for a short period.

Additional licences can be entered under the Help/Registration menu option (a dialog will automatically appear when OxMetrics is not registered). In the subsequent dialog you can enter your code as well as your name and affiliation. Help/Registration can be also used to check the available licences for using client applications.

1.9 Upgrades

Minor upgrades can be downloaded from www.doornik.com/products.html.

1.10 Algebra

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**.

Algebra is a simple vector language, operating on the variables in the database. The operation is applied to each observation in turn, although it is possible to limit access to a subsample. The syntax is described in [Chapter 10](#).

1.11 Batch

OxMetrics is a menu-driven program for ease of use, but some operations can be implemented by entering commands. These commands are parts of simple ‘computer language’ which allow some control of OxMetrics through Batch operations. A Batch program may contain sections of Algebra code.

Batch files allow you to load data, append results, implement algebra and save current PcGive (or STAMP etc.) models. Later, these can be run from the Model/Batch menu or by clicking on the OxMetrics icon on the toolbar. Thus, when a complicated model has been created interactively, it can be saved as a batch file for further editing, or easy recall in a later session. This is also the most convenient way to create a batch file. Like Algebra, batch code can be run directly by highlighting the block of text, and then pressing **Ctrl+B**.

Once saved to disk, a batch file can also be run directly using File/Open, or even by double clicking on the batch file in the Windows Explorer. Batch files have the `.fl` extension, which originally stood for Fiml Language. The syntax is described in [Chapter 11](#).

1.12 Data storage

The primary mode of data storage is a file with extension **.oxdata**. This is a zip file that contains two files. If you rename `data.oxdata` to `data.oxdata.zip`, you can see that it contains `data.in7` and `data.bn7`. The latter is a binary file containing the actual data, whereas the former holds the information on the contents of the binary file.

The `.bn7/.in7` pairing is still supported, but the `oxdata` format is more convenient, because it always keeps both together (while taking less space).

Please be careful not to overwrite precious data sets. It is always wise to make regular backups of important files: hard disks may break down, accidental deletion occur or viruses could strike.

OxMetrics can read and write human-readable files and **Excel** spreadsheet files. OxMetrics can also read comma-separated (.csv) and Stata files. See [Chapter 9](#) for detailed lists on the file formats supported by OxMetrics.

1.13 Filenames and extensions

All file names have automatic default extensions which need not be input: a detailed discussion is provided in [Chapter 9](#). Say the basic data set is called `m1ukq`, then the data file might be `m1ukq.oxdata`, the OxMetrics Results window storage file could be `m1ukq.out`, the algebra storage file `m1ukq.alg`, and batch files `m1ukq.f1`. Graphs can be saved in many formats, including PDF (.pdf), Windows metafiles (.wmf), enhanced metafiles (.emf), Scalable Vector Graphics (.svg). The OxMetrics graphics file (.gwg, for GiveWin graphics) is always saved alongside the other format, and this is the file that is reloaded by OxMetrics for further editing.

Note that by default Windows does not show the file extension in the Explorer window — only the icon shows what the file type is. If you wish, you can switch the display of file extensions on in the Folder Options entry in the Explorer.

1.14 Sample periods

For data samples, reference is by the absolute date in the form **Year(Period)** to **Year(Period)**, for example: 1965(1) to 1985(3). Whenever a sample choice has to be made, OxMetrics will show the maximum available and will not allow choices outside that range.

When the data is *dated*, for example consisting of daily data, the choice is made by entering dates in ISO format of yyyy-mm-dd. The year is always four digits, the month one or two digits (1=January) and the date also one or two digits. For example: 1965-01-31 to 1985-9-5.

Time is written as hh:mm:ss.uuu where hour is two digits on the 24-hour clock (so 22:00 is 10 PM), mm is minutes, the seconds and hundreds are optional. When

combined with a date, the ISO standard uses a T to glue the parts together: 1980-1-1T12:05:05.

There is no facility to handle time zones.

1.15 Status bar

The status bar is displayed at the bottom of the OxMetrics window and consists of four areas:

1. The left area shows current status messages.
2. If a tool is running, this is indicated on a yellow background in the second panel.
3. The third panel shows the location of the current document. The content is different for text, data and graphics.
 - For text it is the position in the document of the caret.
For example: L 113 C 14 indicates that the caret is at Line 113, Column 14. In binary/hexadecimal viewing mode this shows the position of the caret in decimal (the first character of the document is at byte 0). This field will also indicate if the document is read-only, or when the editor is in overwrite mode.
 - For data it is the position in the database of the caret. If there is an active selection, it shows that instead.
 - For graphics it is the *X* coordinate of the mouse cursor.
4. The right area of the status bar also depends on the document type:
 - For text it indicates the document encoding properties:
 - *End-of-line marker*
One of: Win, Lnx, Mac for Windows (\r\n), macOS/Linux (\n).
 - *Multilingual file format*
If the file format is different from the default (ASCII) encoding, this is indicated by one of UTF8, UTF16, or UTF32. The can be followed by BE or LE to indicated big-endian or little-endian encoding.
These properties can be changed using Edit/Text properties.
 - For data it is the value of the observation which has the caret. If there is an active selection, it shows the selected sample.
 - For graphics it is the *Y* coordinate of the mouse cursor.

1.16 Tool bars

Three tool bars are displayed across the top of OxMetrics, below the menu bar. The tool bars can be customized from a right click on the bar. Customization allows adding or removing icons, and choosing small medium or large icon size (medium is the default).

The Find/Replace tool bar, by default on the second row, replicates most commands of the Find/Replace dialog. The two are actively linked, and the Find/Replace bar is often a convenient way to quickly search for text or execute search/replace actions. The default layout consists of:

Find text Specify the text to search for. The drop down list box holds previously used search texts. You can use a context menu (right click) to insert text from the clipboard.

Find down Finds the next occurrence of the search text downwards from the current location of the caret.

Find up Finds the next occurrence of the search text upwards from the current location of the caret.

Case sensitive Toggle case sensitive searching on or off (if the button is down, the search is case sensitive).

Whole word Toggle word matching on or off (if the button is down, the search will only find whole words).

Goto next instance Finds the next instance of the currently selected text downwards from the current location. The selected text will become the default for subsequent searches.

Goto previous instance Finds the previous instance of the currently selected text upwards from the current location. The selected text will become the default for subsequent searches.

Show Find dialog Open the Find dialog.

Show Replace dialog Open the Find/Replace dialog.

Find in Files Open a dialog that allows searching for text strings in disk files.

1.17 Documentation conventions

Code, instructions that need to be type, and file names are displayed in Typewriter font. Commands on menus, toolbar buttons, and dialog items (buttons, checkboxes etc.) are shown in the text in a Sans Serif font.

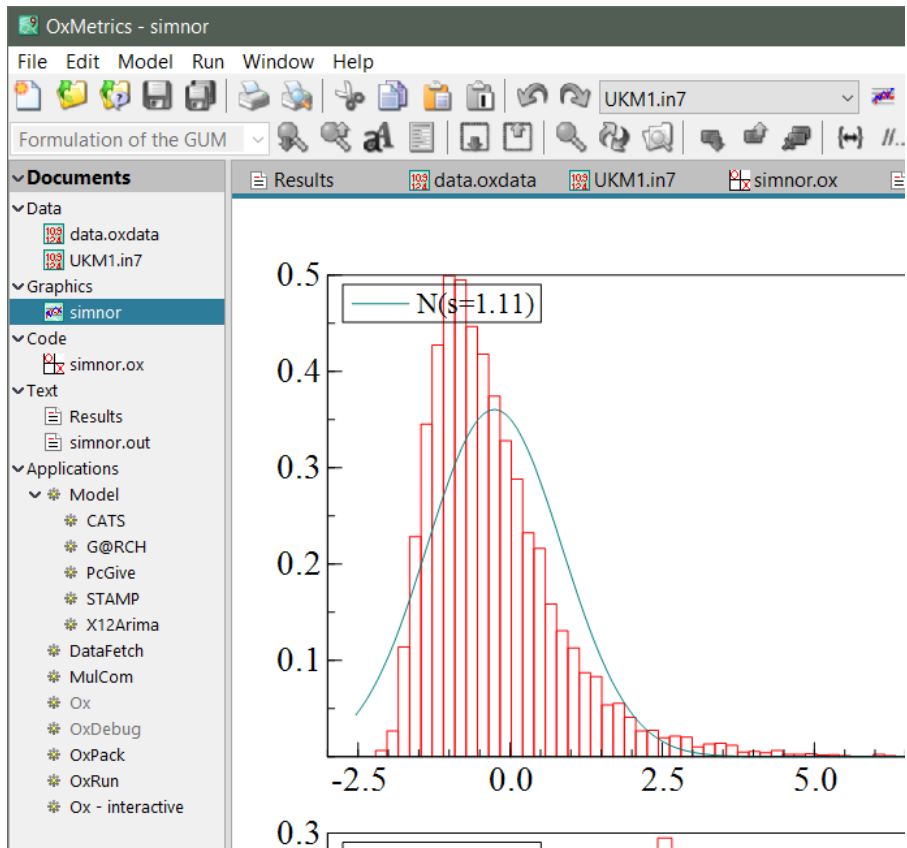
Equations are numbered as (chapter.number); for example, (8.1) refers to equation 8.1, which is the first equation in Chapter 8.

Multiple graphs are numbered from left to right and top to bottom, so b is the top-right graph of four, and c the bottom left.

Chapter 2

Getting Started

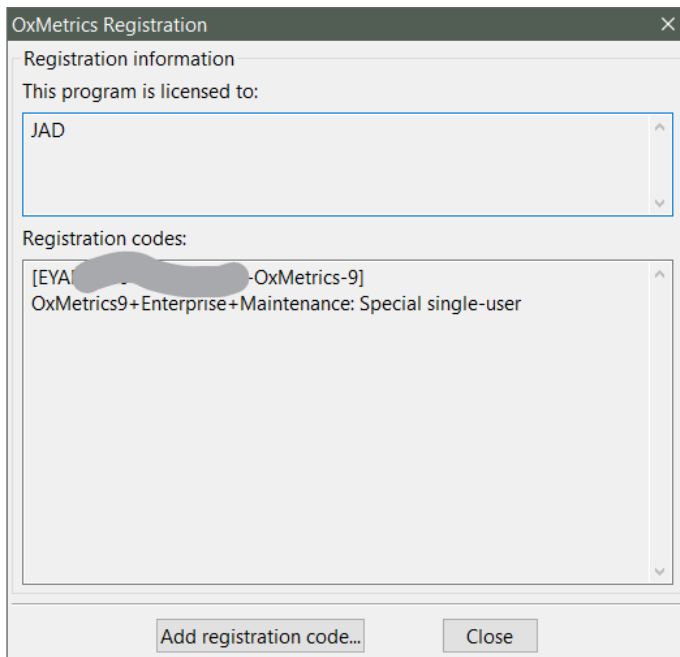
This chapter discusses some of the basic skills required to get started with OxMetrics. The functionality of OxMetrics is the same under Linux, macOS, and Windows. The screen captures presented below were taken on Windows 10.



Since OxMetrics provides the data which other applications analyze, and receives all the text output and graphics which you create in (say) PcGive, you should begin by briefly learning how OxMetrics works.

Once OxMetrics has started, you will be able to load data, create graphs, and transform data using the OxMetrics calculator or algebra. Then, use the Model command on the Model menu to start econometric modelling using, for example, PcGive. When you exit OxMetrics it will automatically close all active applications which depend on it. In some of the coming tutorials we shall be using OxMetrics as a stand-alone program. There are many interesting things to be done even without using additional applications!

2.1 Registering OxMetrics



If you install OxMetrics normally, it will already be registered, and you can start working immediately. An unregistered version will show the registration dialog on screen, as displayed above.

Enter the licensing code and your name in the respective fields (you must enter the code provided by the distributor).

An *OxMetrics Enterprise* licensing code enables most applications. Otherwise, many applications have their own additional licence code, which can be entered in the dialog. Once OxMetrics has the first registration code, the above dialog will not appear automatically anymore. To enter additional codes, activate the Help/Registration menu.

We trust that you use a legal copy of OxMetrics and client applications. That will allow us to keep developing the programs.

2.2 Loading and viewing the tutorial data set

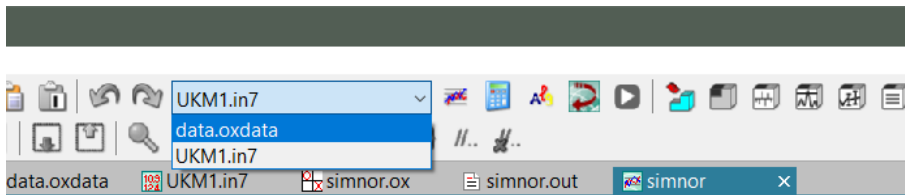
Without data, there is not much that can be done, so the first step is to load data. Many tutorials use a data set called `data.oxdata`.

The data file contains artificial data on consumption, income, inflation and output, denoted by CONS, INC, INFLAT, and OUTPUT respectively. OxMetrics can handle a wide range of data files, among them Excel files. You can also cut and paste data series directly from a spreadsheet (but not formulae).

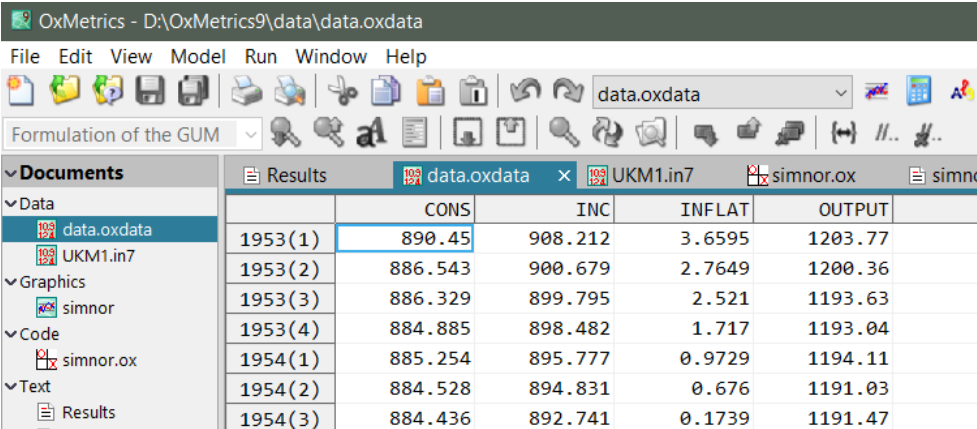
We shall load this tutorial data set here. Access the File menu in OxMetrics, and choose Open. If you installed in the default directory structure, the data will be in the directory `\Program Files\OxMetrics9\data`, so locate that directory and select the `data.oxdata` file. (You will also see pairs of `.in7/.bn7` files – this was the default format in OxMetrics 8, and remains supported. In that case load the `.in7` file.)

Tip By default, Windows does *not* show extensions of registered file types. Those registered by OxMetrics extensions are displayed, however, to facilitate discrimination between the different types. If you wish to see all your registered extension, start the Windows Explorer. Select the option labelled: File name extensions, which is listed under View.

After loading, the data file becomes the currently active database for plotting and transformations. More than one database can be loaded, and the name of the currently active database is shown in the toolbar. If more than one database is open, this toolbar box can be used to switch active database:



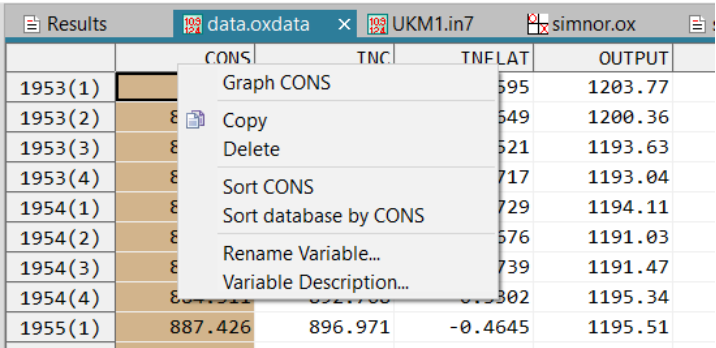
A notification of the loaded file is written to the Results window, so the loaded database does not acquire the focus. To see the database, click on it in the workspace. Although data values are stored internally with about 15 to 17 digit accuracy (8 byte floating point values), the data are displayed with 6 digits only in the spreadsheet:



On the bottom right of the OxMetrics window, in the status bar, the observation under the cursor is shown with full accuracy:

CONS[1953(1)]	890.4495849609375
---------------	-------------------

Double clicking on the variable name shows the name and documentation of the variable, both of which can be changed. For the CONS variable:



The data can be manipulated, much like in a spreadsheet program. Pressing the left mouse button, and keeping it depressed, then dragging the mouse highlights a block of data which can be copied to the clipboard for insertion in another part of the database (click on the two-pages icon for copy, or the `Ctrl+C` key; paste is the clipboard icon or the `Ctrl+V` key).

Right-clicking on the variable name brings up a context menu related to that variable, e.g. to quickly graph it. Here it is shown for CONS:

1953(1)				FLAT
1953(2)	8			5595
1953(3)	8			7649
1953(4)	8			521
1954(1)	8			717
1954(2)	8			9729
1954(3)	8			676
1954(4)	8			1739
1955(1)	887.426	896.971	-0.4645	
1955(2)	889.556	901.406	-0.3819	

Two clicks will bring up the inline editor:

1954(4)	884.311	892.768	-0.3302	1195.34
1955(1)	887.426	896.9713745117188	545	1195.51
1955(2)	889.556	901.406	-0.3819	1198.2

Right clicking shows the context menu. Select Edit Value to bring up a revision box, where corrected values can be entered (or missing values set to an observed outcome):

1953(4)	884.311	892.768	-0.3302	1195.34
1954(1)	885.254	895.777		
1954(2)	884.528	894.831		
1954(3)	884.436	892.741		
1954(4)	884.311	892.768		
1955(1)	887.426	896.971		
1955(2)	889.556	901.406		
1955(3)	890.659	901.479		
1955(4)	894.079	905.117		

INC[1955(1)]

896.9713745117188

☐ Missing Value

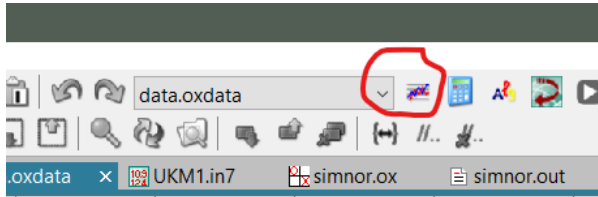
OK Cancel

2.3 OxMetrics graphics

The graphics facilities of OxMetrics are powerful yet easy to use. This section will show you how to make time plots and scatter plots of variables in the database. OxMetrics offers automatic selections of scaling etc., but you will be able to edit these graphs, and change the default layout such as line colours and line types. Graphs can also be saved in a variety of formats for later use in a word processor, or for reloading into OxMetrics.

2.3.1 A first graph

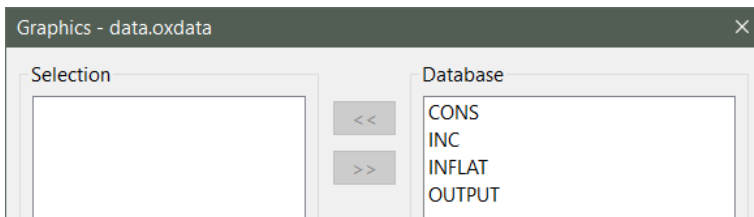
Graphics is the first entry on the Model menu. Alternatively, click on the graphics icon on the toolbar:



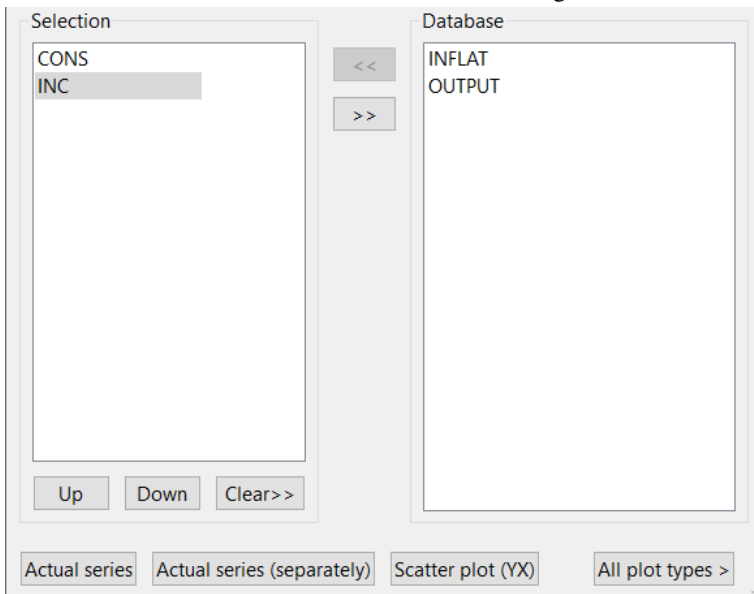
Either way brings up the following dialog box. This is an example of a dialog with a multiple selection list control. In such a list, you mark as many items as you want. Here we mark all the variables we wish to graph. With the keyboard, you can only mark a single variable (by using the arrow up and down keys), or range of variables (hold the shift key down while using the arrow up or down keys).

With the mouse there is more flexibility:

- single click to select one variable;
- hold the Ctrl key down and click to select additional variables;
- hold the Shift key down and click to extend the selection range.



Here we select CONS and INC and then press the << button. This moves the variables from the database list to the selection list, activating the buttons at the bottom:



Press Actual series. Now the graph appears, which looks very much like [Figure 2.1](#). The only difference is the position of the legend. You can pick that up with the mouse,

and move it to another position in the graph as desired.

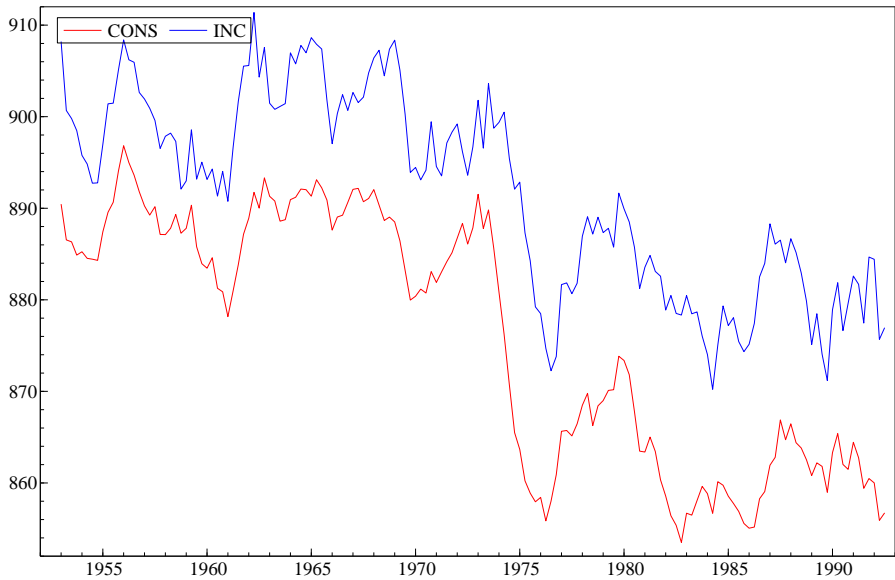
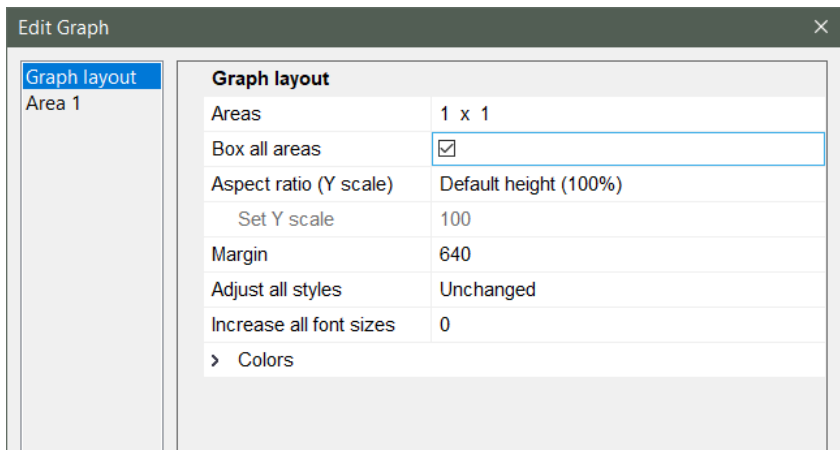
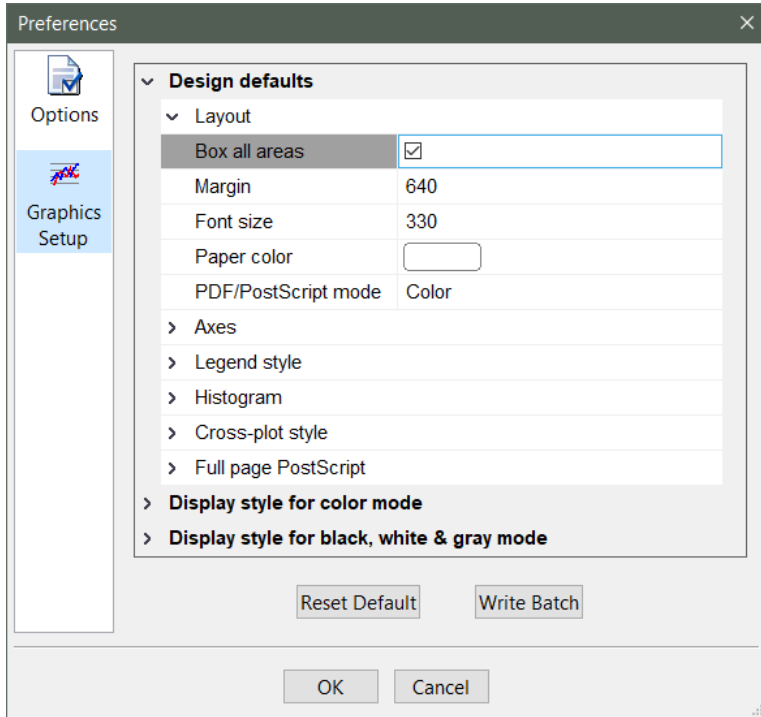


Figure 2.1 Time plot of CONS and INC

Most graphs in this book are boxed in. You can change the default, for the current graph only, by double clicking on the graph (or selecting Edit/Edit Graph), then clicking on Graph layout in the left-hand column, and activating Box all areas, as shown here:



Alternatively, choose Graphics from the Model/Preferences menu, and activate Box all areas:



This will change the default for all future graphs that are made with OxMetrics.

Note that, when the graph is active, and the mouse is moved over the graph, the status bar at the foot of the OxMetrics window displays the graph coordinates. Inside a graph these are the ‘real-world coordinates’ (X,Y), the actual data values. Outside these are ‘pixel coordinates’ (pX,pY): a graphics window in OxMetrics runs from (0,0) in the bottom left, to (15000,10000) in the top right (further explanation is in [Chapter 12](#)).

2.3.2 Multiple graphs

One of the powerful features of OxMetrics is the ability to draw multiple graphs simultaneously on-screen within one graphics window (and you are not restricted to just one graphics window). We shall now get two graphs on screen, with the second a cross-plot of CONS and INC. Click on the Graphics toolbar button, note that CONS and INC are still selected (you can also variables to the opposing listbox by double clicking on them, or empty the selection by pressing Clear). The first variable in the selected list, CONS here, will be the Y variable (if CONS is not on top you can select it, and press the Move up button). Click on Scatter plot (YX) to create the plot.

If you accidentally did it wrong, click in the new sub plot (or plot *area*, as we tend to call it), selecting the entire second area, then press delete to remove it again.

A useful aspect of OxMetrics is that graphs can be edited and features added while they are on screen, and after adding other graphs if desired. Double click on the scatter plot graph, select Regression in the Edit Graphics dialog, and add one regression line

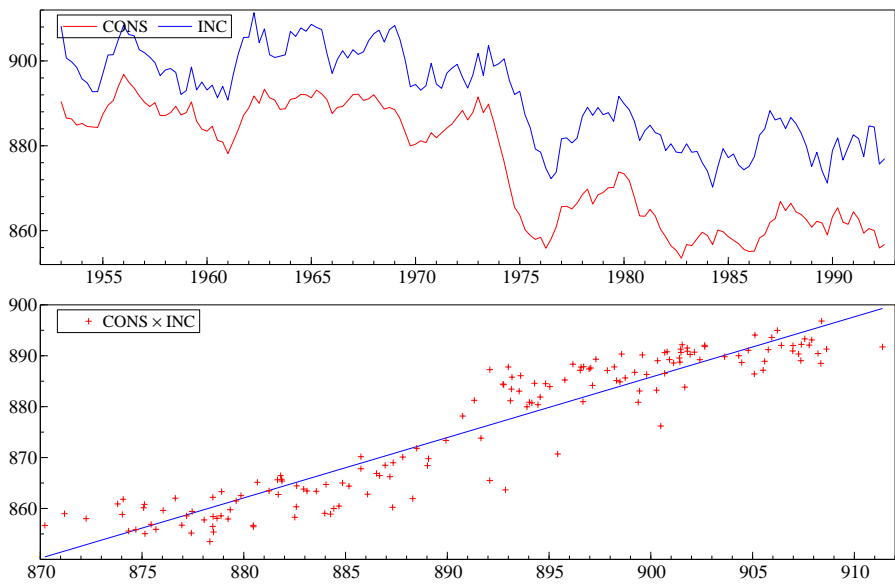
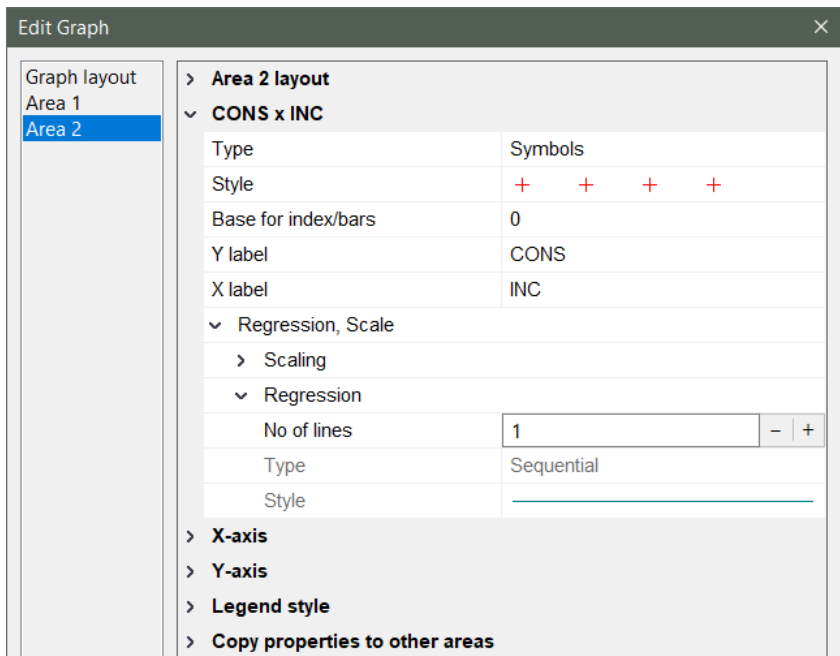


Figure 2.2 Multiple graphs

as shown:



The final result is in [Figure 2.2](#). The editing possibilities are manifold: [Chapter 5](#) provides a detailed discussion, but you can always play around with graphs at your leisure.

Finally, you can undo and redo the changes that were made. In this case, Undo will remove the graphics line again. Graph saving clears the undo/redo buffer, to avoid excessive memory use.

2.3.3 Graph saving and printing

To print a graph directly to the printer when the graphics window has the focus, click on the printer icon in the toolbar. You can preview the result first using the Print Preview command from the File menu. If you have a PostScript printer, you can save the graph to disk as PostScript, and then print it from the command line.

Graphs can be saved to disk in various formats:

- PDF file (**.pdf**), which is the format used to produce all the graphs in this book;
- Scalable Vector Graphics (**.svg**), which can be inserted into web pages, as well as Microsoft Word and Excel using Insert/Pictures;
- OxMetrics Graphics File (**.gwg**);
- Enhanced metafile (**.emf**, Windows); These can easily be inserted into Microsoft Word using Insert/Pictures.
- Encapsulated PostScript (**.eps**);
- PostScript (**.ps**), this is like EPS, but defaulting to a full page print.
- Portable Network Graphics (**.png**), which is a bitmap format that may be useful for insertion in web pages if SVG cannot be used;

The GWG format is particular to OxMetrics; no other program can read it and no printer can handle it. OxMetrics requires the GWG format, because it needs to be able to allow re-editing of the graph when it is reloaded; the other formats do not store sufficient information. *When you save a graph in any format, the GWG file is automatically saved alongside it. Then, when loading a previously saved PDF file (say), OxMetrics can use the GWG file to reload the graph.*

2.3.4 Using the clipboard for graph pasting

There is an important distinction between copying graphs for use in internal OxMetrics graphs, or for external use.

To paste the graph into Word, for example, you can use Edit/Copy Metafile to Clipboard, and then paste it into Word. A metafile stores the actual Windows commands that were used to draw the graph, and thus scales well. The alternative is to copy the bitmap to the clipboard.¹

The standard copy command is used for internal paste. If no area is selected, the whole graph is copied, otherwise the selected area only.

Try this by clicking on the second area in the current graph. Once selected (shown by a hatched boundary) copy the area to the clipboard. Then first deselect the area by clicking somewhere in the margin of the graph. A subsequent paste will add a copy as the third area. Next, select the first area and paste again: this inserts the cross-plot into the time-series plot (not a useful graph).

¹We use L^AT_EX for our typesetting system, and save all graphs as .pdf files.

2.4 Calculator

Two options are available for transforming data: by algebra or by a dialog approach, which mimics the operation of a pocket calculator. We begin with the latter, as it is the simplest.

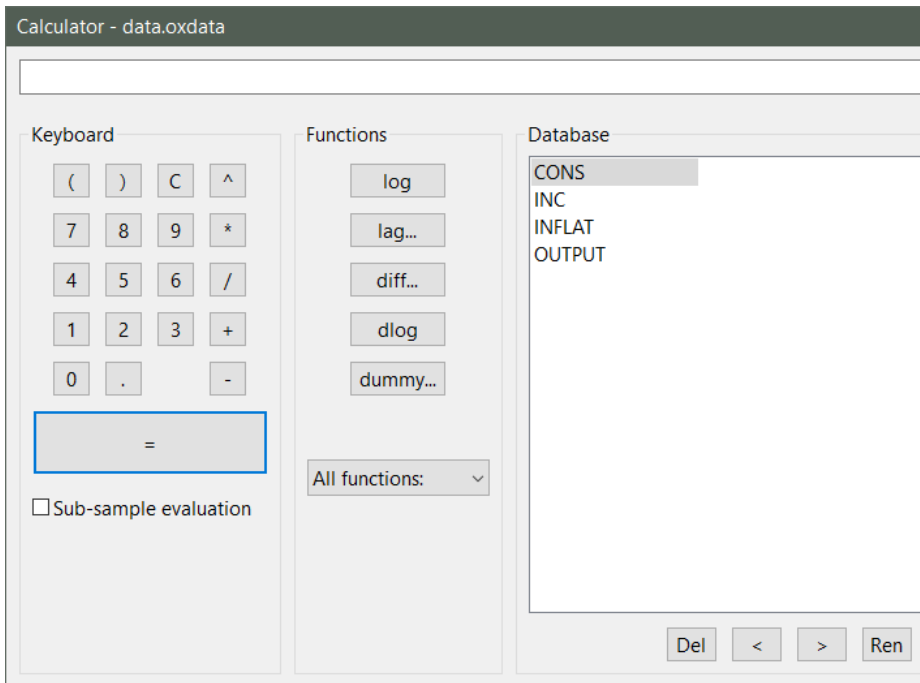
Please note that OxMetrics and client applications are sensitive to the case of the variable names so that ‘cons’, ‘Cons’, and ‘CONS’ are treated as different variables. This can be useful for distinguishing real (lower case) from nominal (capitals) variables, or logs from levels, etc.

In several cases OxMetrics will offer a default name for a newly created variable:

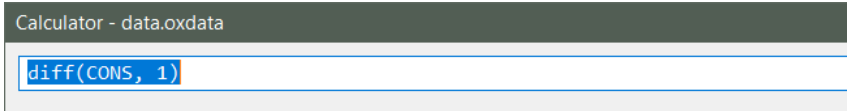
- prefixes of ‘D’ for differences, e.g. DCONS;
- prefixes of ‘L’ for logarithms, e.g. LCONS;
- suffix ‘_n’ for n -period lags, e.g. CONS_1.

So can you guess what DLCONS_1 is likely to be?

The aim is to build up an algebraic expression for the transformation (which is valid Algebra code: see §2.5). Press on the Calculator button leading to the capture shown below (or via the Model menu and Calculator choice).



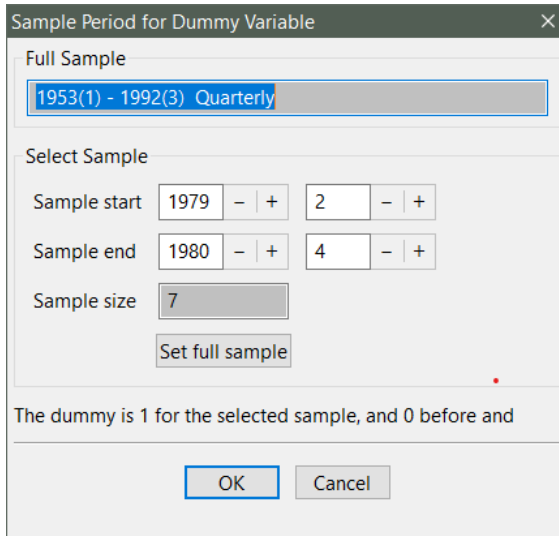
The first transformation is to take the first difference of CONS. Click on the CONS variable just to highlight it (don’t double click), and then on the diff button, accept a lag length of one, to see in the top part of the dialog:



Click on the large button with the = sign, and accept the default name of DCONS, which will be created in the database, and added to the list of variables. A two-period difference is just as easy to compute: select CONS, press the diff button and change the length of the differencing period.

Another transformation to try: $INC - CONS$. Double click on INC, click on the minus button, and then double click on CONS. The expression now reads $INC - CONS$, press on = to evaluate. Name the variable SAVING (you could use a name like $INC - CONS$, but must enclose such a name in double quotes when using it in expressions).

Finally, we create a step dummy (or indicator variable), where the step lasts from 1979(2) to 1980(4). We need zeros outside that period, ones inside. Click on the dummy button, and enter:



Click on OK, and then on = to create the dummy. Give it an informative name, such as `s792t804`.

Tip We deliberately gave no examples involving lags. For estimation in applications such as PcGive, lags are best created at the model formulation stage, where the programs will keep track of their presence for dynamic analysis.

Three additional operations can be performed on variables inside the listbox in the Calculator:

- Delete a variable: select and press the button labelled Del, or the Delete key; you will have to confirm the deletion as the variable is eliminated from the database.
- Move variables: use the < or > button to move (groups of) variables up or down. This changes the order in the database.
- Rename a variable: select and press the Ren button; you will be prompted for a new name.

All changes to the database can be undone from the Edit menu.

Exit the calculator (Esc or click on the x button), and graph some of the variables to check whether the transformations are correct. All transformations are logged to the

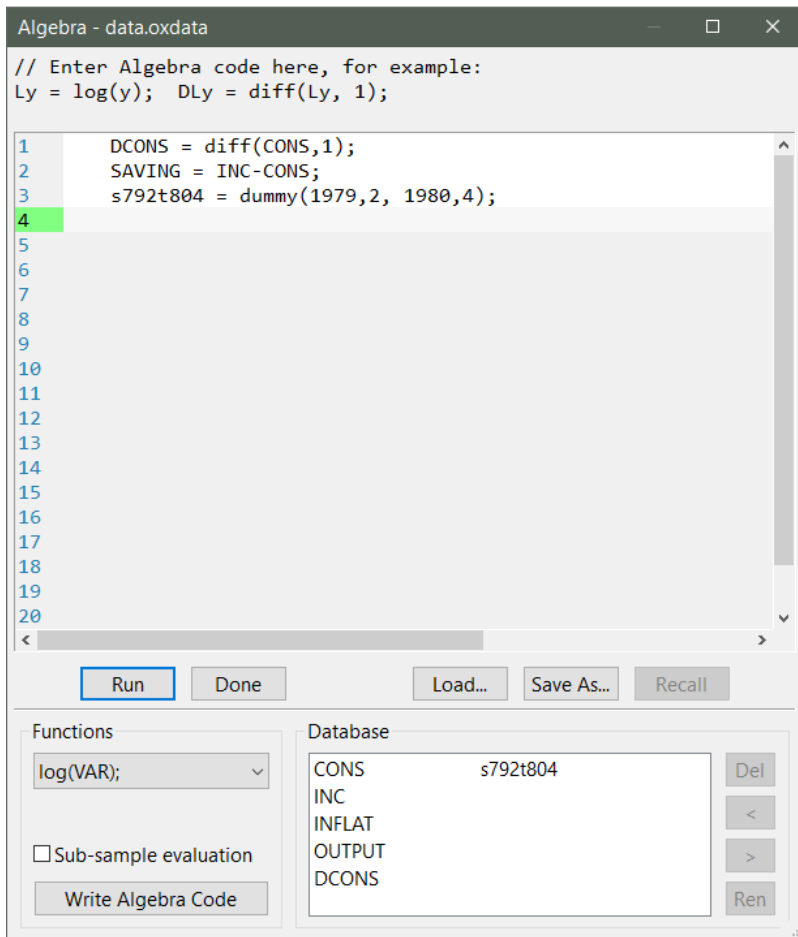
Results window:

```
Algebra code for data.oxdata:
DCONS = diff(CONS,1);
SAVING = INC-CONS;
s792t804 = dummy(1979,2, 1980,4);
```

This leads us to the next topic.

2.5 Algebra

Algebra enables us to do transformations by typing the expressions directly into an editor. It also allows saving and loading from disk of a whole chunk of statements:



As an example, we rerun the transformations of the previous section. To verify if that will work, activate the database, and press Undo until the database is back in its original state. Set focus to the Results window and locate the logged Algebra code in the Results window. There are two ways to quickly run this code:

- select the block of Algebra code, and type Ctrl+A to execute it;
- using the Algebra editor.

Because the former is straightforward, we now discuss the latter method.

Highlight the three algebra lines, and copy them to the clipboard (Ctrl+C). From the Model menu, select Algebra, and paste the code (Ctrl+V) as shown above.

Now press Run. The variables have been recreated. We can inspect the database to see what happened. The algebra code is again logged to the Results window.

Tip The bottom part of the Algebra editor is there to save typing. For example, to insert the code `LCONS=log(CONS) ;`, select CONS in the Database box of the Algebra editor. Next, select `log(VAR)` under Functions, and click on Write Algebra Code.

Tip As in the Calculator, database variables can be deleted, moved and renamed.

The only variable names allowed directly in Algebra are those that would be valid names in computer languages like Ox or C: the first character must be a letter or an underscore, the rest a letter, underscore or digit. Names which do not follow this convention may be used as well, but must be enclosed in double quotes.

Algebra uses database variables as follows: if a left-hand variable is already in the database it will be overwritten, otherwise it will be created. Variables on the right must exist, possibly because of preceding lines of algebra code. Algebra is case-sensitive, meaning that LCONS, LCons and lcons refer to three different variables.

An impulse indicator for 1980(1) can be created as

```
i1980p1 = II(1980, 1);
```

or using the default notation used in saturation estimators, for the next period:

```
i1980p2 = II#1980(2);
```

The same dummy can also be created using the `insample` function and a *conditional assignment*. The `insample` function has four arguments: startyear, startperiod, endyear, endperiod. It returns 1 (or TRUE: everything which is not 0 is TRUE) if the observation under consideration falls within the sample, otherwise it returns 0 (FALSE). The conditional assignment works as follows: the conditional statement (the 'if' part) is followed by a question mark and the 'then' part, which is followed by a colon and the 'else' part. Read:

```
i1980p1 = insample(1980, 1, 1980, 1) ? 1 : 0;
```

as: i1980p1 takes on the value 1 for the observations which are in the specified sample, and the value 0 for the other observations. In this case, the same result can be obtained by writing:

```
i1980p1 = insample(1980, 1, 1980, 1);.
```

An error message pops up if you make a mistake. The error can be corrected on returning to the algebra editor. The text at the top of the Algebra error shows the text of the last error (if one occurred).

Tip A sensible strategy is to store algebra code and basic data only, and recompute transforms during each run: this economizes on storage and facilitates updating analyses when data are revised or corrected.

Tip If you load a database, text file, or graph from disk, and make a change to it, this is shown by a star before the name. When the document is saved, the star disappears.

Do not save the modified data set. Either undo the changes, or quit the data set, and reload the original `data.oxdata`. Algebra is documented in [Chapter 10](#).

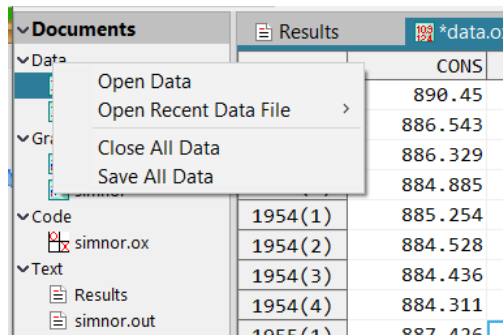
This completes the getting-started chapter. Before giving some examples on modelling with OxMetrics, we briefly discuss the workspace window. The remainder of the tutorial are more detailed examples on graph editing, graphics, data loading and saving, and data transformations.

2.6 The workspace

Often, after working in OxMetrics for a while, there are many open documents, with graphs, data and results. The workspace on the left helps navigating between these, by showing which documents are open in OxMetrics.

An example is shown at the start of this chapter. Here there are two data files open (if a document has been modified it indicated by the * in front of the name). There is one graph, a code file `simnor.ox`, and the finally the Results window. The available apps are listed under Applications (with all interactive modelling packages such as PcGive, STAMP, etc. now grouped under the Model header).

Right clicking on the document category (Data, Graphics, Code, or Text) gives a context menu. For Data it is:



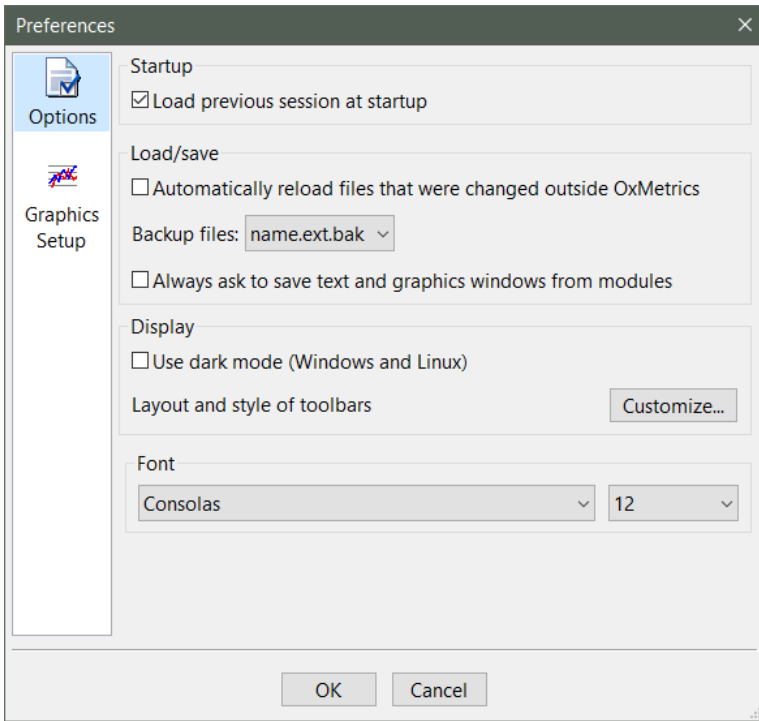
while right clicking on the document name gives:

2.7 Preferences

General settings are controlled from Options on the Preferences dialog.

For Windows and Linux this allows a switch to dark mode (requiring a restart of OxMetrics). Under macOS dark mode is set in the operating system. It also allows for

a change of the default fixed width font and size. But the text size can also be controlled directly from the View menu.



Chapter 3, which is the next in line, introduces the OxMetrics applications. This includes an example on estimating a GARCH(1,1) model using PcGive. Because OxMetrics behaves very similarly on Windows, OS X and Linux, we shall not explicitly distinguish between the different platforms in the remainder.

Chapter 3

OxMetrics Applications

3.1 OxMetrics Modules

There is an increasing number of computer programs which cooperate with OxMetrics to deliver an easy-to-use yet powerful user experience. Such applications implement different services, and may use OxMetrics:

- to receive the data for analysis;
- to store the text output from the analysis;
- to display the graphical output.

At the time of writing, the OxMetrics compatible applications accessed via the Model command include:

- CATS (cointegration analysis, [Doornik and Juselius, 2018](#))
- G@RCH (financial modelling, [Laurent, 2021](#))
- PcGive (econometric modelling, [Hendry and Doornik, 2021](#)), including PcNaive (Monte Carlo analysis, [Doornik and Hendry, 2021](#))
- STAMP (time-series modelling, [Koopman, Harvey, Doornik, and Shephard, 2013](#))
- X12arima (time-series modelling and seasonal adjustment, [Findley, Monsell, Bell, Otto, and Chen, 1998](#))

Further applications include:

- Ox Professional: OxRun and OxPack ([Doornik, 2021](#))
- Datafetch (downloading data from the internet)
- Mulcom (multiple comparison)

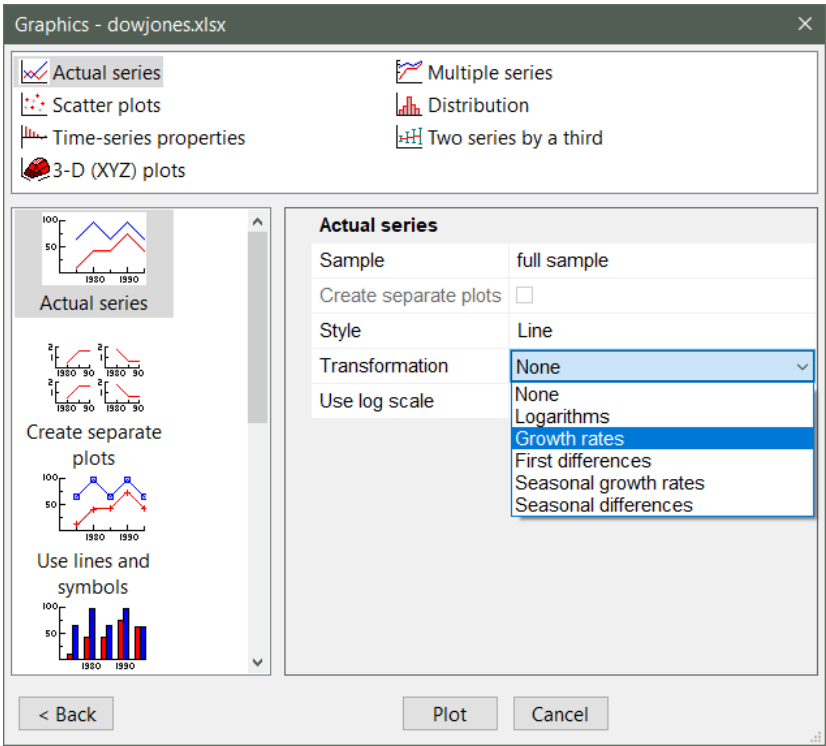
In addition, there are interactive packages written in Ox which can be run via OxPack.

It is possible that only a subset of the applications are installed. The remainder of this chapter gives a brief introduction on how to use PcGive. Even if PcGive is not one of the applications you plan to use, it will still be useful to read it: most applications work in a similar way. Before we start, we load another database, and digress by talking about financial data.

3.2 Financial data

Load the `dowjones.xlsx` data set from the `OxMetrics9\data` folder. The `DOWJONES` variable holds the weekly Dow–Jones index (Dow Jones Industrial Average): close at midweek¹ from January 1980 to September 1994, 770 observations in total.

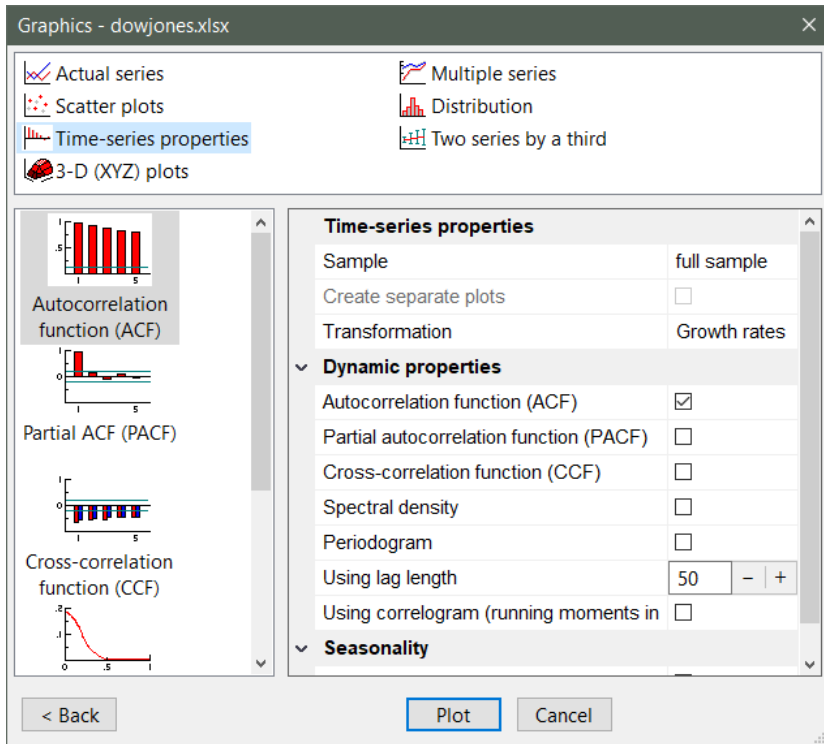
We are interested in the returns, $\log Y_t - \log Y_{t-1}$, already in the database as `DL-DOWJONES`. First, we illustrate the the returns can also be plotted directly from the levels. Make sure `dowjones.xlsx` is the active database. Click on Graphics, select `DOWJONES`, and click on `All plot types>`. Then select transformation `Growth rates`:



The outcomes are given in the top panel of [Figure 3.1](#). The large negative return of -17.4% corresponds to the Black Monday crash of 19 October 1987. The data show behaviour that is typical of financial time-series: clustering of volatility and so-called thick tails: there are more observations in the tail than can be expected from a normal distribution.

Next, activate Graphics again, using the same variable. Select time-series properties and note that the transformation is still set to growth rates. Click on ACF, and set the lag length to 50:

¹The figures are for Wednesday, or Tuesday if the stock market was closed on Wednesday. The data are from www.djindexes.com.



The outcomes are in the second panel of top panel of **Figure 3.1**, indicating that there does not seem to be much memory, if any, in the returns. This should be expected. Next, use the calculator, and create the squared returns. The ACF of the squared returns are in the final panel of **Figure 3.1** (remember to reset the transformation back to none). This plot suggests that there is indeed some persistence in volatility.

3.3 Weekly and daily data

When moving the cursor over the graph of the actual returns, the status bar at the bottom can be seen to display the date in the plot. There are too many observations for an accurate reading, but the large negative shock can be pinpointed in October 1987.

Inspection of the database reveals that the first variable is called Date, and shows the same dates as in the row labels:

Results			dowjones.xlsx		
	CONS	INC		Date	DOWJONES
1953(1)	890.45	908.212		1980-01-02	824.57
1953(2)	886.543	900.679		1980-01-09	850.09
1953(3)	886.329	899.795		1980-01-16	865.19
1953(4)	884.885	898.482		1980-01-23	877.56

Double-clicking on the variable name reveals that it is of type Date. Changing it to Default:

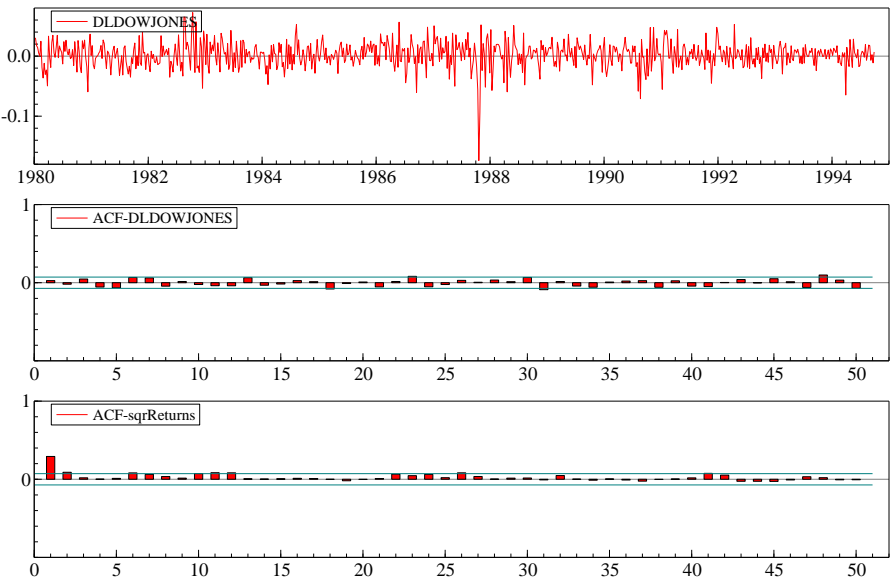
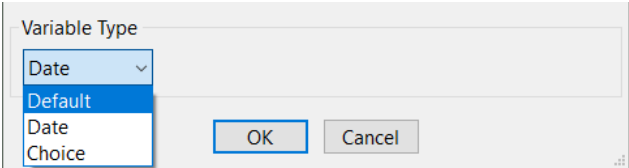


Figure 3.1 Dow Jones returns, ACF of returns, ACF of squared returns



sets the labelling to an undated dataset, and the Date values to large numbers:

*dowjones.xlsx x					
	Date	DOWJONES	LDOWJONES	DLLOWJONES	d408
1	2444241	824.57	6.71486	missing	0
2	2444248	850.09	6.74534	0.0304802	0
3	2444255	865.19	6.76295	0.0176069	0
4	2444262	877.56	6.77715	0.0141962	0
5	2444269	881.91	6.78209	0.00494468	0
6	2444276	881.83	6.782	-9.07163e-5	0

These are the numbers that OxMetrics and Ox use to represent dates. Any fractional value represents fractional time (so 0.5 is 12:00 and 0.75 18:00 on the 24-hour clock).

Undo the changes to get back to the dated database. With weekly data, there are some years that have 52 weeks, and others with 53. Therefore, the method of using a fixed frequency, as used for annual, quarterly and monthly databases, does not work. Instead, a database can now be dated:

- the first column must be of type Date,
- the first column holds date values (there are several Algebra functions to help creating these),

- the optional fractional part of this indicates time,
- the first and last observation must be valid, i.e. cannot be missing.

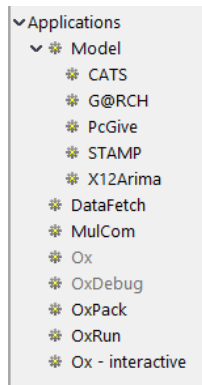
These criteria are satisfied in `dowjones.xlsx`, and the Excel dates are translated in Ox dates when reading the file (and the other way round when saving). Database!— with dates

Next, to illustrate the modelling sequence in OxMetrics, we estimate a GARCH model without paying too much attention to the actual outcomes.

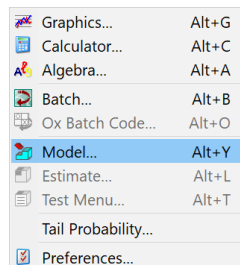
3.4 PcGive

PcGive is an interactive program for dynamic econometric modelling. Like other interactive modelling packages, it can be started in three ways:

- By clicking on the Model entry under Applications in the workspace on the left-hand side:



- Via Model on the Model menu:

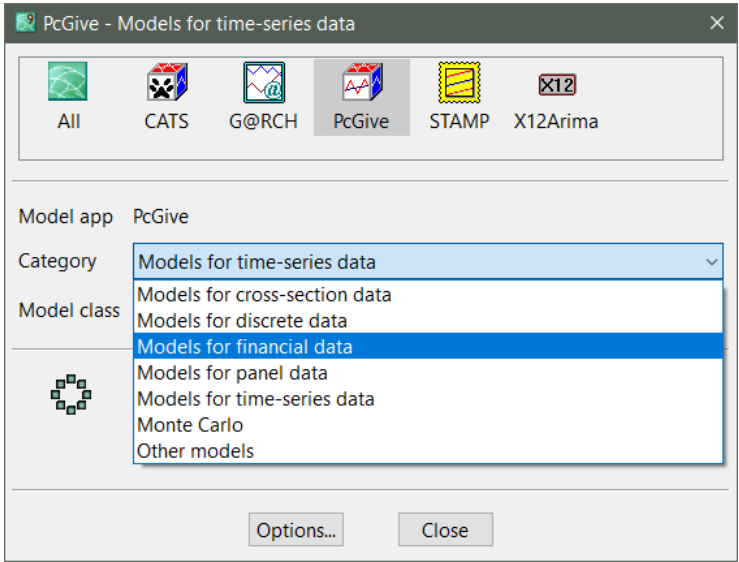


- Using the Model toolbar button:

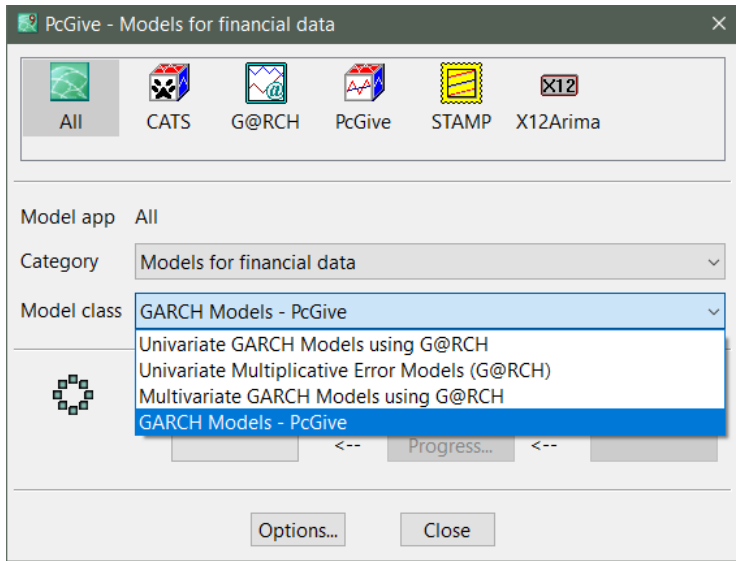


The modelling dialog gives access to all the modelling features of the OxMetrics applications. The application will use one of the databases loaded in OxMetrics.

The first step is to select a model category, and for the category a model type. The images at the top allow you to see the choices for all available applications, or to restrict it to a specific application. Select Models for financial data:



When we set the Model app to All by clicking on the corresponding picture, we can see all possible financial models categories: select GARCH Models – PcGive.



3.4.1 Formulate a model

Click on the large Formulate button, to see the standard formulation dialog. On the right-hand side is the list with the database content, while the box on the left, empty as yet, is for the model. Double click on the DLLOWJONES variable:

The screenshot shows the 'Formulate - GARCH Models - dowjones.xlsx' dialog box. It has two main sections: 'Selection' on the left and 'Database' on the right. In the 'Selection' section, there is a list with 'Y DLDOWJONES' and 'X Constant'. Below this list is a dropdown menu set to 'Use default status' and a 'Set' button. In the 'Database' section, there is a list containing 'Date', 'DOWJONES', 'LDOWJONES', 'DLDOWJONES', and 'd408'. Below this list is another list containing 'Constant' and 'Trend'. Between the two sections are buttons for '<<', 'Lags 0 to:' (with a dropdown set to 0), '>>', and 'Clear>>'. At the bottom of the dialog are 'Cancel' and 'Next >' buttons. The 'Next >' button is highlighted with a blue border.

This dialog is used by most applications to formulate the linear part of the model. Other actions that can be taken are:

- Change the default lag length when selecting. The choices are: lag 0 to a specified lag, just the specified lag, or none.
- Empty the entire model formulation by pressing Clear>>.
- The box immediately below the database contains the so-called ‘special’ variables. These are made available even when not present in the database, and may be treated in a special way.
- Below that, still on the database side, is the option to change databases, if more than one are open in OxMetrics. But the model only works on one database.
- On the left-hand side, below the selection (the model formulation), is a drop-down box to change the status of selected variables. It currently reads Use default status, which means that variables are added to the model as such: the first addition to an empty model is the dependent variable, the rest are regressors in the mean.

For the GARCH model there are three types:

Y the endogenous (dependent) variable, by default the first,

X regressor, the default for all other variables,

H a regressor that enters the conditional variance equation.

If another status is selected in the drop down box, that will be used for the added variables instead of the default (but lagged variables can never be the dependent variable).

To change status of variables that are already in the model: highlight the variables in the selection box, choose a status, and press the Set button. Note that it is also possible to change status by right-clicking on a variable.

- F3 enables searching for a variable.
- Finally, the last drop-down box below the selection allows the recall of a previously formulated model.

Press on Next to set the GARCH model properties. There are quite a few options, but the default of a GARCH(1,1) model suffices:

Model Settings - GARCH Models

GARCH(p,q)

p =

1

q =

1

EGARCH

☐

Non-normal error distribution

☐

▼ GARCH variations

Asymmetric GARCH☐

Threshold GARCH☐

No conditional variance in mean☒

h_t in mean☐

sqrt(h_t) in mean☐

log(h_t) in mean☐

› GARCH parameter restrictions

› Advanced options

Back

Cancel

Next >

The final step is the estimation period, where we can accept the default again:

Estimate - GARCH Models

Choose the estimation sample:

Selection sample	1980-01-09 - 1994-09-28
Estimation starts at	1980-01-09
Estimation ends at	1994-09-28
Less forecasts	0

Choose the estimation method:

Estimation method:	Maximum Likelihood
Recursive estimation	<input type="checkbox"/>
Initialization	0

Back Cancel Next >

Click on OK and the output appears in the results window.

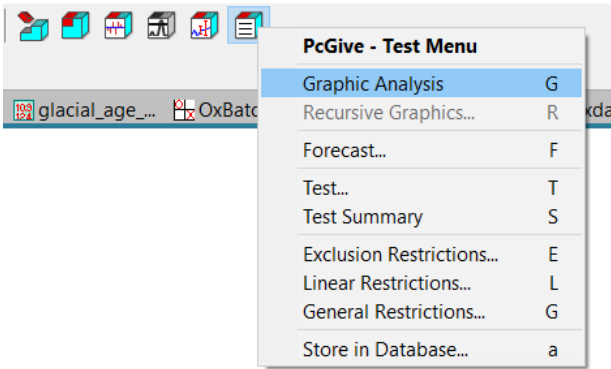
To end this section, consider all the buttons available for modelling:



They are respectively:

- Model
- Estimate
- Graphic Analysis
- Recursive Graphics
- Forecast
- Test menu

Graphic Analysis is available from the toolbar button as well as the Test menu button. To see this, click on the latter:



and accept the default choice. The figure is as in [Figure 3.2](#). Note that in the middle panel (the standardized residuals), we zoomed in on the crash and changed the style to index lines. It is quite likely that this large outlier affects the quality of this model ([Doornik and Ooms, 2008](#) and [Doornik and Ooms, 2005](#)).

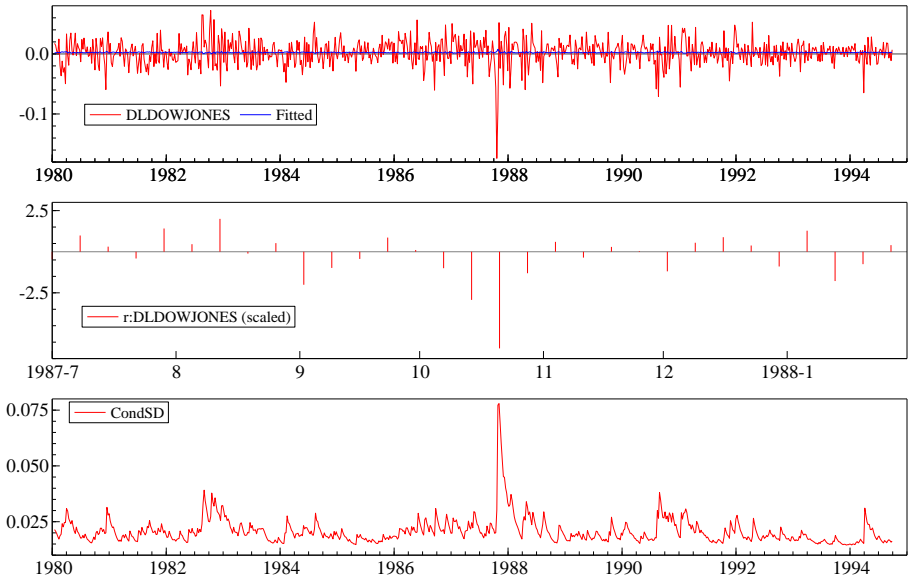


Figure 3.2 GARCH(1,1) model for Dow Jones returns

OxMetrics Tutorials

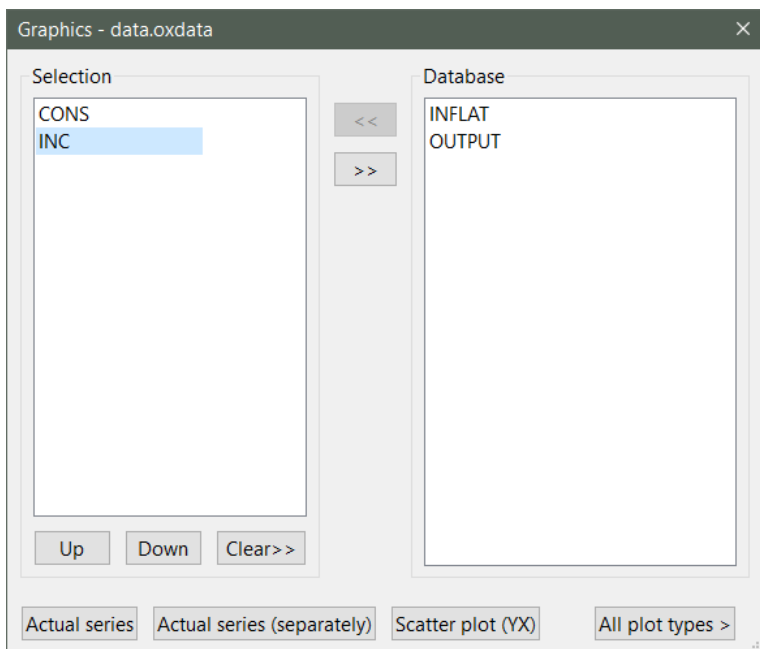
Chapter 4

Tutorial on Graphics

This chapter discusses the various types of graphs which are available for (exploratory) data analysis. **Chapter 3** already gave some examples using weekly Dow Jones data; Ch. 5 is mainly concerned with the mechanics of changing the appearance of graphs.

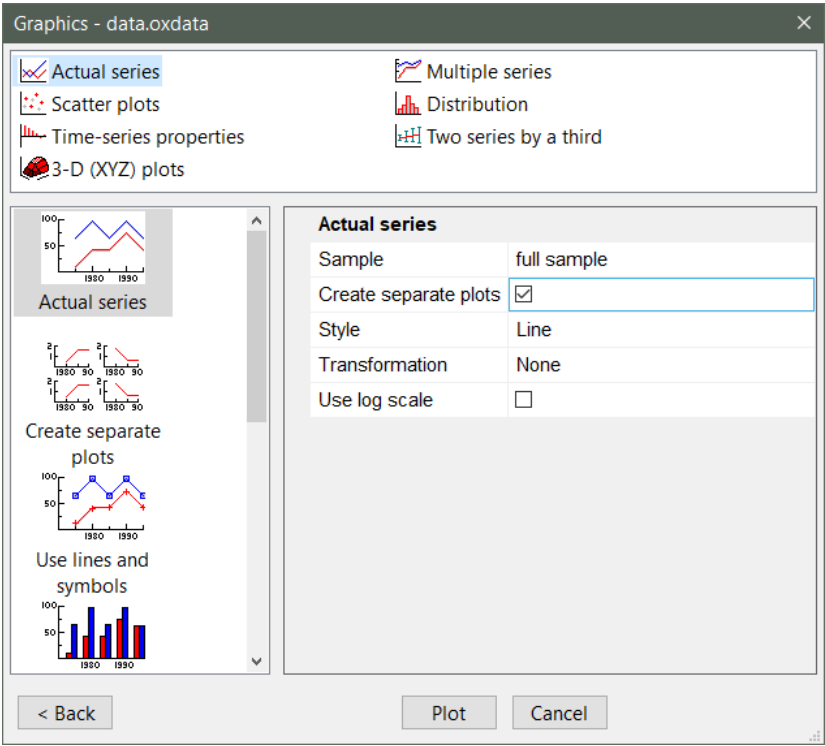
If you're not inside OxMetrics at the moment, restart and load the tutorial data set `data.oxdata`. Note that recently loaded files are listed at on the File menu. Recent data files can also be accessed by right clicking on the Data folder in the workspace. Create `DCONS` as the first difference of `CONS`, as shown in the Getting Started chapters for your platform (the remaining screen captures are all made under Windows).

4.1 Descriptive graphics



Activate the Graphics choice on the Model menu, select CONS and DCONS, by marking them in the Database list box (remember: to select two (or more) variables, click on CONS, then hold the Ctrl key down and click on DCONS), then press the << button to move them into the selection list (the list on the left-hand side). The screen capture is shown on the previous page.

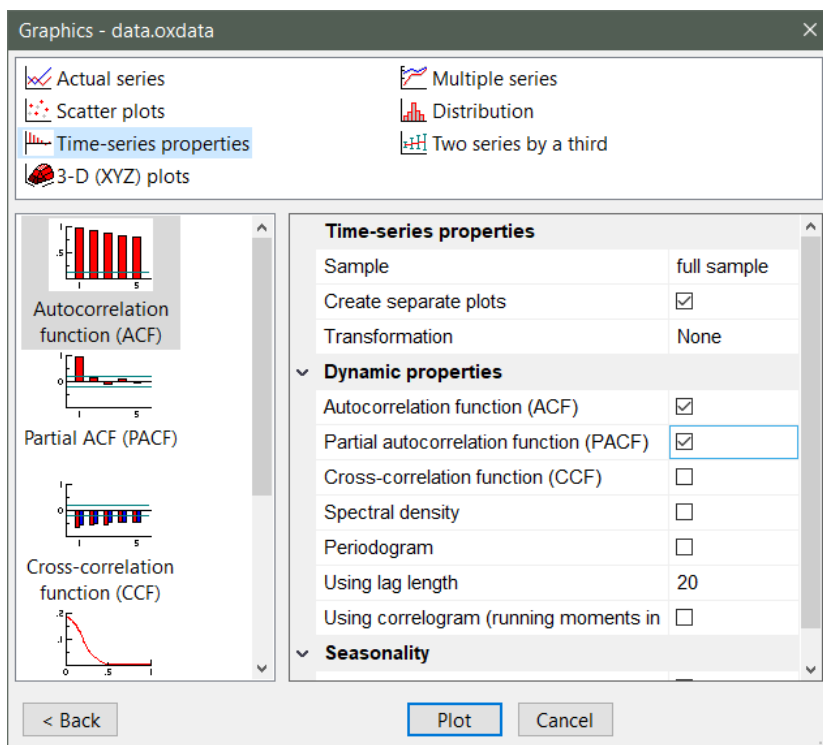
Next press All plot types >. Select Actual series at the top, and at the left, and tick the Create separate plots box on the right:



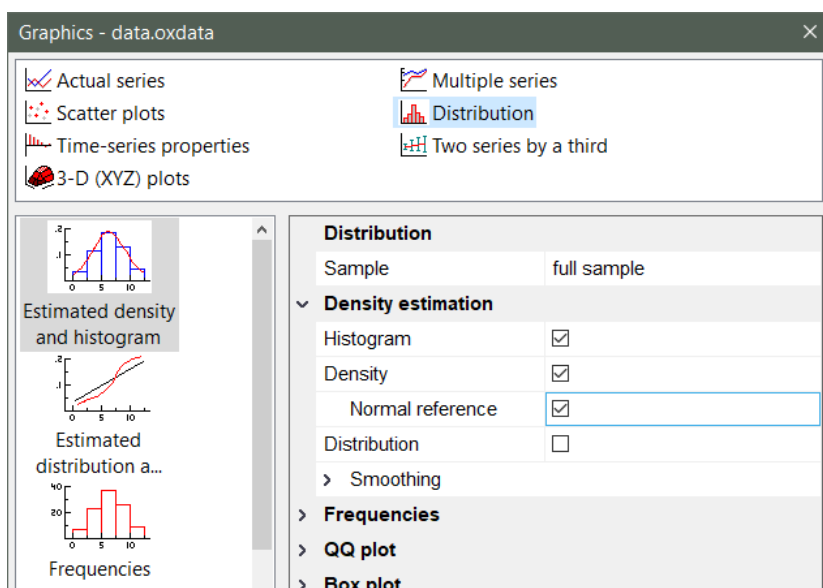
Then press Plot, to create the graph, and Cancel to exit the dialog. The output should correspond to the top two graphs of [Figure 4.1](#).

To add some graphs of the time-series properties, restart Graphics (note that CONS and DCONS are still selected), press All plot types >, and choose Time-series properties as the graph type.

Click Next to choose both ACF for the sample autocorrelation function (or covariogram; you can also choose to plot the correlogram instead, which uses running means and variances, see §8.5) and PACF for the partial autocorrelation function. You can also change the length of the ACF and PACF if you wish. Tick both ACF and PACF. If necessary, mark Create separate plots and set the lag length to 20:



Click on Plot, now selecting Distribution as the plot type. Mark Normal reference:



and plot again.

The options we used for each variable correspond to a time-plot showing its historical behaviour, a correlogram and partial correlogram which reflect its autocorrelation, and a histogram, with estimated density and the normal distribution for reference to evaluate the distributional shape. This generates six graphs in total: see Figure 4.1. A detailed description of the formulae underlying these graphs is given in Chapter 8.

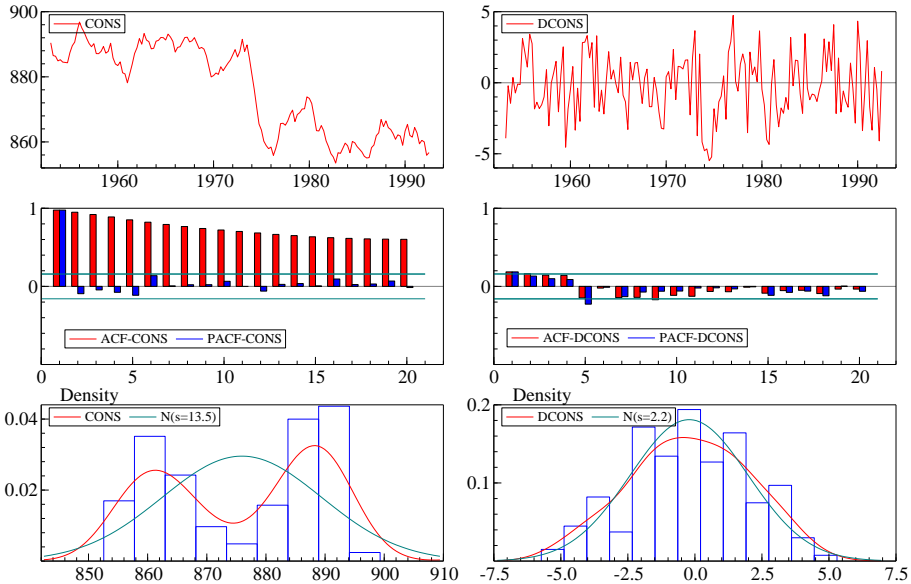


Figure 4.1 Descriptive graphics of CONS and DCONS

4.2 Actual series with optional transformations

Plots the actual values of all selected variables against time (or the observation index for undated series), together or each in a separate graph. If there are missing values, these show up as gaps in the line.

- Create separate Plots

This creates as many graphs as there are series.

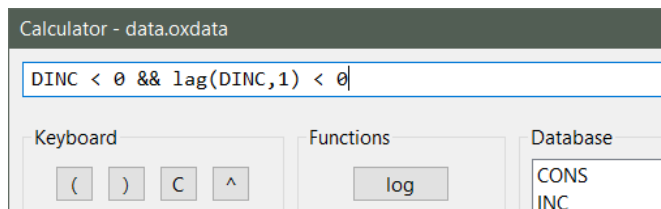
- Style

- 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

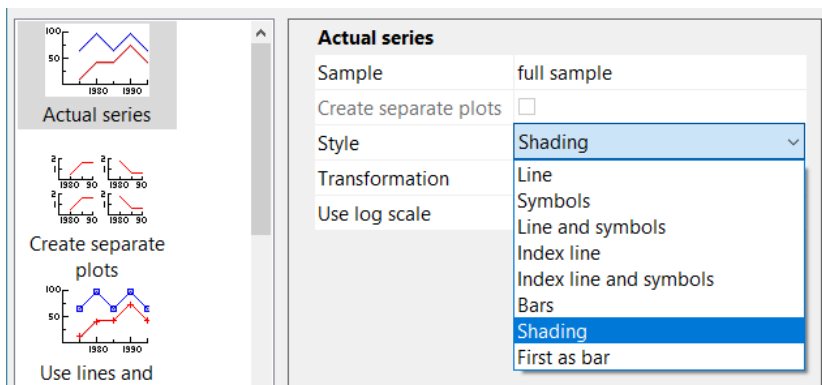
- Transformation:

- Logarithms: natural logs of the series: $\log(y_t)$
- Growth rates: $\log(y_t - y_{t-1})$
- First differences: $\Delta y_t = y_t - y_{t-1}$
- Seasonal growth rates: $\log(y_t - y_{t-s})$, $s = 4$ for quarterly data, $s = 12$ for monthly data,
- Seasonal differences: $\Delta_s y_t = y_t - y_{t-s}$
- Use log scale: assumes the data are in natural logs and powers up the axis to reflect the log scale.

We now give an example of a plot that uses recession shading based on INC. Create DINC as the first difference of INC. Then create a ‘recession’ variable which is one when the current and previous change in INC is negative. The Calculator expression is:



Next, starting from a new Data Plot window (closing the existing one first), select CONS and INC in the Graphics dialog, and press Actual series. Then, right-click on the graph, choosing Add Graph from the context menu (also see §5.9). This brings up the Graphics dialog again, now with title Graphics - data.oxdata - Add to area 1. Remove CONS and INC from the selection, and add the just created recession variable. Next, click on All plot types, select Actual series, and change the style to shading:



Click on Plot, then cancel the Graphics dialog. The shading is done in the next available color, and a lighter would be better (see §5.7), see Figure 4.2.

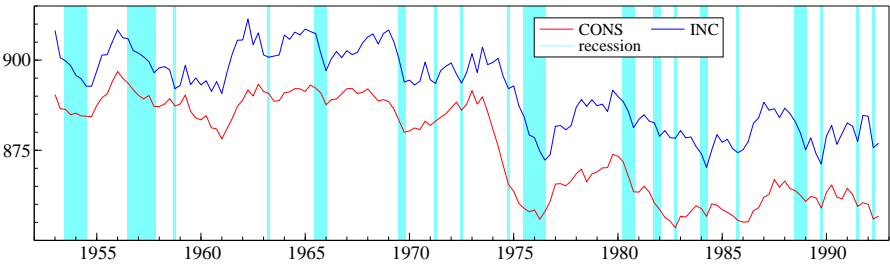


Figure 4.2 Time-series plot of CONS and INC with ‘recession’ shading

4.3 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

You can try to replicate **Figure 4.3** (make sure that Shading is changed back to Line). The first graph shows that graphing INFLAT and OUTPUT together is not very informative. To maximize the visual correlation between them, we can match the mean and range of OUTPUT to that of INFLAT, as shown in the middle graph. This has the same effect of plotting each on their own scale, also see §5.13.

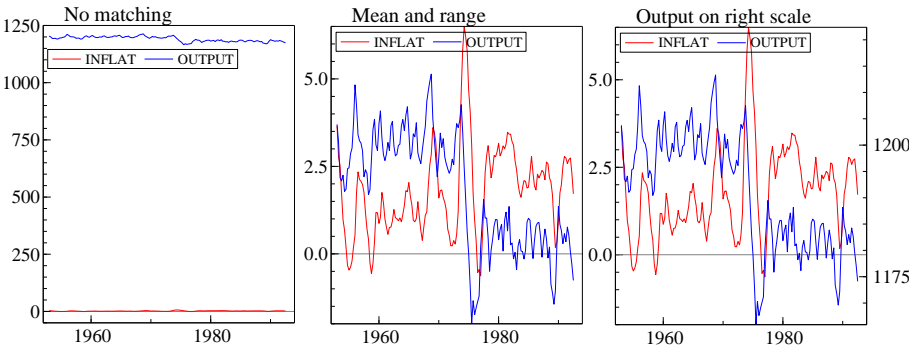


Figure 4.3 Time-series plot of INFLAT and OUTPUT: maximizing visual correlation

4.4 Scatter plots

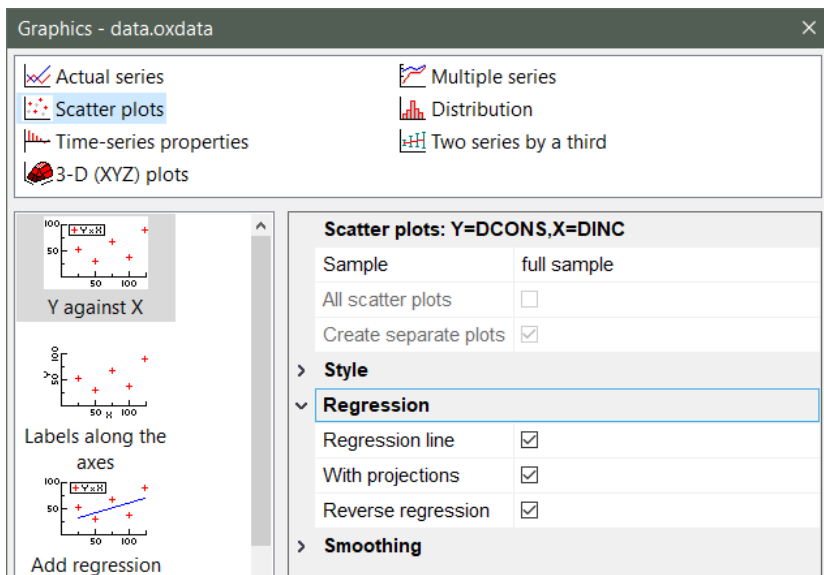
The available scatter-plot types are:

- Y against X
- Y against X, labels along the axes
- Scatter plot with regression line
- With cubic spline smooth, automatic bandwidth
- All scatter plots

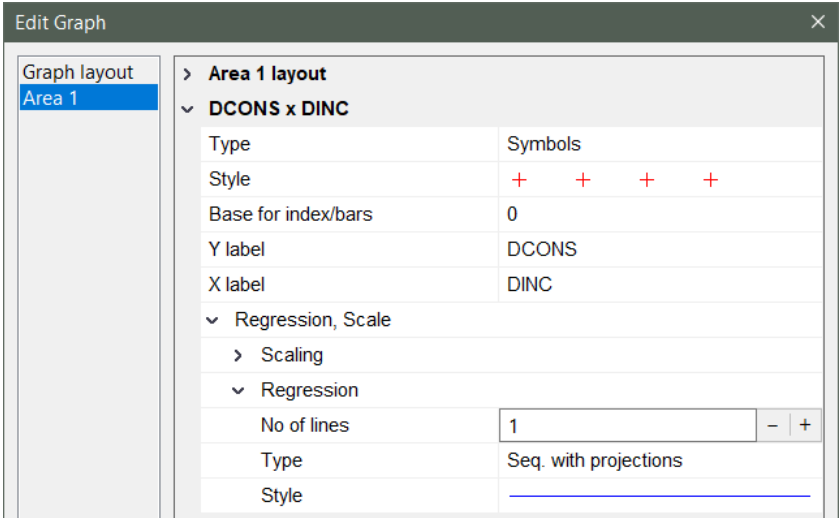
The Style, Regression and Smoothing sections allow additional options and combinations.

If you wish to keep the current graph as it is, click on File/New Data Plot Window. This will rename the current Data Plot window to Data Plot 1 (or the relevant next sequence number), and open a new Data Plot window. Otherwise OxMetrics will keep on adding to the existing window.

If DINC is not yet in your database, create it as the first difference of INC, using Algebra or the Calculator. Then click on the graph icon and select DCONS as the first (Y) and DINC as the second (X) variable, then on All plot types >, Scatter plots/Add regression line, expanding the Regression section:



Mark With projections, and Reverse regression, as shown in the previous capture. Click Finish to see the graph, shown in Figure 4.4a (we changed the symbols from the default plus symbol to a box). It shows the lines of best fit, minimizing vertical squared deviations of points from the line for the normal regression, and horizontal for the reverse regression. The 'steeper' line corresponds to horizontal minimization: can you see why? To check, double click on the graph and expand the Regression, Scale sections:



Remove the regression by setting the number of lines to zero. This just leaves the reverse regression line. Click on Undo to restore the normal regression line, as in Figure 4.4a below.

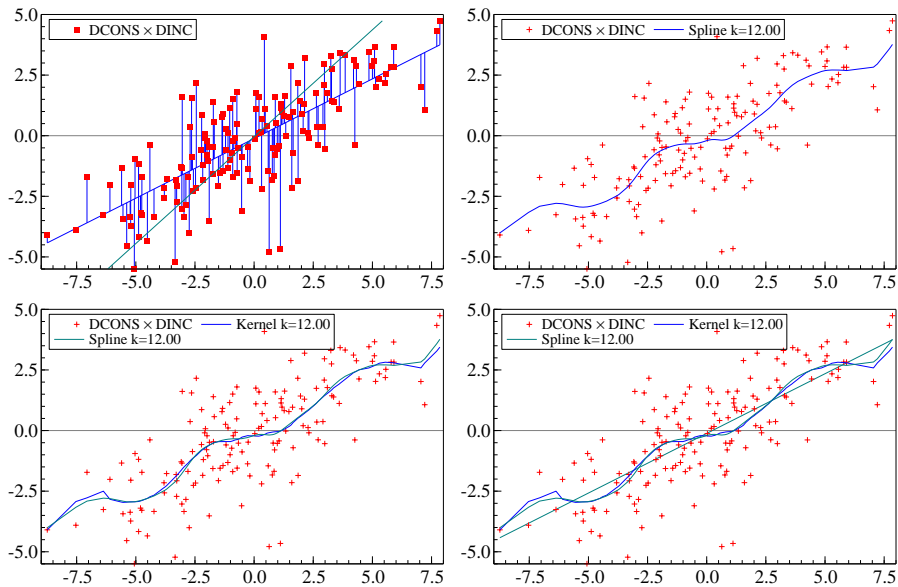


Figure 4.4 Sophisticated scatter plotting graphs

Return to the Scatter plot dialog, keep the selection of DCONS as Y and DINC as X, but now select Add smoothed line. Deselect the regression options, and expand the Smoothing section:

Scatter plots: Y=DCONS,X=DINC	
Sample	full sample
All scatter plots	<input type="checkbox"/>
Create separate plots	<input checked="" type="checkbox"/>
> Style	
> Regression	
√ Smoothing	
Cubic spline	<input checked="" type="checkbox"/>
Kernel	<input type="checkbox"/>
Bandwidth method	Automatic
No of parameters	12
Bandwidth	1600
EWMA	<input type="checkbox"/>
level alpha	0.2
level beta	0

Set the Bandwidth method to Parameters and create the plot as in [Figure 4.4b](#).

The cubic spline is one of two ways of fitting ‘smoothed’ lines that nevertheless track the general movements in scatter plots; the other is Kernel smooth – so draw another plot with both the kernel and spline smooths ([Figure 4.4b](#)).

These are actually quite close to a straight line. You can fit a straight line as well for comparison, or be more enterprising as follows. Click on the scatter plot with both kernel and spline lines (then shown by the ‘fuzzy’ outline) and copy it to the clipboard (two pages icon); next, click outside any graph area ensure that there is no selection, then paste to make a block of four graphs. Note that if an area is selected when pasting, the paste will be added to the selected graph. Then add a regression line to this graph. The direct comparison is clear: they are close approximations. And for all their ‘wiggles’, both are nearly straight lines. Setting the number of parameters to (say) 3 will produce a fairly smooth fit.

All scatter plots involving the selected variables, each in a separate graph. If CONS, INC, INFLAT and OUTPUT are marked, for example, the scatter plots are: CONS on INC, on INFLAT and on OUTPUT; INC on INFLAT and OUTPUT; and finally INFLAT on OUTPUT. If there are missing values, these are omitted from the graphs.

4.5 Distribution

- 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

4.5.1 Density estimation: Histogram and density

The histogram is a simple graph: the range of x_t is divided into intervals, and the number of observations in each interval is counted. The height of each bar records the number of entries in that interval. In OxMetrics, these are divided by the total number of observations to show the relative frequency (use bar to keep the count). OxMetrics sets a default number of intervals dependent on the sample size, but this can be changed by the user by clicking on Smoothing and Use custom bar count to set a different value.

The bimodal curved line in [Figure 4.1](#) is a smoothed version of the histogram. Because we divided the histogram by T , the heights of the bars add up to unity. Correspondingly, the area underneath the smoothed histogram is unity, and the curve is called a (non-parametric) density estimate. The familiar ‘bell’ shape of the normal distribution is added for comparison. It is clear that CONS does not look very normal. DCONS is much closer to a normal distribution: at least it is symmetric, but could still be too narrow, or too wide (i.e., have excess kurtosis relative to the normal).

Tip If you want to test for normality, use Data/Descriptive Statistics in PcGive and select the Normality option. Or regress on a constant in PcGive, and get the residual tests.

4.5.2 Distribution

Start with an empty Data Plot window, and return to Graphics, select CONS and DCONS, and use the Distribution option, this time marking Distribution, Frequency and Cumulative frequencies, and Box plot:

Distribution	
Sample	full sample
▼ Density estimation	
Histogram	<input type="checkbox"/>
Density	<input type="checkbox"/>
Normal reference	<input type="checkbox"/>
Distribution	<input checked="" type="checkbox"/>
> Smoothing	
▼ Frequencies	
Frequencies	<input checked="" type="checkbox"/>
Cumulative frequencies	<input checked="" type="checkbox"/>
> Bars	
> QQ plot	
▼ Box plot	
Box plot	<input checked="" type="checkbox"/>

We obtain eight graphs for CONS and DCONS, as shown in [Figure 4.5](#).

[Figure 4.5a](#) shows the cumulative distribution of the variable, integrating the estimated density. The result is presented in the form of a QQ plot against the normal

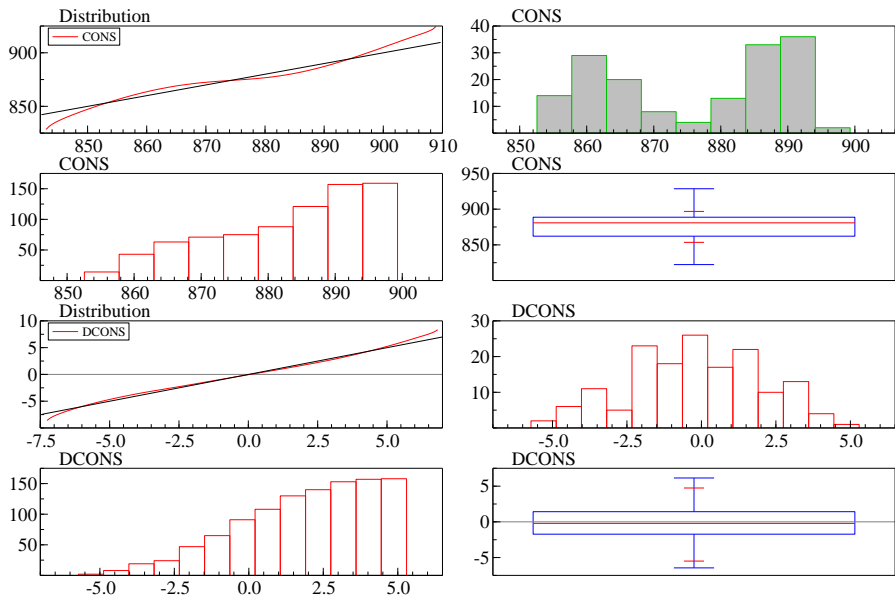
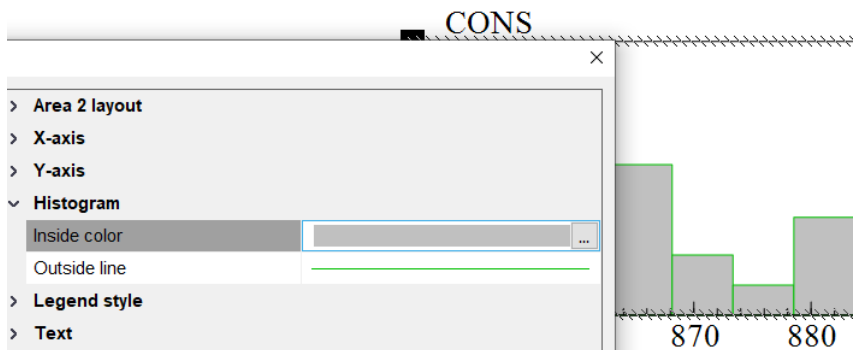


Figure 4.5 Further descriptive graphics of CONS and DCONS

distribution, see §4.7.

If you double click on a graph with a histogram, you can fill the histogram bars in any desired colour. By default, the inside colour is white, here we made it grey:



4.5.3 Frequencies

Figure 4.5b,f are just unscaled histograms, and the cumulative versions (Figure 4.5c,g) literally add up the total numbers below or inside the current interval (CONS has 159 observations, DCONS 158).

4.5.4 Box plot

A box plot shows the distribution of a variable in a more condensed form. Refer to [Figure 4.6](#), which shows the total range of a variable and the concentration in a central region, using the quartiles of the distribution, and the inter-quartile range (IQR).

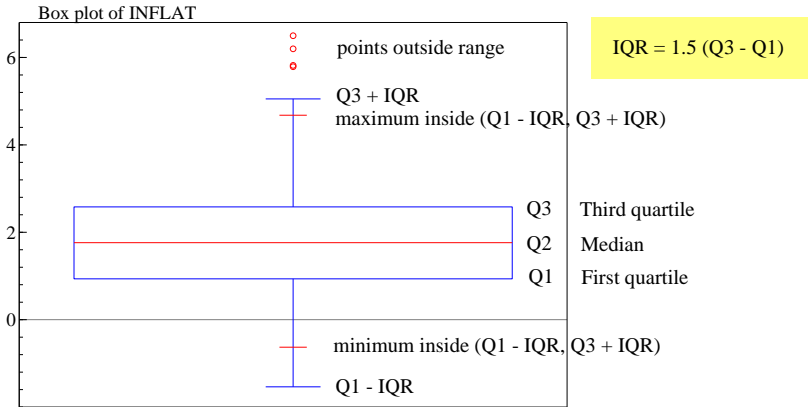


Figure 4.6 Box plot of INFLAT

4.6 Time-series: ACF etc.

- Autocorrelation function or correlogram
- Partial autocorrelation function
- Cross-correlation function
- Spectral density
- Periodogram
- Seasonal sub-plot

In a seasonal sub-plot, the data are displayed by season. For example, for quarterly data, first the quarter 1 data are graphed, then quarter 2, etc. This helps detecting changing seasonal patterns. Examples are in [Figure 4.12](#).

4.6.1 Autocorrelation function or correlogram

The sample autocorrelation function (ACF), or covariogram, plots the correlations \hat{r}_j between x_t and successive x_{t-j} , in [Figure 4.1](#) for $j = 1, \dots, 12$. Also see [§8.3](#). The length of the ACF can be set by the user. Since the correlation between x_t and x_t is always unity, it is not drawn in the graphs. We see that CONS remains strongly (positively) correlated with its own lags, even after 12 periods (3 years). This suggests some form of ‘long memory’ or non-stationarity. DCONS on the other hand shows little autocorrelation.

Tip If you want to know the exact values of the ACF, you can use Algebra code as described in [§10.5.3](#). You can also regress DCONS on a constant in PcGive, and use Test/Residual autocorrelations and Portmanteau statistic.

The ACF (or covariogram) uses the full sample mean and variance to compute the autocorrelations. This is the version presented in most textbooks. The correlogram uses the running mean and variance instead, see §8.5. As argued in Nielsen (2006), the correlogram provides a better discrimination between stationary and non-stationary variables.

To plot the correlogram (or the PACF derived from the correlogram), mark Using correlogram in the dialog. Figure 4.7 compares the two versions.

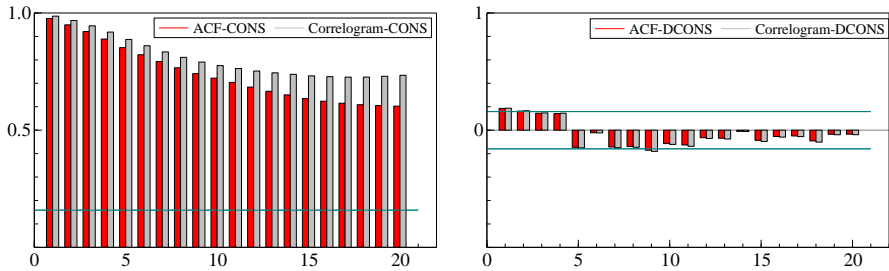


Figure 4.7 Comparison of correlogram and ACF for CONS and DCONS

4.6.2 Partial autocorrelation function (PACF)

The partial autocorrelation coefficients correct the autocorrelation for the effects of previous lags. So the first partial autocorrelation coefficient equals the first normal autocorrelation coefficient. The sample PACF of CONS shows that after the first, the remaining partial autocorrelations are close to zero.

4.6.3 Cross-correlation function

The sample cross-correlation function (CCF) graphs the correlation between a series and the lags of another series. Figure 4.8 shows that the CCF between CONS and DCONS is different between that of DCONS and CONS.

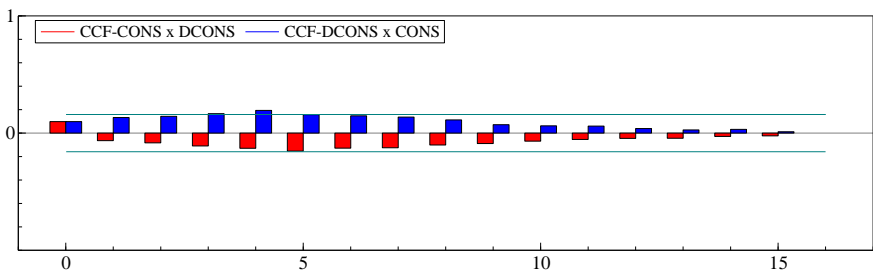


Figure 4.8 Cross-correlogram of CONS and DCONS

4.6.4 Spectrum and periodogram

The periodogram and the spectrum (or more accurately here: spectral density) graph the series in the frequency domain. The sample periodogram is based on the coefficients of the Fourier decomposition of the sample autocorrelations (that is, the correlations $\{\hat{r}_j\}$ in §4.6.1 between x_t and x_{t-j}). Figure 4.9 graphs the periodograms of CONS and DCONS using a lag length of 20.

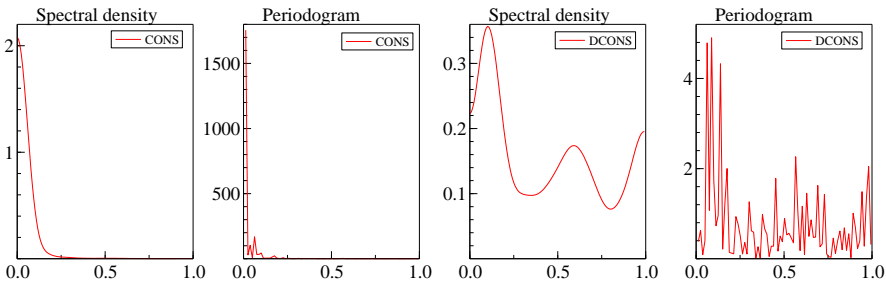
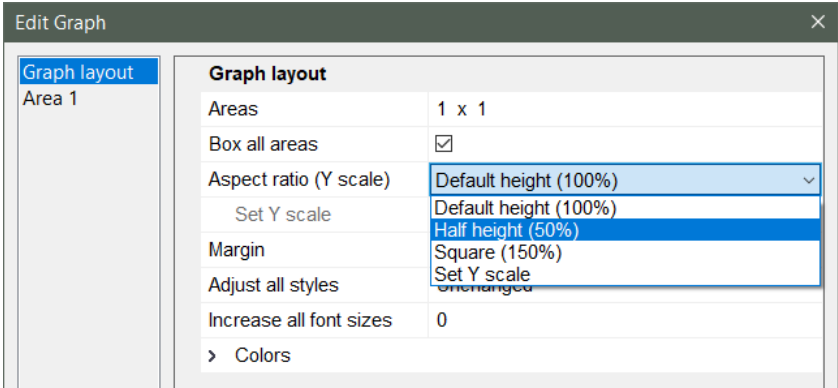


Figure 4.9 Frequency domain graphics of CONS and DCONS

The sample spectral density is a smoothed (and scaled) function of the periodogram. It is symmetric between $-\pi$ and π , and so is only graphed for $[0, \pi]$; 1 on the horizontal axis stands for π , 0.5 for 0.5π , etc. Peaks at certain frequencies can indicate regular (cyclical or seasonal) behaviour in the series. The spectrum of CONS shows the ‘typical spectral shape’ (a term introduced by Granger, 1966), which many macroeconomic variables appear to have: a pronounced peak at the lowest frequencies. Like the correlogram, this indicates that successive values of CONS are strongly correlated. Figure 4.9 shows that spectrum of DCONS is much ‘flatter’, indicating a series which is closer to white noise. More information is in §8.8.

Tip You may wonder how graphs such as Figure 4.8 are made smaller. In this case, that is done using Graph layout/Aspect ratio (Y scale)/Half height (50%):



4.7 QQ plots

The next graph type to be considered in this chapter concern QQ (quantile-quantile), or cross-probability, plots:

- Quantile plot: against uniform
- QQ plot against normal (same mean, variance)
- QQ plot with choice of distribution

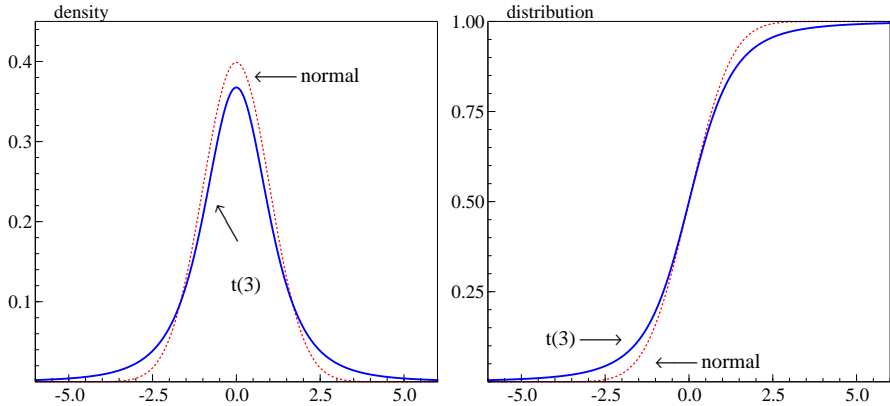


Figure 4.10 Densities and distributions of normal and t

As discussed in the companion book on PcGive, statisticians view variables as being described by probability distributions. If X is the variable, and x a value it could take, then

$$P_x(X > x)$$

is the probability that the value is in fact greater than x . For example, if the variable is the height of a child, when x is one metre, $P_x(X > 1)$ is the probability that the child is taller than a metre. The values of x can be any number, but $P_x(\cdot)$ cannot be negative or exceed unity. Plotting $P_x(X > x)$ against x generates an \int -shaped curve (but more stretched out horizontally), which is hard to interpret. **Figure 4.10** shows a normal and a t density together with their cumulative distributions.

When X has a uniform distribution over $(0, 1)$, if $P_x(\cdot)$ is plotted against x in a unit square, the result is a straight line. A similar idea applies to all distributions and QQ plots can be selected so some reference distribution is a straight line with the empirical distribution compared to it.

The QQ plot options page on the Graphics dialog allows for a choice of reference distributions. For the t, F and χ^2 distribution it is necessary to also supply the degrees of freedom arguments. Drawing a variable against a uniform results in a so-called quantile plot.

Figure 4.11 illustrates for CONS and INC and their differences against the normal as a reference. The normal QQ plots include a 95% pointwise confidence interval, see §8.11. Clearly, there is a vast range of graphics options to suit your needs.

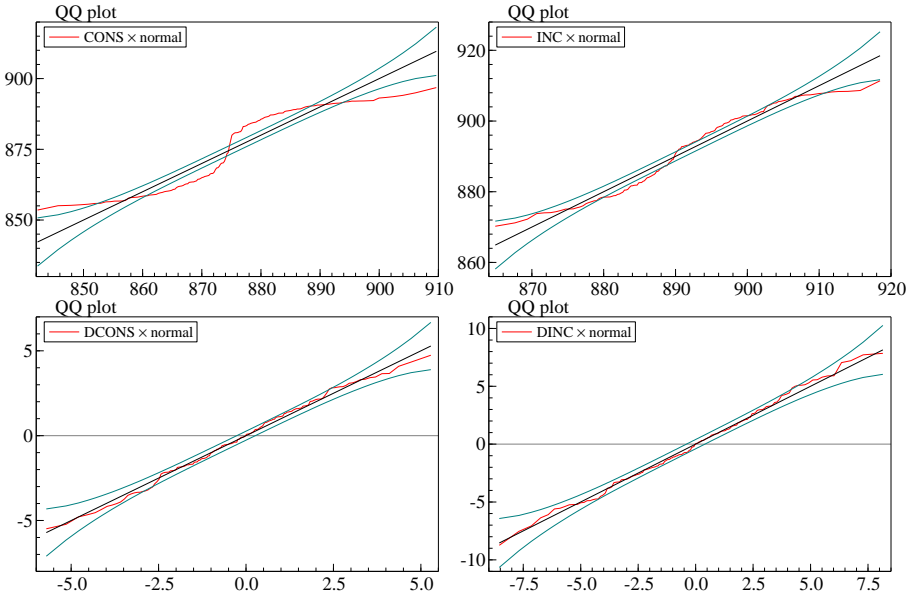


Figure 4.11 QQ plots

4.8 Two series by a third

The following types of graphs can be created as ‘two series by a third’:

- Error bars
- Error bands
- Error fans
- High-low
- Show Z values
- As symbol size: Bubble chart

To illustrate 3-way cross plotting, we use the data on the levels and first differences of UK log consumption and disposable income from [Davidson, Hendry, Srba, and Yeo \(1978\)](#) (the data set is supplied with OxMetrics, and will be in the data folder).

Start by loading the DHSY data set. Create create DLC and DLY as the first difference of LC, LY in Algebra (or the Calculator). Also create a seasonal indicator for the first quarter, and variable quarter that holds the quarter (values one to four):

```
Q1 = seasonal(0);
Quarter = period();
```

Activate the Graphics dialog, selecting LC as first, LY as second, and Quarter as third variable. Choose as graph type Two series by a third/Show values:

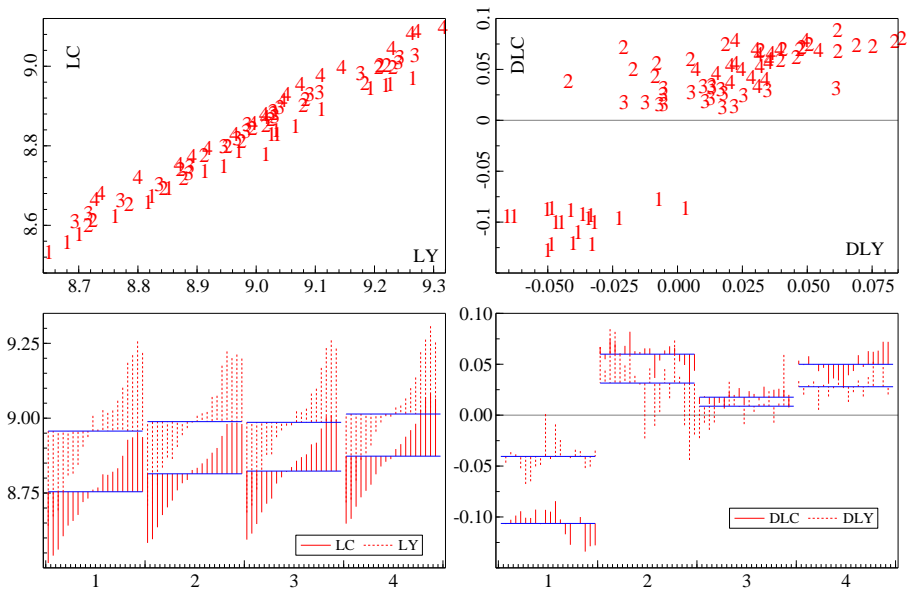
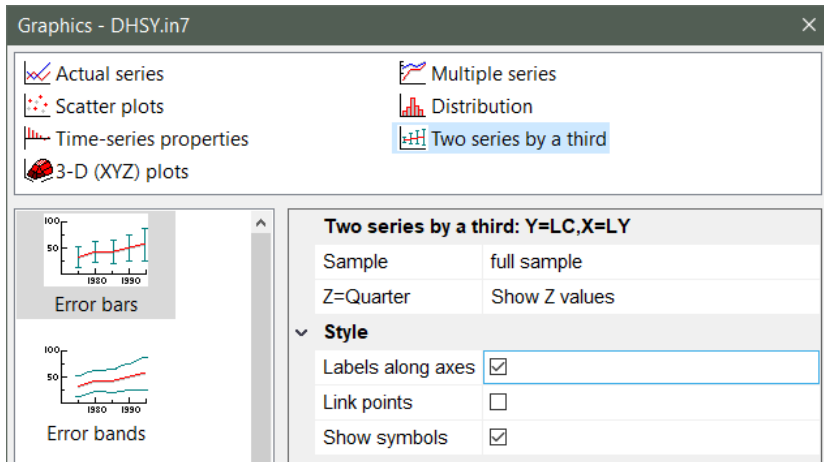


Figure 4.12 Cross plotting consumption against income by seasons (DHSY)

The result is in panel a of **Figure 4.12**, with a similar graph for the first differences in panel b. Note that we also choose to have the labels along the axes, instead of a legend. In the top-left panel (levels), the fourth (Christmas) quarter is uniformly highest, and the first lowest; in the lower panel (first differences), the first-quarter points are grouped far below all other quarters, the second are scattered along the X-axis, and the last two quarters are bunched together. Thus, plotting reveals distinct patterns in this instance. The remaining panels show the seasonal sub-plots, which essentially reveal the same information.

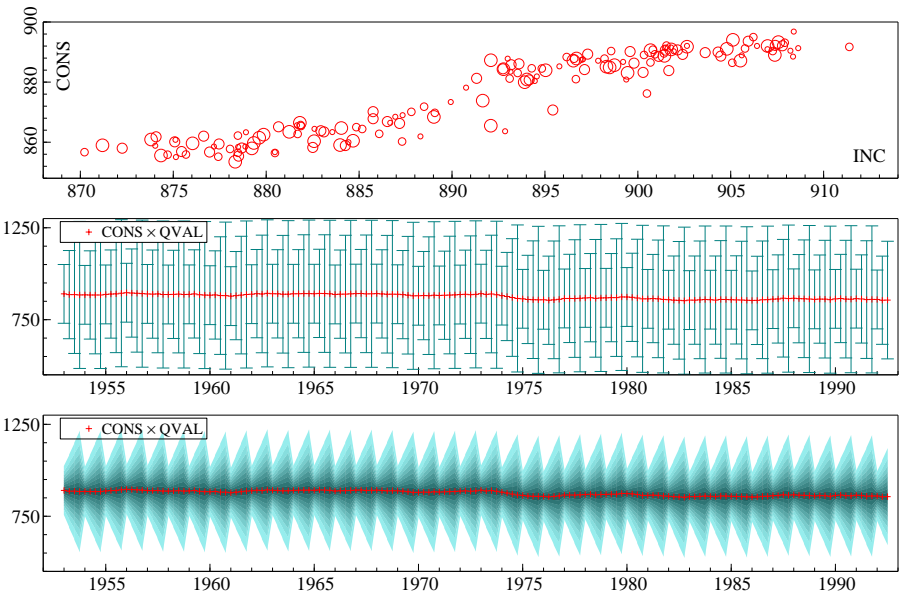
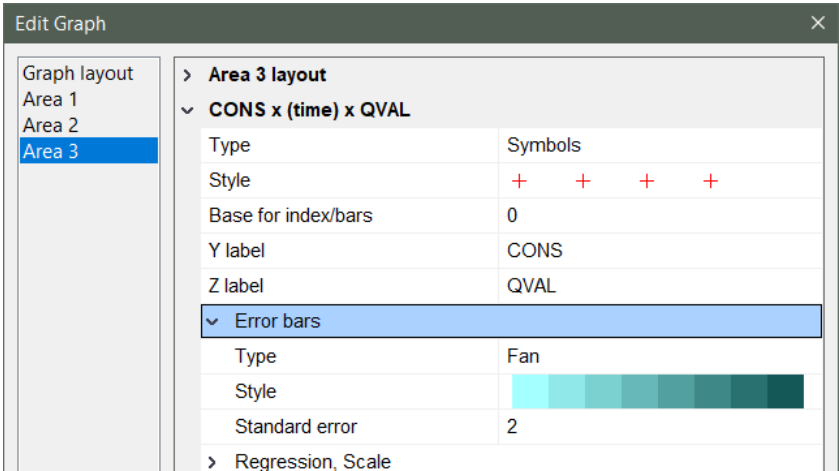


Figure 4.13 Bubble chart and error bars

There are more cross-plot options. One is a so-called bubble chart. It consists of a scatter plot, where the symbols are circles, and the size of the circle indicates a third dimension (e.g. market share).

Return to the data.oxdata tutorial data set, and use Algebra to create `QVAL=period()*40+40`. Create a Scatter Plot of CONS on INC by QVAL. Select Two series by a third/Bubble chart. Double click on the graph to set the symbol to a circle. The result is in Figure 4.13a. The final graph of this section illustrates error bars/bands. Select CONS as the first variable, and QVAL as the second, and choose Two series by a third/Error bars at the second stage. When only two variables are selected, the horizontal (X) axis is set to 'time'. See Figure 4.13b.



Afterwards, the colour and size of the error bars can be changed in the section related to the error bars, see the capture on the next page. It is also possible to switch between bars and bands. For example, changing to error fans as shown in [Figure 4.13c](#).

4.9 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

With 3-dimensional plots there are two ways of presenting the data: as a scatter of points in three-dimensional space, or as a table with the X and Y values along the side, and the Z values in the cells. The tabular format specifies the 3D surface directly, whereas for the scatter it is left to OxMetrics to work out the surface. Although the tabular format leads to better results, the scatter is easier to use, as it requires only three variables.

4.9.1 Surface from scatter

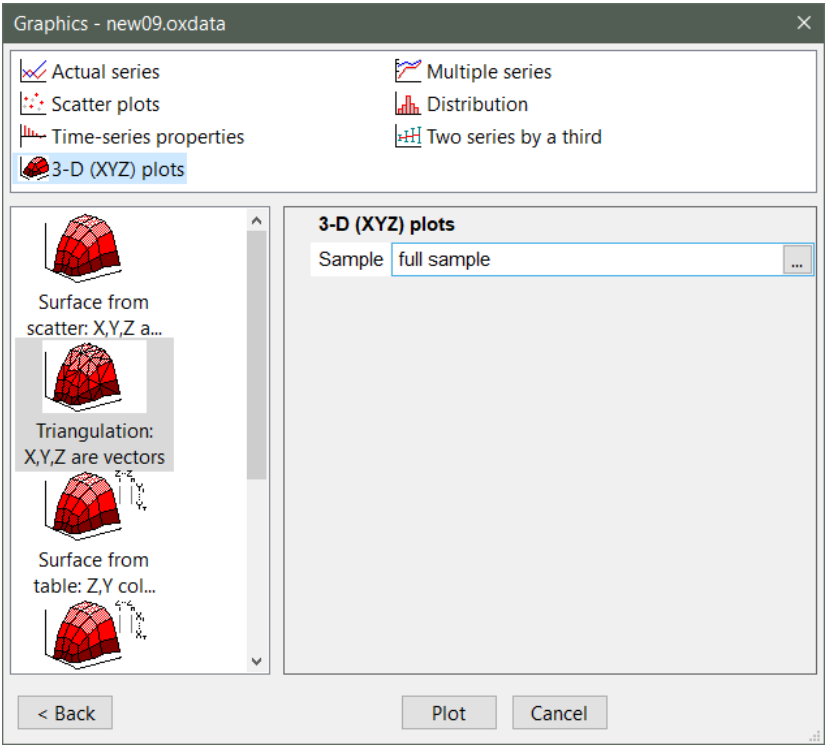
To create a surface from a scatter, OxMetrics derives a triangulation, and from the triangulation a smooth surface. This method works best when the surface is smooth. To illustrate this, create a new database with 1000 observations. Create a random X and Y value, uniformly distributed between -3 and 3 . Create Z as the independent normal density (without the normalizing constant):

```
X = 6 * (ranu() - 0.5);
Y = 6 * (ranu() - 0.5);
Z = exp(-0.5*(X^2+Y^2));
Z1 = X^2 - Y^2;
Z2 = fabs(X) - fabs(Y);
```

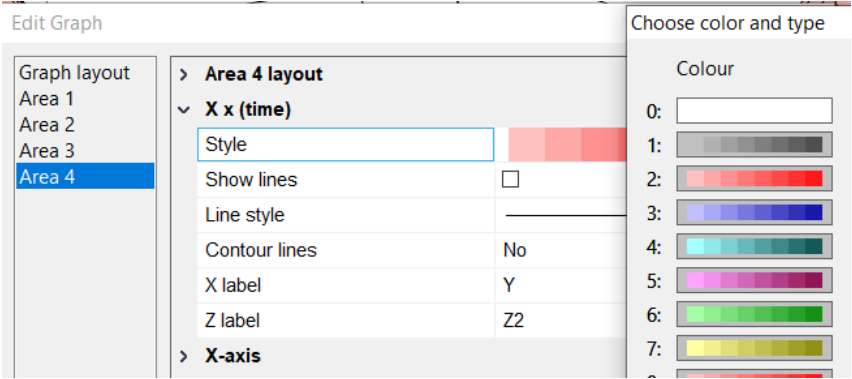
The first five observations for graphing are:

X	Y	Z
.386658	-1.05248	.533332
1.61961	-2.49774	.0119035
-.501558	-.032988	.881329
-2.0472	-1.57673	.0354911
-2.41075	2.07573	.0063444

Select Y , X and Z in the graphics dialog (in that order), select Triangulation: X, Y, Z are vectors.



Repeat this, but selecting Surface from scatter: The result can be seen in the first two panels of Figure 4.14. The third graph is a surface of Y,X,Z1, and the fourth of Y,X,Z2. The last graph has it colour removed, so that it appears as a wire diagram.



There are a few additional edit actions available for 3-dimensional graphs:

- Rotation

The Edit menu has an entry for rotation: left/right (azimuth, keyboard short-cuts are left-arrow, right-arrow), up/down (elevation, keyboard short-cuts are up-arrow, down-arrow) and side ways (twist). This can be done from the menu for the currently selected 3-d area, or for all graphs. The keyboard short-cuts only work when a 3D area is selected.

- Palette

The Style entry for the 3D graph on Edit Graphs allows for a choice of palette. For example, Figure 4.10d is shown with a wireframe.

- Contour lines

The same page allows for the addition of contour lines to a surface (but not a triangulation).

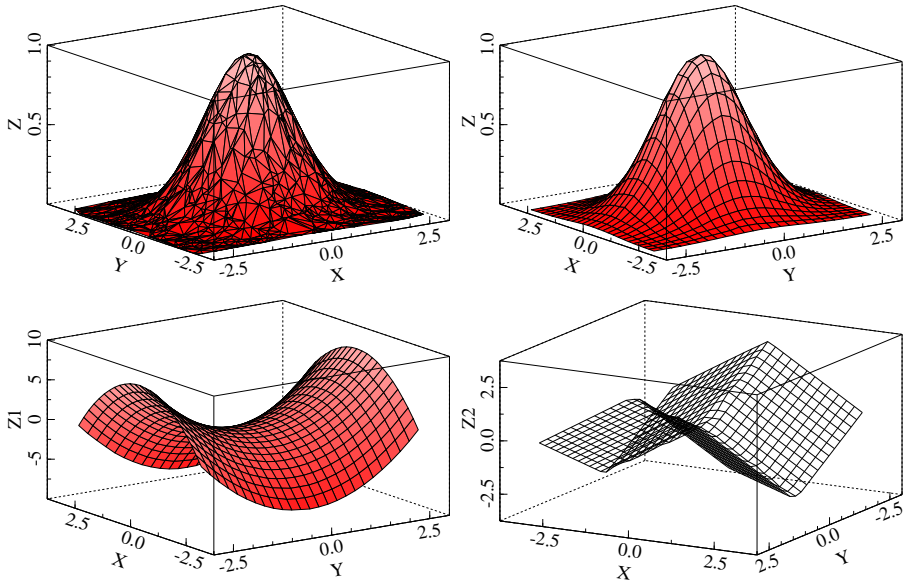


Figure 4.14 3-dimensional graphs: bivariate normal and saddles

4.9.2 Surface from table

Tables are harder to work with, because OxMetrics essentially only recognizes vectors, not matrices (unlike Ox).

Create a new database with only seven observations. Let X be $-3, -2, \dots, 3$, and Y the same. The Algebra code then defines $Z1 = f(X_1, Y)$, $Z2 = f(X_2, Y)$, etc.:

```
X = trend() - 4;
Y = X;
Z1 = exp(-0.5*(X[1(1)]^2+Y^2));
Z2 = exp(-0.5*(X[2(1)]^2+Y^2));
Z3 = exp(-0.5*(X[3(1)]^2+Y^2));
Z4 = exp(-0.5*(X[4(1)]^2+Y^2));
Z5 = exp(-0.5*(X[5(1)]^2+Y^2));
Z6 = exp(-0.5*(X[6(1)]^2+Y^2));
Z7 = exp(-0.5*(X[7(1)]^2+Y^2));
```

Note the indexing of the form `year[period]`, which is explained in §10.3.7. The

dataset looks like:

<i>X</i>	<i>Y</i>	<i>Z1</i>	<i>Z2</i>	...	<i>Z7</i>
-3	-3	0.00012341	0.00150344	...	0.00012341
-2	-2	0.00150344	0.0183156	...	0.00150344
-1	-1	0.00673795	0.082085	...	0.00673795
0	0	0.011109	0.135335	...	0.011109
1	1	0.00673795	0.082085	...	0.00673795
2	2	0.00150344	0.018315	...	0.00150344
3	3	0.00012341	0.00150344	...	0.00012341

Select *X*, *Y*, *Z1*, ... *Z7*, and 3-dimensional plots/Surface from table (because of the symmetry '*Z*,*Y* columns match' and '*Z*,*X* columns match' will give the same result), to see [Figure 4.15a](#).

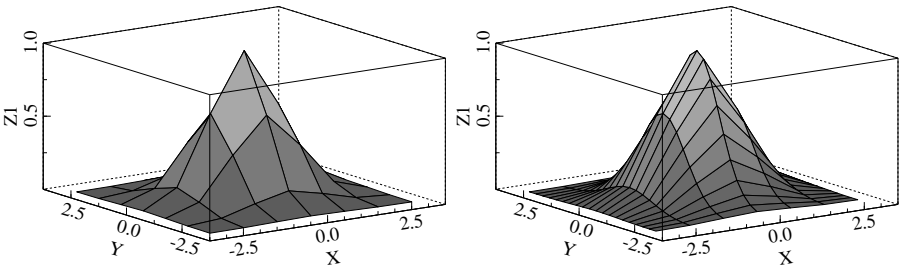


Figure 4.15 3-dimensional graphs from tabular data

The difference in storage can be illustrated by increasing the number of points in *Y*, while keeping *X* fixed. To do this, create a new database with 20 observations. Paste *X* from the previous database, so *X* is -3, -2, ..., 3, with missing values for the remaining observations. Create *Y* as:

```
Y = (trend()/20 - 0.5) * 6;
```

and *Z* as before. Select *X*, *Y*, *Z1*, ... *Z7*, and 3-dimensional plots/Surface from table: *Z*,*Y* columns match to see [Figure 4.15b](#).

4.10 Conclusions

Many different types of plots are available. New ones can be derived by editing or copying and pasting, as discussed in the next chapter.

For a quick example of the plotting capabilities, open the batch file `plots.fl` (in the `0xMetrics9\batch` folder) and run it.

Chapter 5

Tutorial on Graph Editing

The prologue introduced the basics of using OxMetrics, accessing its menus and dialogs, loading, saving and transforming data, using graphics, including saving, printing and pasting them, and described the Calculator and Algebra. [Chapter 4](#) gave an overview of the many different types of graphs. Now we consider the facilities for editing graphs in more detail. [Chapter 12](#) discusses various graphics options more systematically.

If you're not inside OxMetrics at the moment, restart and load the tutorial data set `data.oxdata`.

5.1 Multiple graphs

Create DCONS and DINC as the first differences of CONS and INC, then activate the Graphics choice on tools which defaults to the Variables options as shown in [Chapter 2](#). Select CONS and INC, then click Actual series, at which point the graph will appear; restart the graphics dialog, mark and select CONS and INC again, but now click on the YX (Scatter plot) button, then Apply; and repeat both of these operations for the pairs DCONS and DINC, clicking OK after the fourth, to see [Figure 5.1](#). We number the four graphs from left to right and top to bottom as:

$$\begin{array}{cc} a & b \\ c & d \end{array}$$

5.2 Graphics paper: areas and coordinates

To understand the editing facilities, it is easiest to see the graphics window as a piece of graphics paper, consisting of 15 000 'pixels' along the x -axis, and 10 000 along the y -axis (this is very much higher than the actual screen resolution). The four graphs in [Figure 5.1](#) all have an *area* attached to them. By default, these areas are evenly spread

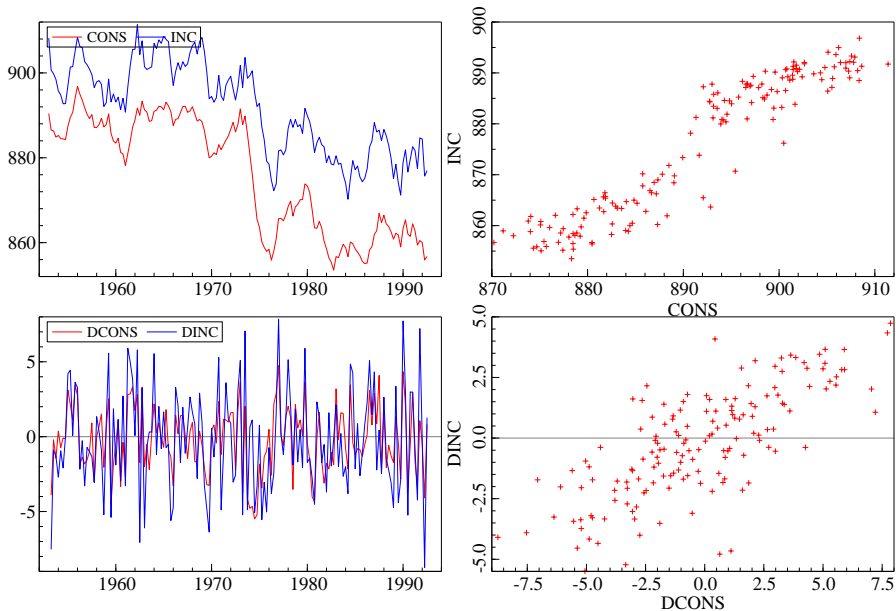


Figure 5.1 Descriptive graphics of CONS, INC, DCONS and DINC

out on the paper. An area has pixel coordinates to fix it on the paper, and real world coordinates to draw variables, axes, etc. inside the area.

Edit Graph

Graph layout

Area 1

Area 2

Area 3

Area 4

Area 1 layout

Boxed

> World coordinates

> Pixel coordinates

Set

☒

Left

680

Right

7060

Bottom

5363

Top

9403

> CONS x (time)

Type

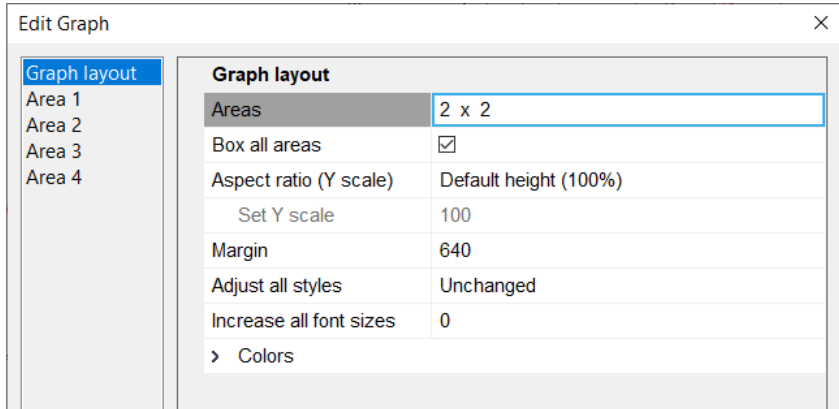
Line

Style

An area can be selected with the mouse by clicking on it. Then it is shown selected (in a 'fuzzy' rectangle), and you can pick it up and move it around on the paper, or resize it. Once moved around, the pixel coordinates switch from automatic to manual. To see this, click on the first area to select it, then move it with the mouse. Next, use a double click on the area (or use Edit/Edit Graph) to activate the Edit Graphics dialog and select the Area Layout page for Area 1. Expand the Pixel coordinates section, and uncheck Set to reset to the default paper location.

Similarly, it is possible to redefine the world coordinates of the area (e.g. to zoom in on a part of the graph). This is done by deselecting the Set check box in the World coordinates section. To apply the new settings to all areas, use the Copy properties to other areas section on the page.

The Graph Layout entry in the left-hand column gives access to the properties that apply to the whole graph. For example, there is a drop-down box that allows for changing the area ‘matrix’. With four graphs on paper, various ways of layout can be chosen. Select Box all areas to get the graphs boxed as in [Figure 5.1](#).



5.3 Graphics view

When resizing a Graphics window, the content automatically grows or shrinks accordingly. However the aspect ratio of the ‘graph paper’ (the size along the x -axis relative to that along the y -axis) remains constant. To change this, see [§12.7.1](#).

5.4 New Data Plot Window

The default graphics window is called ‘Data Plot’. By default, OxMetrics keeps on adding graphs to this window. If you wish to keep the current window, and start with a clean piece of paper, use New Data Plot Window on the File menu.

To restart with a clean sheet: close the graphics window; a new one will be opened automatically when required.

5.5 Copy and paste

The use of the clipboard works at two levels:

- an area can be copied into another area, or added to the page;
- a whole page can be copied to another program.

As an example of the first mode, select the last area, and copy it to the clipboard ('two pages' icon). To add it to the page, make sure that nothing is selected by clicking on the page outside any area. Now you can paste the graph which will appear as the fifth area.

Then select the third area, and paste again. Now the scatter plot of DINC and DCONS is added to the time plot. Since the axes are automatically adjusted to accommodate the new data, this rather messes up the graph.

The use of the clipboard is also discussed in §12.8.

5.6 About line colour and style

Lines in a graph can have different colours and styles (solid, dotted, dashed, etc.). This is implemented in two levels:

- There is a *palette* of 16 line types, defining colour, style and symbol type.
- Each line is defined through an index in that palette.

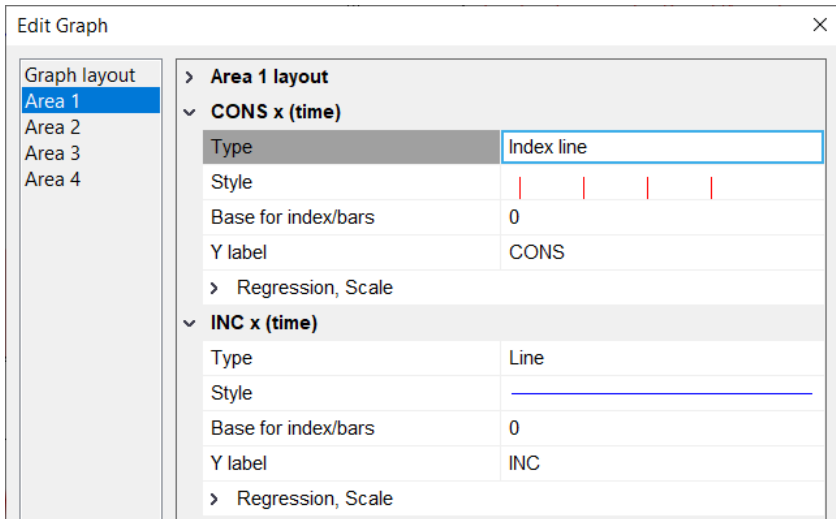
So, for example, when a line has type 3, it is drawn in whatever colour and style is defined for that type. When editing a graph on screen, it is possible to change its line type (see §5.7). In addition, and separately, the global palette settings can be modified (§5.8).

Line type 0 refers to the background colour (white), and type 1 to the foreground (normally black). Therefore, the first index used in drawing variables is 2.

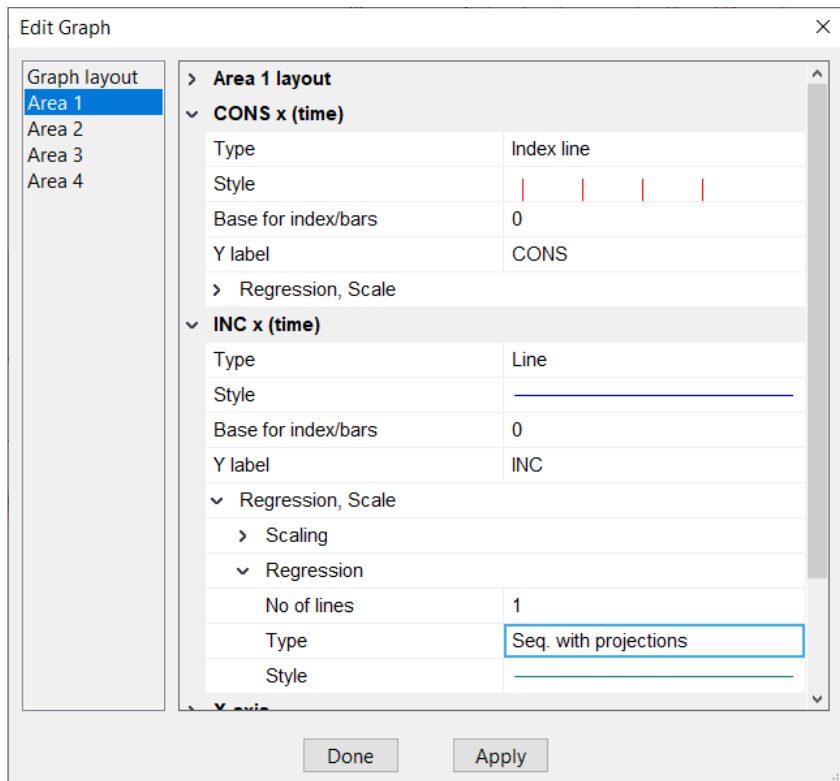
5.7 Editing graphs: Graphics properties

By default, all graphs use the automatic selections offered by OxMetrics: these are usually fine for immediate usage, but are open to many possible amendments depending on their purpose. We begin with Figure 5.1a. Double click on it to bring up the Edit Graphics dialog (a single click makes it become the focus, shown by the 'fuzzy' boxed outline mentioned before). Almost any aspect can be changed in the Edit Graphics dialog.

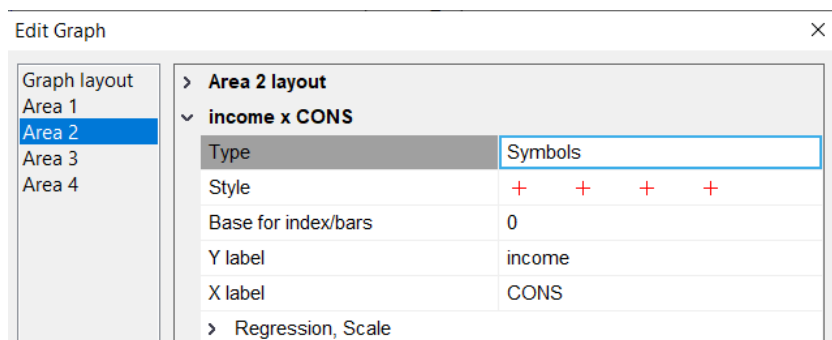
Lines are chosen from a set of predefined line types, which define colour, pattern, width and default symbol type. Model/Preferences/Graphics Setup defines the line definitions. Here we start by changing the line type to index line (as shown below in Figure 5.2 on page 68) after double clicking on the first area:



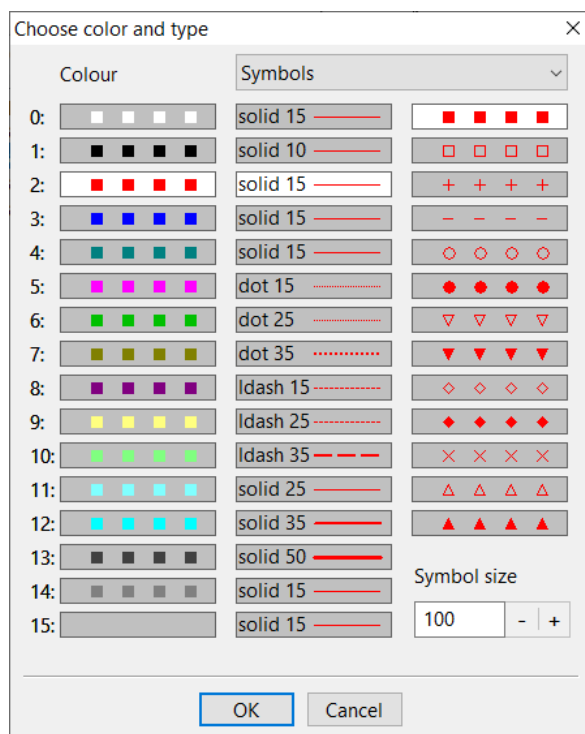
Next, select the INC \times (time) section, and under Regression, Scale, and set the number of regression line to one (as in §2.3.2), as well as Seq. with projections:



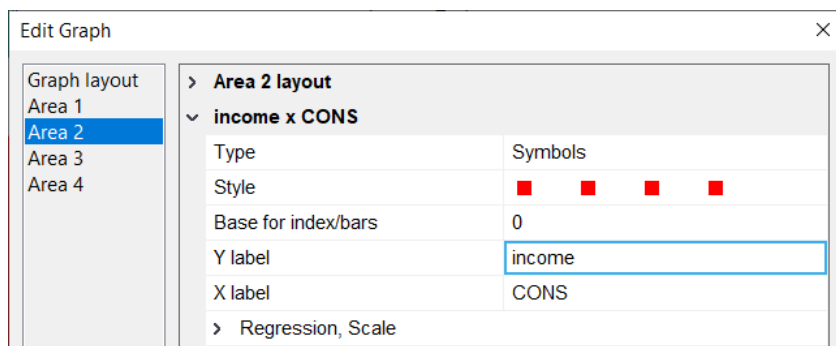
Selecting Area 2 (Figure 5.1b), change the Style:



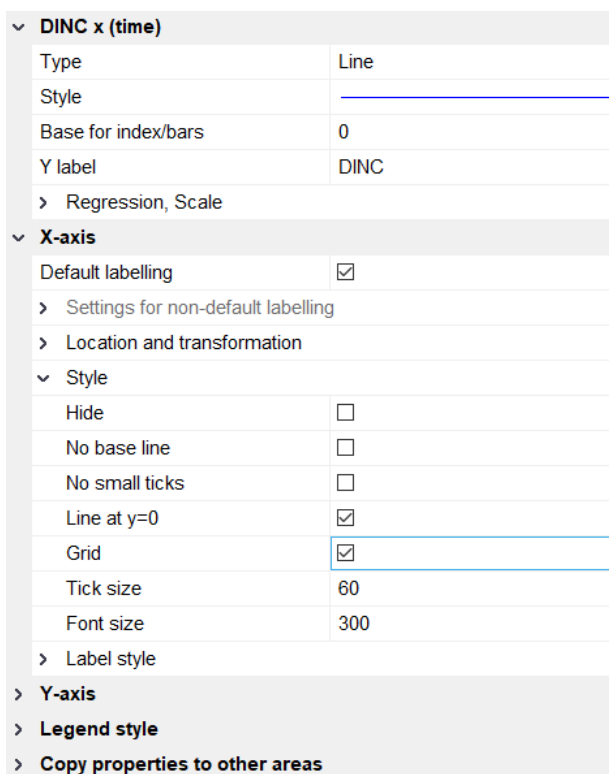
to a solid box, and alter its size to 100:



Click on Regression, and mark 4 lines, and Seq. with projections. This page also allows for renaming the labels of variables in graphs by clicking on Y or X label, in this case to income:



For graph c, select X-axis on Edit Graphics, and click Grid, as shown in the next capture. Repeat for the Y-axis, then close: on the graph, grab the legend and move it to a more convenient location (click with the mouse and holding the button, ‘drag’ it to the place required) as shown. Changing Style alters the line colour or style: we have done that for DCONS in c to a thicker line as explained in the next section.



Finally for graph d, mark 20 (sequential) regression lines. The final result is in [Figure 5.2](#). This is a very different ‘picture’ from our first figure, and such facilities can be used to reveal hidden features of data.

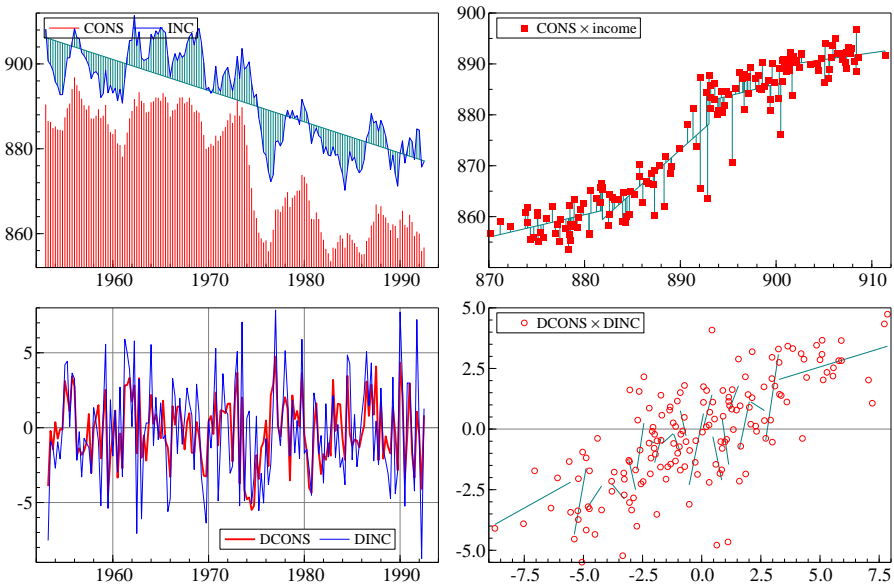


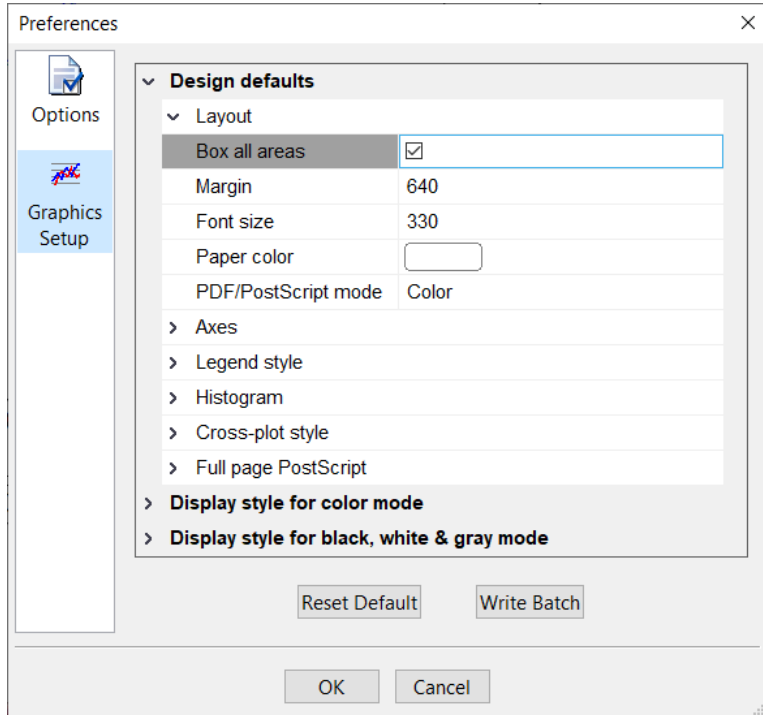
Figure 5.2 Edited and amended descriptive graph

5.8 Graphics setup

There are many default graphics settings that can be generically controlled by the user, and are persistent between runs of OxMetrics.

- Lines (Tools/Graphics Setup):
define the default line attributes;
- Layout:
define defaults for margins, boxing and font size;
- Axes:
define defaults for axes, grids, axis labels and grids;
- Legend style
- Histogram
- Cross-plot style
- Palettes (Tools/Graphics Setup):
for three-dimensional surfaces and error fans;
- Display style for color mode:
Line colours, types, symbol style, palettes, etc.
- Display style for black, white & gray mode:
Line colours, types, symbol style, palettes, etc.

Click on Model, Preferences, Graphics Setup to bring up the dialog shown below.



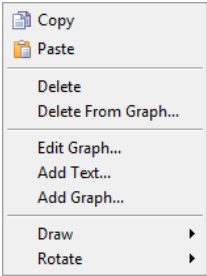
The colours, types and thickness of all lines can be set as wished: for example, much thicker lines are more useful for live demonstrations than printed graphs; some colours show much better than others, etc.

Any changes made via Preferences are permanent: they will still apply to subsequent runs of OxMetrics, or when displaying graphs from Ox. If only temporary changes are needed, or different settings for each graph, these can be implemented from the Edit Graphics dialog after double clicking on a graph. The operating principles are the same throughout the OxMetrics system, so it is easy to develop a setup as desired – and change it quickly if needed.

All these settings can also be controlled from the OxMetrics batch language. The easiest *modus operandi* is perhaps to design the layout using the dialogs shown in this section. Then use the Write batch command to record the batch code in the results window. This can then be pasted to the batch editor, and saved to a file. Or edited, and run directly from the results window. The batch command used to control the graphics settings is `setdraw`.

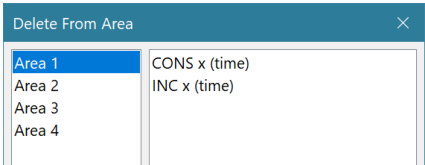
5.9 Adding to and removing from a graph

Close the current graphics windows, and create a scatter plot of INFLAT against OUTPUT. It is possible to add further graphs to this area. The default graphics command will always create new areas, but the Add Graph command from the Edit menu or the context menu:



adds to a selected area. You can try this by adding a scatter of INFLAT against INC to the graph. Similar results can be achieved by copying one area to the clipboard, selecting another, and pasting into that area.

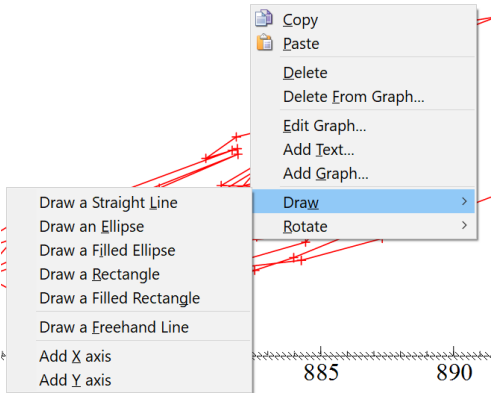
The simplest way to remove it again is to use Undo. An alternative is to delete the second scatter. Select Delete From Graph:



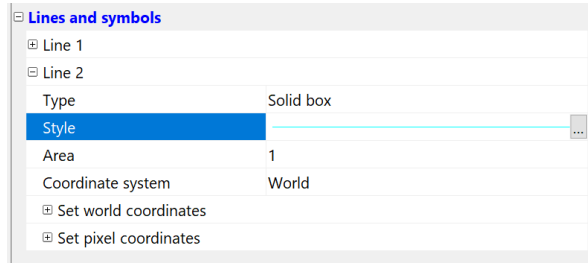
mark INFLAT x INC and press Delete.

5.10 Drawing

Change the line type of the INFLAT x OUTPUT scatter plot to symbol and line to link the points. Next, access Draw on the context menu (right click):



Use Draw a Filled Rectangle to highlight the two clusters on the graph. Click on Draw Symbol/Draw Filled Rectangle — the mouse point becomes like a pencil, as indeed it can now draw a rectangle. Put the mouse down near the top left of the first cluster. while holding the mouse down, continue the expansion till the cluster of points around the low OUTPUT ‘equilibrium’ is highlighted as in Figure 5.3a. Next double click on the rectangle and a Lines and symbols section will appear:



Use Style to change the colour to 12 (for example).

If you now add another box to highlight the second equilibrium we get the final graph in as [Figure 5.3b](#).

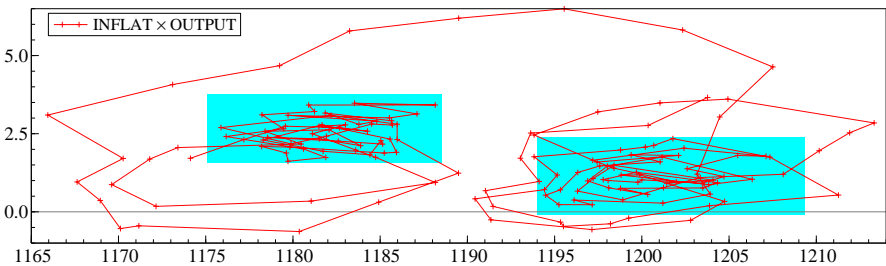
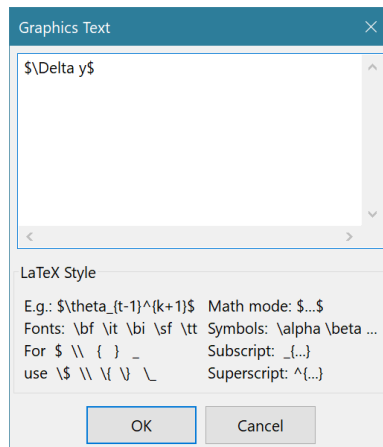


Figure 5.3 Drawing on an INFLAT/OUTPUT scatter plot

5.11 Adding text and variables

To add text, proceed as in the previous section: use the edit menu or click with the right button, and select Add text. The mouse cursor becomes a letter. Now click where you wish the text to appear. A dialog pops up in which the text is entered:



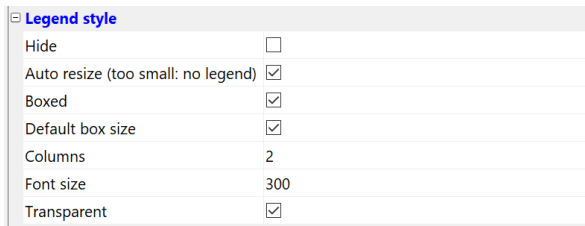
You may note from the dialog that graphics text can use a \LaTeX -style of formatting, more information is given in [§12.10](#).

Another simple way to add text is to start typing a letter, which will become the first letter in the dialog box. The text will be added at the location of the mouse (but can be picked up and moved). When adding text just above the graph, it becomes the title.

5.12 Legends

Legends are created for all the variables in each graph, located in the top left corner of the graph. The legend can be picked up with the mouse and moved around the graph. They are automatically resized as the graph gets smaller. When the graph is very small (usually with more than sixteen graphs in one window) the legend is omitted altogether.

Another way of removing the legend is by hiding it. The Legend style section on the Edit Graphics dialog allows hiding and unhiding the legend:



Legends are transparent by default. In that case, they are drawn below data plots, with the exception of shading and separately drawn symbols. Opaque legends, on the other hand, are drawn on top of everything else.

5.13 Scaling variables

Often it is useful to draw two or more variables in the same graph to allow easy comparison. However, when the variables have widely differing magnitude, all visual information will disappear. For example, graph CONS and OUTPUT (after closing the existing graph). They appear as two nearly straight lines, one at the top and one at the bottom of the graph: most visual information is lost.

Two methods are available to enhance the visual correlation:

- scale OUTPUT, adjusting it to the CONS values;
- draw them separately, and then put the areas on top of each other. (This can also be done directly, by choosing Multiple series/ Second series on right scale as graph style.)

The first method is easiest to implement: select Scaling under CONS × (time), in Edit Graphics, and set Match by to Means and ranges, as shown below.

CONS is the selected variable, and all other variables (OUTPUT here), will be adjusted to it. With Means and ranges, the mean of OUTPUT is changed to that of CONS, then the scale factor is set to give OUTPUT the same minimum and maximum:

$$\text{OUTPUT}^* = \frac{\text{OUTPUT} - 233.52}{1.093}.$$

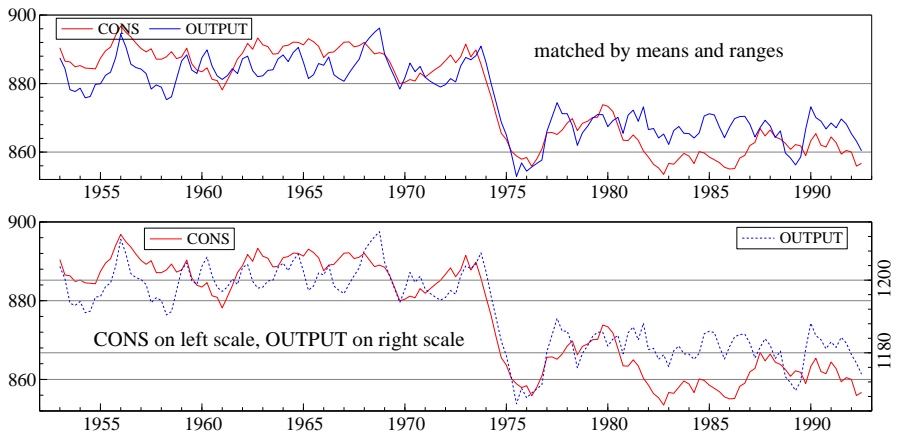
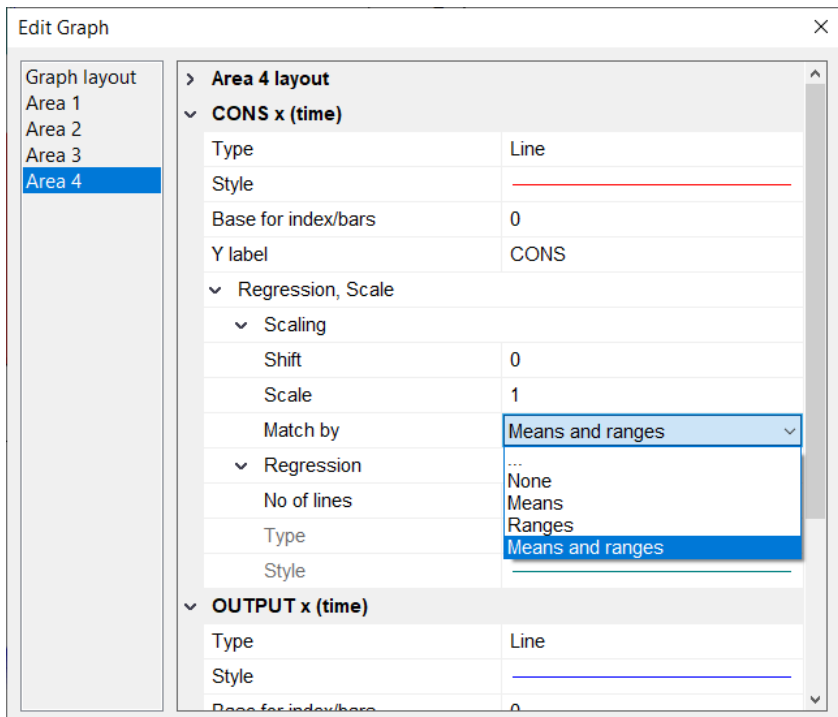
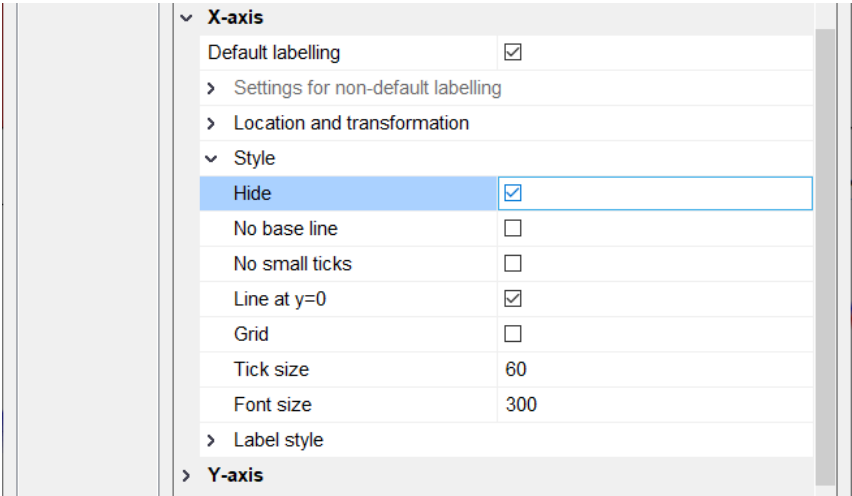


Figure 5.4 CONS and OUTPUT with matching mean and range



The second method involves more extensive editing. Add separate graphs of CONS and OUTPUT (there are three areas now). Both already have the same time axis, but when put on top of each other, the y axes will clash. Double click on the OUTPUT graph, and select the X-axis section. Check Hide:



Then expand the Y-axis section, and, under location and (log)scale or dates, set the Anchor to Maximum. To get the labels and tick marks on the other side, select Labels above.

Change the line type of OUTPUT to number 3. **Figure 5.4b** shows the result. Note that the labels along the *y*-axis have been rotated using Rotate labels on each *y*-axis.

To put the areas exactly on top of each other, it is necessary to set the pixel coordinates of the areas.

Chapter 6

Tutorial on Data Input and Output

In this tutorial, we turn to what can be one of the more tedious operations: inputting new data. Fortunately, OxMetrics can load a wide range of formats to allow for easy exchange with other programs; and can do so from disk files or by copy and paste. [Table 6.1](#) summarizes the available file formats.

Table 6.1 OxMetrics data input/output options		
Load	OxMetrics data files (.oxdata)	§6.3.1
	OxMetrics data files (.in7+.bn7)	§6.3.1
	Excel spreadsheets (.xlsx)	§6.3.2
	Comma separated spreadsheet files (.csv, .csv.zip, .zip)	
	Text files with load info (.dat)	§6.3.3
	Text files by observation (.dat)	§6.3.3
	Stata data files (.dta)	
Save	OxMetrics data files (.oxdata)	§6.2
	OxMetrics data files (.in7+.bn7)	§6.2
	Excel spreadsheets (.xlsx)	§6.2
	Comma separated spreadsheet (.csv)	
	Text files with load info (.dat)	§6.2
	Text files by observation (.dat)	§6.1.2
	Stata data files (.dta)	

The spreadsheet formats are useful to exchange data with other programs.

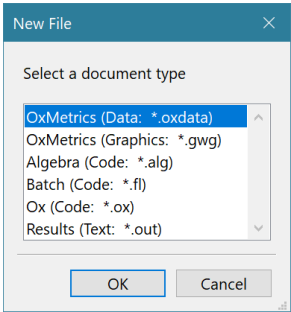
The order in the tutorials below is somewhat different from the table: first we enter a small data set into OxMetrics (16 observations from the tutorial data set variables CONS and INFLAT, from the first quarter of 1955 up to the last quarter of 1958). Next we save this in various formats, which we shall reload. After all, we do not want to spend too much time typing in data.

6.1 Entering data directly into OxMetrics

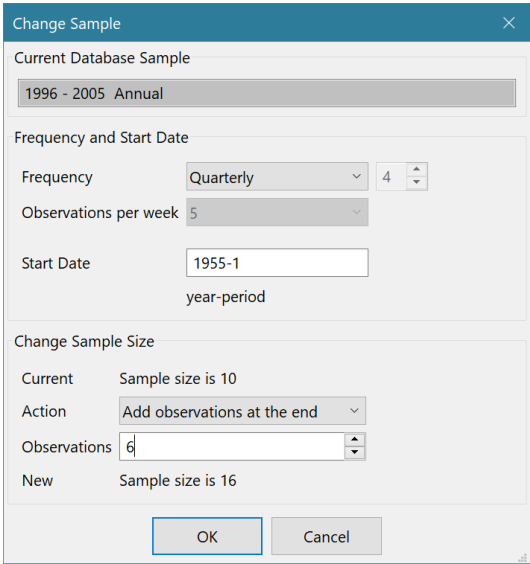
OxMetrics allows you to type data directly into a database, or paste from another program. We look at this first, even though this is likely to be the least used option.

6.1.1 Directly into the database

Start OxMetrics. One of the available options on the File menu is New: click on it then select OxMetrics (Data: *.oxdata):



In the current situation, this brings up the Create a new database dialog to ask you for the information needed to create the database. Fill in the information as shown in the screen capture: Quarterly frequency, start date 1955–1, add 6 observations (the database already has 10):



Press OK to accept the dialog: the empty database appears on screen.

To add variables to the database, keep the focus on it and click on Edit, New variable and type Cons; repeat to create Inflat so there are two variables in the database.

Alternatively, double click on the empty label box at the column head, then enter the name of the variable to be created in the Variable description dialog:

All the observations of Cons and Inflat are set to missing. (NaN, *not a number*, is used internally on computers for the missing value). OxMetrics takes account of this when determining sample sizes for operations, but just leaves gaps when graphing.

There are three ways of entering a value:

1. Edit Value from the context menu to get a pop-up editor to enter values:

2. type a first digit, and the inline editor for the current field is entered.
3. click to select and single click (or press the Enter key) to get the inline editor:

	Cons	Inflat
1955(1)	887	
1955(2)	missing	missing
1955(3)	missing	missing

Type 887 as shown, and press Enter to go to the next field.

Change the next observations to 890, 891, 894, 897, 895, 894, 892, 890, 889, 890, 887, 887, 888, 889, 897. Then for Inflat: -0.46 , -0.38 , -0.2 , 0.2 , 0.5 , 1.7 , 2.3 , 2.1 , 2.1 , 2.0 , 1.8 , 1.2 , 0.7 , 0.4 , -0.26 , -0.57 . Repeat this until you have entered all observations, as in:

	Cons	Inflat
1955(1)	887	-.46
1955(2)	890	-.38
1955(3)	891	-.2
1955(4)	894	.2
1956(1)	897	.5
1956(2)	895	1.7
1956(3)	894	2.3
1956(4)	892	2.1
1957(1)	890	2.1
1957(2)	889	2
1957(3)	890	1.8
1957(4)	887	1.2
1958(1)	887	.7
1958(2)	888	.4
1958(3)	889	-.26
1958(4)	897	-.57

To document each variable, position the cursor on the variable name and press \leftrightarrow , or double-click on the variable name. To return to the Results window, click on Windows and select Results: you may wish to minimize the database window first.

Tip After conversion or entering new data, it is useful to do a graphic inspection which can help to find mistakes. See Figure 6.1 for the two variables created in this tutorial.

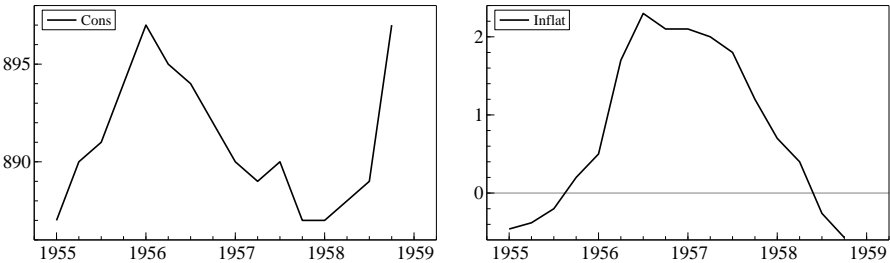


Figure 6.1 Check of Cons and Inflat

The next important task is to save the database to disk. Losing data can be disastrous, and it is best to save it as soon as possible (and make backup copies!). But before moving to §6.2 to see how to save the results of your efforts, we consider data entry by another way. We suggest that you read the next section, but there is no need at this stage to enter the numbers again: you can highlight them in the database and use Edit/copy to put them on the clipboard, then paste to the results window if you wish: the next section explains.

6.1.2 Using the clipboard; using OxMetrics as editor

To get data into a human-readable file we need the services of an editor: any text window will do that job. First change the 1956(3) value for Cons to a missing value. There

are three ways to do that:

1. type missing as the value,
2. Edit Value and check missing in the popup editor,
3. press Del, and select fill with missing value.

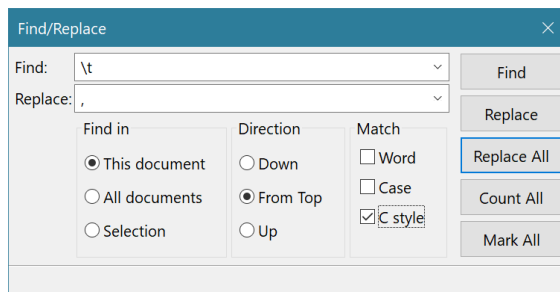
The Ox notation for a missing value is .NaN, which stands for not-a-number.

To avoid typing the data again, select all observations in the database: put the mouse down on the first observation of Cons, then, while keeping the left mouse button down, move to the last observation of Inflat. Or more quickly: select all by starting selection on the first variable label, finishing on the last. Quicker still: click on the top-left empty cell. The screen should look like the capture below.

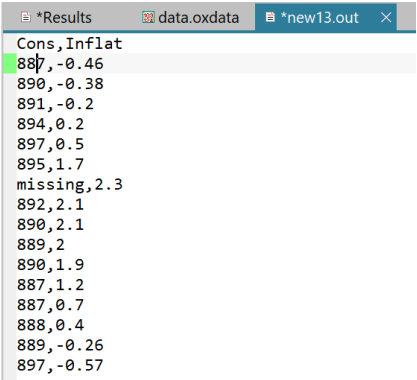
	Cons	Inflat
1955(1)	887	-0.46
1955(2)	890	-0.38
1955(3)	891	-0.2
1955(4)	894	0.2
1956(1)	897	0.5
1956(2)	895	1.7
1956(3)	missing	2.3
1956(4)	892	2.1
1957(1)	890	2.1
1957(2)	889	2
1957(3)	890	1.9
1957(4)	887	1.2
1958(1)	887	0.7
1958(2)	888	0.4
1958(3)	889	-0.26
1958(4)	897	-0.57

The bottom left of the status bar gives the current selection. The right panel still gives the data value under the cursor. Click on the copy icon (shown as two pages on the toolbar) to copy the data to the clipboard.

Next, set create a new results window, and click on the paste icon. At the moment the columns are separated by tab characters. We wish to save as a comma-separated file, so need to change the tabs to a comma. Because the tabs are special characters, this can be done using the Replace dialog using `\t` for the tab, and C-style:



This window now holds:



```
*Results  data.oxdata  *new13.out  X
Cons,Inflat
887,-0.46
890,-0.38
891,-0.2
894,0.2
897,0.5
895,1.7
missing,2.3
892,2.1
890,2.1
889,2
890,1.9
887,1.2
887,0.7
888,0.4
889,-0.26
897,-0.57
```

Note that the names are along the top, and that the data is ordered by observation (the variables are in columns), with the assumption of one observation per line.

The same copy and paste operations would have worked between Excel and OxMetrics. Of course it is also possible to copy and paste directly between an Excel spreadsheet and a OxMetrics database.

See the note regarding copying from Excel on page 82.

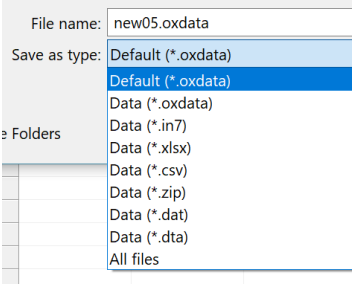
The next step is to save the contents of this window. Type `Alt+f` and then a to execute Save as on the File menu. Enter `testobs.csv` for the filename, and close it in OxMetrics.

Loading the created file is considered in §6.3.3, after we have looked at how to save the database.

6.2 Saving data

With new data entered as in §6.1.1, it is most convenient to save the database in the OxMetrics format. By default this is as a `.oxdata` file.¹

Activate the database, currently called something like `new08.oxdata`, but not yet saved. Save this as `test.oxdata`. The options are:



The OxMetrics format preserves data documentation entered in the database, and the file documentation. These can be modified in a later run of OxMetrics.

¹Previous versions of OxMetrics used two files for its data storage. The first with `.in7` extension, and the second with `.bn7`.

For completeness, we also save the data in the Excel spreadsheet format, and in plain ASCII (text) format. Select File/Save As/Data (*.xlsx); type test in the Save File dialog, the remainder is the same. This creates test.xlsx.

Next repeat the process to save as Data (*.dat), creating test.dat. This is a text file with load information: a line is added for each variable giving the name, sample period and frequency, very similar to the information that is contained in an .in7 file:

```
>Cons 1955 1 1958 4 4
          887          890          891          894
          897          895          894          892
...

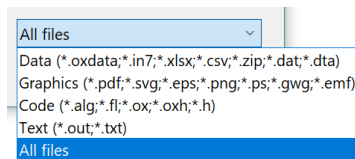
```

This is different from testobs.csv created before: now the data is ordered by variable and has sample information.

6.3 Loading data into OxMetrics

This section discusses the three main types of file which can be loaded directly into OxMetrics. Consult [Chapter 9](#) for additional information on the file formats and further examples.

The file type entry on the dialog when using File/Open specifies which types of file OxMetrics can load:



6.3.1 Loading OxMetrics files

This is the simplest way of getting data into OxMetrics. So once you have your data from another format loaded into OxMetrics, you might wish to save it in the OxMetrics format, and use that from then onwards. An example was given in the first tutorial, where we loaded the tutorial data file data.oxdata. Try loading the test.oxdata file which was created earlier.

6.3.2 Loading spreadsheet files

A test file called test.xlsx was created above. To load it in OxMetrics, select File/Open, and select the file. The spreadsheet is loaded directly into OxMetrics.

This is what test.xlsx looks like in Excel (try to load it if you own a copy), for the first few observations:

	A	B	C
1		Cons	Inflat
2	1955(1)	887	-0.46
3	1955(2)	890	-0.38
4	1955(3)	891	-0.2
5	1955(4)	894	0.2
6	1956(1)	897	0.5
7	1956(2)	895	1.7
8	1956(3)	#N/A	2.3
9	1956(4)	892	2.1
10	1957(1)	890	2.1
11	1957(2)	889	2

Note that the missing value is encoded as #N/A in Excel.

This illustrates what OxMetrics expects when reading a spreadsheet file:

- ordered by observation (that is, variables are in columns);
- columns with variables are labelled;
- there is an unlabelled column with the dates (as a text field), in the form year–period, for example, 1980–1 (or: 1980Q1 1980P1 1980:1 etc.); Excel dates can be read as well, see below.
- the data form a contiguous sample; non-numeric fields are interpreted as missing values.

More details are in [Chapter 9](#). However, switching between programs using Windows facilities with copying and pasting is usually the easiest approach to entering data that already exist in another program, *provided that all significant digits are retained*. This is a problem with Excel: for example, when copying an Excel cell with the value 908.212280273437 on to the clipboard, it is put there as 908.212280 (the display value), thus losing significant digits.

6.3.3 Loading a text (human-readable) file

Two human-readable data files were created in this chapter, one by copying to the results window, and saving as `testobs.csv` (ordered by observations), and `test.dat` at the end of [§6.2](#) (ordered by variable, with load info).

To load `test.dat` into OxMetrics, use File/Open/Human readable. After selecting the correct file (`test.dat`), we note that the file is loaded automatically as desired (with the help of the load info that is embedded in the file). View the database or graph the data to check the results.

Finally, load `testobs.csv`. This has no sample information. Activate the database and use Edit/Change Sample to set the frequency to quarterly and the start date to 1955–1.

6.4 Adding variables using the clipboard

Often one wishes to lift variables out of other data files, and add them to the current database. OxMetrics supports this operation through copy and paste. You can select individual variables in one database and copy them to another. Once created, the fre-

quency of the database is fixed, but the sample period will be extended automatically if required, or can be extended explicitly as shown in the next section.

Let us try to append the INC variable from the tutorial data set to our database. Select File/Open, choose `data.oxdata` on the file open dialog, and press **Enter** to accept. Highlight the observations on INC from 1955(1) to 1958(4) only, and then press the two-pages icon to copy to the clipboard. Now set focus to the test database, put the cursor on the empty field at 1955(1), next to Inflat, and press the paste icon. This copies the data, and the new variable gets the name Var3, which is easily changed.

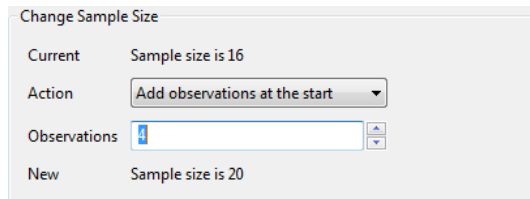
When a whole variable is copied onto the clipboard, the name is also included. When pasted into a database as a new variable, the name will be used for the new variable (unless a variable with that name already exists).

6.5 Changing the sample period

6.5.1 Extending the sample period

It is possible to add observations prior to the current sample start, or after the sample end. Such an operation is required when new data are collected, or ex-ante forecasts are to be stored for further analysis. When storing forecasts, or pasting data, the sample end is extended automatically.

Suppose we wish to add a year of data at the beginning of the test data created in §6.2. Remember that the data in `test.oxdata` have a sample period of 1955 (1) to 1958 (4). Focus on the database and select Edit/Change Sample. Fill this in as shown: 4 observations at the start, then press OK to accept:



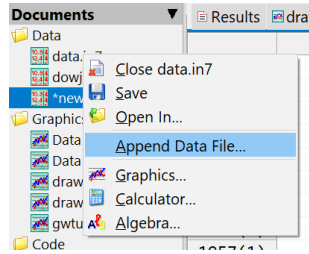
The same procedure can be used to add observations, but a simpler way is just to double click beyond the end of the database: the editor pops up, and the database is automatically extended.

6.5.2 Reset starting date

This was done in §6.3.3: the number of observations remained unchanged, and no data will be lost or changed.

6.6 Appending data

It is possible to append a database from disk to an open database, using Append Data File on the context menu in the workspace:



Append will only work if the frequencies match. OxMetrics will make additional adjustments when these are required:

- the sample of the open database will be extended to encompass that of the appended database;
- if a variable from the appended database already exists in the open database (remember that this requires the names to be identical, including upper/lower case), then the existing variable is updated, except where the appended variable has missing values (for those observations the existing variable is not changed).

In this case, a message will warn you about the update: *Found duplicate variables in appended database: existing variables adjusted.*

For example:

existing	appended	resulting
Cons	Cons	Cons
887	880	880
890	missing	890
891	900	900
894	missing	894
897	910	910

6.7 Working with daily and weekly data

In §3.3 we gave an introduction to dated data, loading the weekly Dow Jones index. In this section we illustrate how an undated database can be converted to a dated database.

First we contrast the two methods for dating observations:

Fixed frequency Denoted by a start year, start period and frequency. From this, the date of any subsequent observation can be worked out. For example, if the start year is 2005, the start period 1, and the frequency 4 (quarterly), then the sixth observations is for 2006(2), the second quarter. This works well for monthly, quarterly and annual data. For undated data we simply set the start year and period to one, and the frequency as well.

Dated The fixed frequency method does not work for weekly or daily data: not every year has the same numbers of observations. The solution adopted in OxMetrics is as follows: an integer value represents a date. A database variable can be classified as of type date, in which case the date is displayed, rather than the underlying value.

For example 2005-1-1 (1st January 2005 — OxMetrics uses the ISO representation year-month-day) corresponds to 2453372. To check this, just enter 2005-1-1 as an observation value in a database: this is automatically translated² to 2453372. It is not necessary to work with these large numbers directly, there are several Algebra functions to help.

The following criteria must be satisfied for a database to be dated:

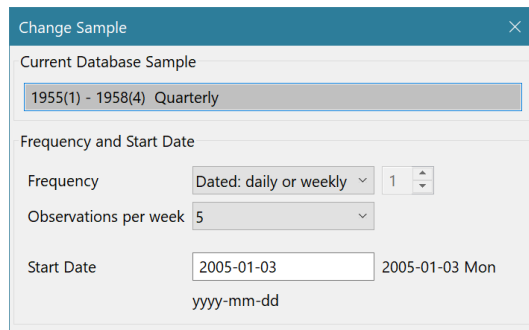
1. the first column must be of type Date,
2. the first column holds date values,
3. the optional fractional part of this indicates time as a fraction of 24 hours; e.g. 2453372.75 is 18:00 on 2005-1-1 (2005-1-1T18:00 in ISO notation),
4. the first and last observation must be valid, i.e. cannot be missing.

Note that the fixed sample period is still available for a dated database, but this is usually set as for an undated database (start year, start period, frequency all set to one).

6.7.1 Using Change Sample to create a daily database

Our next step is to create a database that is dated. We continue with the Cons, Inflat test database that was constructed, but any will do.

The simplest way is to use the Change Sample command, set a daily frequency with 5 days per week and set the start date to 2005-1-3 (the first Monday in January, as can be seen from the dialog:

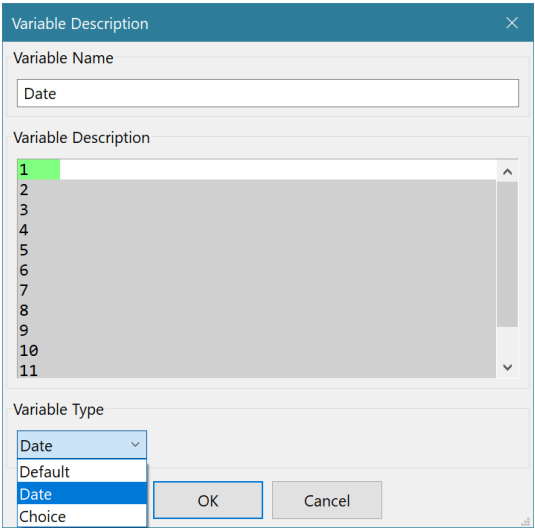


Accepting this inserts a variable called Date as the first column:

	Date	Cons	Inflat	
2005-01-03	2005-01-03	887	- .46	
2005-01-04	2005-01-04	890	- .38	
2005-01-05	2005-01-05	891	- .2	
2005-01-06	2005-01-06	894	.2	
2005-01-07	2005-01-07	897	.5	
2005-01-10	2005-01-10	895	1.7	

This is a variable as any other: it can be renamed, transformed, moved, deleted, etc. Although some of these will make the database undated again. Double clicking on the Date label, and changing the variable type to Default:

²Only when the variable type is 'Date' will it be shown as a date.



will make the database undated:

	Date	Cons	Inflat	
1	2453374	887	-.46	
2	2453375	890	-.38	
3	2453376	891	-.2	
4	2453377	894	.2	

Click on Undo to reset the database. Renaming the variable does not make the database undated, because the name is not really used elsewhere.

6.7.2 Using Algebra to create a daily database

The following Algebra code creates separate variables with the year, month (January=1) and day (1–31 depending on the month):

```
y = year();
m = month();
d = day();
```

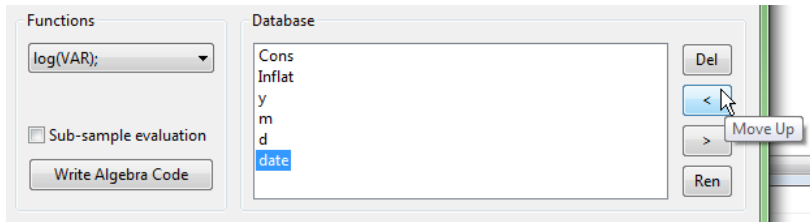
If we now delete the Date variable, the database becomes undated:

	Cons	Inflat	y	m	d	
1	887	-.46	2005	1	3	
2	890	-.38	2005	1	4	
3	891	-.2	2005	1	5	
4	894	.2	2005	1	6	

This is the format in which some data sets come. To make it dated, use makedate to create a date variable:

```
date = makedate(y, m, d);
```

Then move the variable to the first position in the database using the Move Up button in the Algebra editor (or in the Calculator):



Finally, double click on the new date variable, and change the type to Date. And the database is dated again.

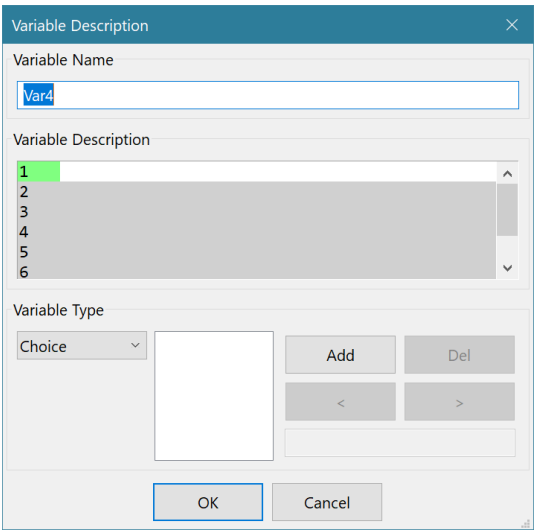
6.8 Working with choice variables

A third type of variable is called ‘Choice’: this has text labels attached to values. Normally the values of such a variable are integers, ranging from zero upwards. Then each zero has the same label, each one another, etc. An example would be a country panel, where we find labels such as UK, USA etc. more informative than 14, 15 (say). Another example is tick-by-tick data, where it can denote a stock exchange. Within OxMetrics, there is fairly limited benefit of this variable type. Ox programs, however, can also access this aspect in the Database class, which can make output more readable.

To illustrate the creation of a choice variable, continue with one of the small test databases, say test.oxdata. (It doesn’t matter if it is dated or undated.) Add a variable called Var4 which at first is all missing values. Then set the first third of observations to zero (select the block, right-click on the selection, choose Edit Value, and set it to 0). Set the second block to one, and the final block to two:

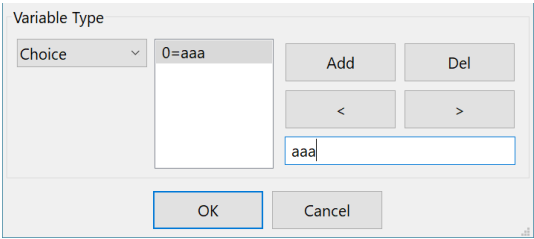
Cons	Inflat	Var4
887	-.46	0
890	-.38	0
891	-.2	0
894	.2	0
897	.5	0
895	1.7	1
missing	2.3	1
892	2.1	1
890	2.1	1
889	2	1
890	1.8	2
887	1.2	2
887	.7	2
888	.4	2
889	-.26	2
897	-.57	2

Double click on the variable name, and change the type to Choice:



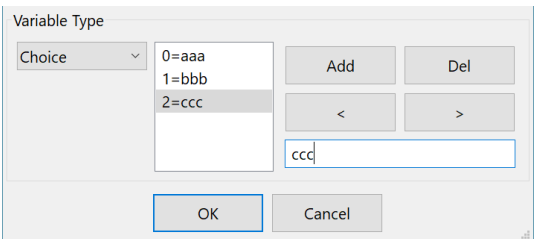
The 'Variable Description' dialog box shows 'Var4' as the variable name. The 'Variable Description' list contains numbers 1 through 6, with '1' selected. The 'Variable Type' is set to 'Choice'. There are 'Add', 'Del', '<', and '>' buttons, and an empty text field for adding labels. 'OK' and 'Cancel' buttons are at the bottom.

Now click on Add to add the first label, aaa, in the edit field:



The 'Variable Type' dialog box shows 'Choice' selected. The list contains '0=aaa'. The text field contains 'aaa'. 'Add', 'Del', '<', and '>' buttons are present. 'OK' and 'Cancel' buttons are at the bottom.

Then add the next value, label pair. Finally, also add cc:



The 'Variable Type' dialog box shows 'Choice' selected. The list contains '0=aaa', '1=bbb', and '2=ccc'. The text field contains 'ccc'. 'Add', 'Del', '<', and '>' buttons are present. 'OK' and 'Cancel' buttons are at the bottom.

Now the Var4 variable is displayed with labels instead of numbers. It is still a numerical variable though, which can be used in modelling as any other variable.

Cons	Inflat	Var4
887	-.46	aaa
890	-.38	aaa
891	-.2	aaa
894	.2	aaa
897	.5	aaa
895	1.7	bbb
missing	2.3	bbb
892	2.1	bbb
890	2.1	bbb
889	2	bbb
890	1.8	ccc
887	1.2	ccc
887	.7	ccc
888	.4	ccc
889	-.26	ccc
897	-.57	ccc

Not all file types can preserve the choice labels correctly. For saving you can use `.oxdata`, `.in7/.bn7`, `.xlsx` or `.dta`. Other formats will either lose the labels completely, or are unable to guarantee that the label–value association stays intact.

This completes our treatment of data input and output. The remainder of the tutorials will use either data that is provided with OxMetrics, or artificial data created inside OxMetrics. After that you will have your own data to analyze; we hope that OxMetrics’s ample facilities allow you to enter, edit, save and document your data easily and quickly.

Chapter 7

Tutorial on Data Transformation

We briefly described the Calculator and Algebra in [Chapter 2](#). Here we explore the transformation and data generation possibilities. Along the way, we round off the options in the Model menu by looking at the Tail probabilities dialog.

If you are not inside OxMetrics at the moment, restart and load the tutorial data set `data.oxdata`.

7.1 Calculator

The calculator enables easy manipulation of the variables in the database, and is a convenient way to write algebra expressions.

The aim is to build a valid algebra expression in the expression window (without the assignment part and the terminating semi-colon). All successful transformations are logged in the Results window, from where they can be cut and pasted into an algebra file for later use if desired. Advanced algebra is discussed in the next section, whereas [Chapter 10](#) has a more formal description of the language.

Suppose, for example, that you have selected the variable called CONS (we use the artificial data set again), and then pressed the log button. Then the expression will read `log(CONS)`. When pressing evaluate (the = sign), the logarithm of CONS will be computed. Now you have to assign the new variable a name, with LCONS as the suggestion made by OxMetrics. If you accept this suggestion, LCONS will be added to the database. If LCONS already exists, OxMetrics will ask whether you wish to overwrite the existing variable.

If you have ticked the Sub-sample evaluation check box, you can also select a sample period over which you wish to compute `log(CONS)`. If you create a new variable in this way, it will be set to the missing value outside the sample. If you overwrite an existing variable, it will be left untouched outside the sample period.

Examples using the calculator were given in [Chapter 2](#). Some additional examples follow here.

First, creating a lag: select CONS, click on lag, specify 4 (to create a four-period lag) in the lag length dialog, OK to accept, and click on the button with the = symbol at which point a suggested name (CONS.4) appears in a small window, to accept and the created variable is added to the database. In general, lags for modelling should not be created this way, although it can be useful for (e.g.) graphing.

Next, to create a step dummy of the 0-1 variety for the oil crisis: click on dummy, then enter 1973 3 in the Sample start part (leave the Sample end edit field unchanged, so the 1 extends to the end of the sample; a capture of the dialog was shown on page 22), press OK and click on = (or C to clear the expression) and specify the name s1973p3.

Third, a complex example: to create the five-period moving average of CONS:

$$\frac{1}{5} \sum_{i=0}^4 \text{CONS}_{t-i}.$$

Select CONS in the database, locate movingavg in the function list, double click on it to see movingavg(CONS, LAG, LEAD). Replace LAG by 4 and LEAD by 0. Click on =, and specify the name MA40CONS. To create the moving average centred around the current observation (i.e. using two lags, the current value and two leads), specify movingavg(CONS, 2, 2). Note that this variable lagged two periods is the same as MA40CONS.

Other operations are equally simple. To create an Almon polynomial in INC of 2^{nd} order over eight lags (producing three terms): Click on INC, and then double click on almon, to get the expression almon(INC, LAG, POWER). Change this to almon(INC, 8, 0) to create the first Almon variable. Call this A0INC. Repeat this to create A1INC as almon(INC, 8, 1) and A2INC as almon(INC, 8, 2). They are computed using

$$\sum_{i=0}^8 (9-i)^j x(t-i) / \sum_{i=0}^8 (9-i)^j \quad \text{for } j = 0, 1, 2,$$

and eight initial observations are lost. Graph these to see how collinear they are.

To create the price level from INFLAT as our final example, we need to integrate (cumulate) INFLAT. This is easily done using the cum or stock function. First highlight INFLAT, then double click on the cum function to cumulate. Press = to accept the expression cum(INFLAT), call it P, so that $P_t = P_{t-1} + \text{INFLAT}$ is created (graph it and see). Another way to create this is selecting INFLAT, but clicking on the stock function instead. Replace ARVAL (one minus the autoregressive coefficient) by 0, and INIT (the initial value) by 0. Call this SINFLAT, and check that P and SINFLAT are identical. If you wanted P equal to 100 in 1970(1), say, when it is 86.28879985, select the same options but replace the INIT argument of stock with 13.71120015, which will set $P = 100$ in 1970(1). Alternatively, return to the Results window where $\text{SINFLAT} = \text{stock}(\text{INFLAT}, 0, 0)$; is shown, copy the observation for P in 1970(1) to the clip board and paste to the Results window, create $c=100-86.2887998521328$; and $\text{SINFLAT} = \text{stock}(\text{INFLAT}, 0, c)$; , highlight these two algebra statements and press Ctrl+A to execute.

7.2 Advanced algebra

7.2.1 Introduction

Algebra allows us to do transformations by typing formulae as described in [Chapter 2](#). These are then executed by OxMetrics. A wide range of functions is available and algebra formulae can be saved and reused. In this section we will only give examples; algebra is documented in [Chapter 10](#).

All the code written by the calculator makes up valid algebra code as just noted. The algebra editor is activated by selecting Tools/Algebra (**Alt+t,a**), the short-cut key **Alt+a**, or the algebra toolbar button.

Algebra uses database variables as follows: if a left-hand variable is already in the database it will be overwritten, otherwise it will be created. Variables on the right must exist, possibly because of preceding lines of algebra code. As noted, algebra is case-sensitive, meaning that **LCONS**, **LCons** and **lcons** refer to three different variables. Algebra code is logged to the Results window.

Tip Select a function in the function list or variable in the variable list to paste it into the editor. This saves typing.

Tip Algebra can also be run directly from the results window: highlight the block of algebra code, and press **Ctrl+A** to run the code.

Consider the `insample` function, which has four arguments: `startyear`, `startperiod`, `endyear`, `endperiod`. It returns 1 (or **TRUE**: everything which is not 0 is **TRUE**) if the observation under consideration falls within the sample, otherwise it returns 0 (**FALSE**). The conditional assignment works as follows: the conditional statement (the ‘if’ part) is followed by a question mark and the ‘then’ part, which is followed by a colon and the ‘else’ part. Read:

```
i1980p1 = insample(1980, 1, 1980, 1) ? 1 : 0;
```

as: `i1980p1` takes on the value 1 for the observations which are in the specified sample, and the value 0 for the other observations. Note that exactly the same result can be obtained by writing `i1980p1 = insample(1980, 1, 1980, 1);`.

An error message pops up if you make a mistake. The error can be corrected on returning to the algebra editor.

7.2.2 Database for advanced algebra

Select File/New to create a database with 600 observations, using frequency one.

7.2.3 Statistical distributions

We shall first look at some graphs of densities and corresponding distribution functions. Most often we work with continuous density functions, and it is assumed that you are familiar with the basic principles (Volume I of the PcGive books discusses statistical theory). For example, the standard normal density is defined as:

$$f_x(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2},$$

with (cumulative) distribution function (CDF):

$$F_x(x) = \int_{-\infty}^x f_x(u) du.$$

This integral cannot be written explicitly for the normal distribution. F is a nondecreasing function and:

$$F_x(-\infty) = 0, F_x(\infty) = 1.$$

If X has a continuous distribution, then the expectation of X is:

$$E[X] = \int_{-\infty}^{\infty} u f_x(u) du.$$

We write $X \sim N(0, 1)$ to say that the random variable X follows a standard normal distribution. Three other distributions of interest are the student-t, the F and the χ^2 distribution:

$$\begin{aligned} t(k) \quad f_x &= \frac{\Gamma(k/2+1/2)}{\Gamma(k/2)} \frac{1}{\sqrt{k\pi}} \left(1 + \frac{x^2}{k}\right)^{-k/2-1/2} & k = 1, 2, \dots \\ F(m, n) \quad f_x &= \frac{\Gamma(m/2+n/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{m/2-1} \left(1 + \frac{m}{n}x\right)^{-m/2-n/2} & x > 0, \\ & & m, n = 1, \dots \\ \chi^2(k) \quad f_x &= \frac{1}{\Gamma(k/2)} \left(\frac{1}{2}\right)^{k/2} x^{k/2-1} e^{-x/2} & x > 0, \\ & & k = 1, 2, \dots \end{aligned}$$

$\Gamma(\cdot)$ is the gamma function:

$$\Gamma(u) = \int_0^{\infty} x^{u-1} e^{-x} dx \quad \text{for } u > 0.$$

For integer arguments n : $\Gamma(n+1) = n! = n(n-1)(n-2) \cdots 1$. Most test statistics in PcGive have one of these distributions (but often only approximately or in large samples).

We compute 600 values from $N(0, 1)$, $t(2)$, $\chi^2(3)$ and $F(3, 2)$, over a range of -6 to 6 , in steps of $.02$. This is the variable called i in the first line of algebra. The algebra code is in the file `tutdist.alg` in the `0xMetrics9\algebra` folder; the section for the densities is:

```
i = (trend() - (NOBS/2)) / (NOBS/12);
normal = densn(i);
chi = (i > 0) ? denschi(i,3);
student = denst(i,2);
F = (i > 0) ? densf(i,3,2);
```

Run the code from `tutdist.alg`, and do a scatter plot of each of the four variables against i , as in Figure 7.1. This shows the four densities.

It is also possible to evaluate the densities directly, instead of using the built-in functions. Note in the second line that we write $-(i^2)$; $-i^2$ would be equivalent

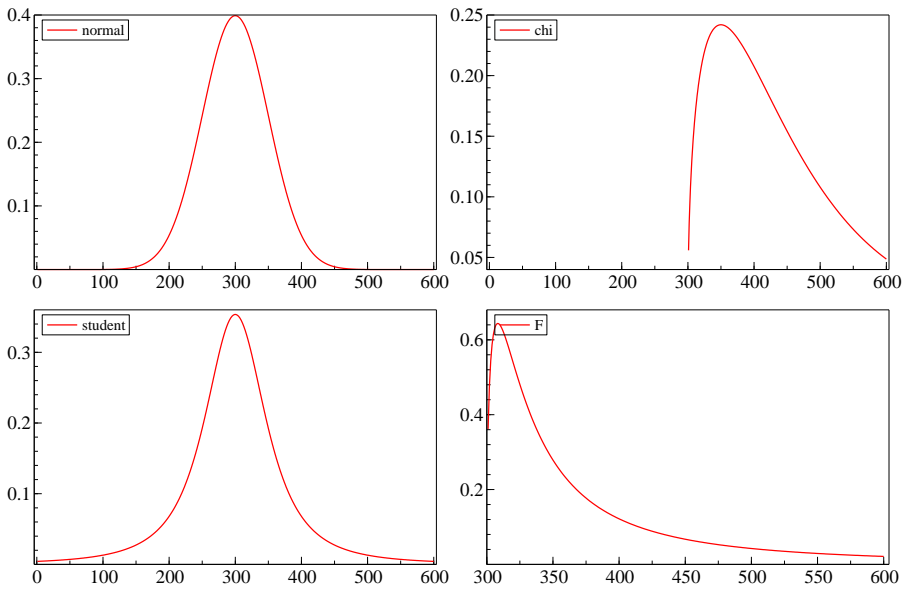


Figure 7.1 Some important densities

to $(-i)^2$, as the unary minus operator has a higher precedence than power (see [Table 10.1](#) on page 131). The $\Gamma(\cdot)$ function is not available in the algebra, but the $\log \Gamma(\cdot)$ is and we use the fact that $\Gamma(\cdot) = \exp(\log \Gamma(\cdot))$.

```
normal = exp(-i^2 / 2) / sqrt(2 * PI);
chi = (i > 0) ? 1 / exp(loggamma(1.5)) * 0.5 ^ 1.5
      * i ^ 0.5 * exp(-i / 2);
student = exp(loggamma(1.5) - loggamma(1)) /
          (sqrt(2 * PI) * (1 + i^2 / 2) ^ 1.5);
F = (i > 0) ? exp(loggamma(2.5) - loggamma(1.5)
                - loggamma(1)) * 1.5 ^ 1.5 * i ^ 0.5
          / (1 + 1.5 * i) ^ 2.5;
```

To translate the densities into approximate CDFs, the following code is used in `tutdist.alg` to cumulate and scale the outcomes (which are in increasing order given our choice of i for the densities):

```
cnormal = cum(normal) / (NOBS/12);
cchi = cum(chi) / (NOBS/12);
cstudent = cum(student) / (NOBS/12);
cF = cum(F) / (NOBS/12);
```

We have actually performed a numerical integration of the densities! At each point we computed the height of the function, and added these up. Since the width of each bar is only $1/50$, we have to divide by 50 (the surface of a bar equals *height* \times *width*). The CDFs are graphed in [Figure 7.2](#).

It can be noted from [Figure 7.1](#) that the $F(3, 2)$ distribution does not have a finite variance: it approaches 1 very slowly. The same holds for the $t(2)$ distribution.

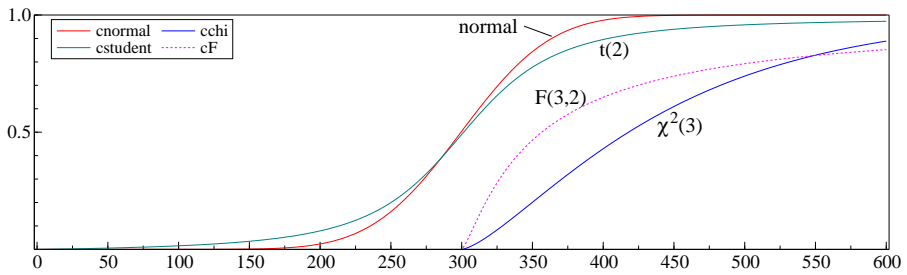


Figure 7.2 Cumulative distribution functions from 600 points

The approximate 75% critical values as read from the database are:

$N(0, 1)$.66
$t(2)$.86
$\chi^2(3)$	4.1
$F(3, 2)$	3.16

The accuracy turns out to be quite good. If we compute the p-values using Tools/Tail probability we find:

$N(0, 1, 1\text{-sided})$	=	0.66	[0.2546]
$t(2, 1\text{-sided})$	=	0.86	[0.2402]
$\chi^2(3)$	=	4.1	[0.2509]
$F(3, 2)$	=	3.16	[0.2496]

This dialog gives $P(X > x)$ when 2-sided is not marked, in which case we would ideally get p-values of 25%.

7.2.4 Random number generators

The random number generators in OxMetrics can be used to illustrate statistical concepts, and to give examples of a wide range of commonly used processes. The main generators are for the uniform distribution (all other distributions can be derived from this), and the standard normal distribution. In addition, OxMetrics provides functions to draw from the t , χ^2 , and F distributions. Other distributions could be derived if desired. An alternative way to draw from a $\chi^2(k)$, for example, is to compute the sum of k squared standard normals. Doornik (2006) discusses recent aspects of random number generation.

The following experiments use 10 000 observations, but you could keep on using the database with 600 observations. Otherwise create a new database with 10 thousand annual observations. Draw from the $N(0, 1)$, $t(100)$, $\chi^2(3)$, and $F(3, 100)$ distributions, and compute the probabilities (using the correct distributions) to get a value less than the observed numbers:

```
phi = rann();
stu = rant(100);
chi = ranchi(3);
fdi = ranf(3, 100);
```

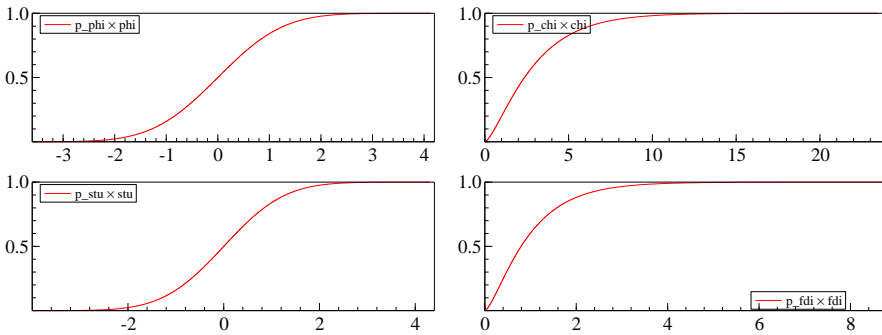


Figure 7.3 Cumulative distribution functions

```
p_phi = 1 - tailn(phi);
p_stu = 1 - tailt(stu, 100);
p_chi = 1 - tailchi(chi, 3);
p_fdi = 1 - tailf(fdi, 3, 100);
```

The algebra code is in the file `tutran1.alg`. When using 10 000 observations, the computations may take a little while.

Using these data, we can draw the cumulative distribution functions, see [Figure 7.3](#), and compare with [Figure 7.2](#). In this case we used a t and F distribution with finite variance. You could try the distributions of [§7.2.3](#): there will be some large outliers, extending the scale of the graphs.

A check of the random number generators can be based on the probabilities. Sort each vector of probabilities, and corresponding random numbers with it:

```
_sortby(p_phi, phi);
_sortby(p_stu, stu);
_sortby(p_chi, chi);
_sortby(p_fdi, fdi);
```

[Figure 7.4](#) shows that the probabilities form a straight line. A plot of the sorted random numbers look like a mirror image of [Figure 7.3](#). Alternatively, graph the histogram and densities of phi, stu, chi and fdi to see sample analogues of the distributions.

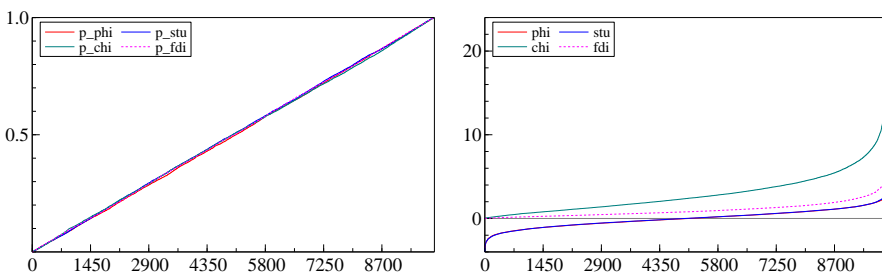


Figure 7.4 Check on CDFs

7.2.5 Generating data

Finally, we shall generate data on various autoregressive and related processes. The database is assumed to consist of 600 ‘annual’ observations, as in §7.2.3. The file `tutran2.alg` contains the algebra code used in this section.

Consider the following examples (note that the roots in the MA(2) process are -0.87 and -0.23):

white noise	$\epsilon_t \sim N(0, 1),$
AR(1)	$y_t = 0.6y_{t-1} + \epsilon_t,$
AR(2)	$y_t = 0.4y_{t-1} - 0.7y_{t-2} + \epsilon_t,$
MA(2)	$y_t = \epsilon_t + 1.1\epsilon_{t-1} + 0.2\epsilon_{t-2},$
ARMA(2,2)	$y_t = -1.4y_{t-1} - 0.5y_{t-2} + \epsilon_t - 0.2\epsilon_{t-1} - 0.1\epsilon_{t-2},$
ARCH	$y_t = \sigma_t \epsilon_t, \quad \sigma_t^2 = 1 + 0.6y_{t-1}^2.$

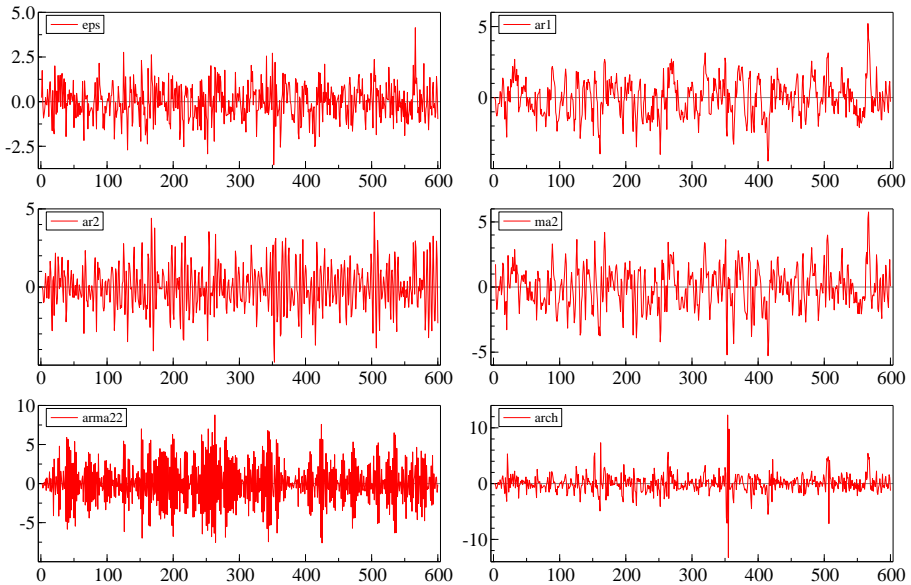


Figure 7.5 Time series graphs of 6 processes

The middle four processes are used as examples in [Priestley \(1981\)](#). For the ARCH (autoregressive conditional heteroscedasticity) process, see, for example, [Harvey \(1993\)](#). The following algebra code generates a sample from the six processes:

```

ranseed(-1);
eps = rann();
ar1 = year() >= 3 ? .6 * lag(ar1,1) + eps : 0;
ar2 = year() >= 3 ? .4 * lag(ar2,1) - .7*lag(ar2,2) + eps : 0;
ma2 = eps + 1.1 * lag(eps,1) + .2 * lag(eps,2);
arma22 = year() >= 3 ? -1.4 * lag(arma22,1)
    - .5 * lag(arma22,2) + eps
    - .2 * lag(eps,1) - .1 * lag(eps,2) : 0;
arch = year() >= 3 ? sqrt(1 + 0.6 * lag(arch,1)^2) * eps : 0;

```

The `ranseed()` function sets the seed of the random number generator, an argument of `-1` resets the seed to the default value. The variable `eps` is $N(0, 1)$ and is reused in each process. The autoregressive parts use conditional statements. If the year is ≥ 3 (we use annual data, so this corresponds to an observation index ≥ 3), the part following the `?` is executed. For the first two observations the value zero is used. In this way a variable can be integrated safely (without the conditional statement all observations would have the missing value). The test `year() >= 3` could also be written as `insample(3,1,600,1)` or `lag(eps,2) != MISSING`. Try to understand and compare these alternatives.

Figure 7.5 graphs the results, after adjusting the axes to give the same vertical range. The correlograms and spectral densities are given in Figure 7.6.

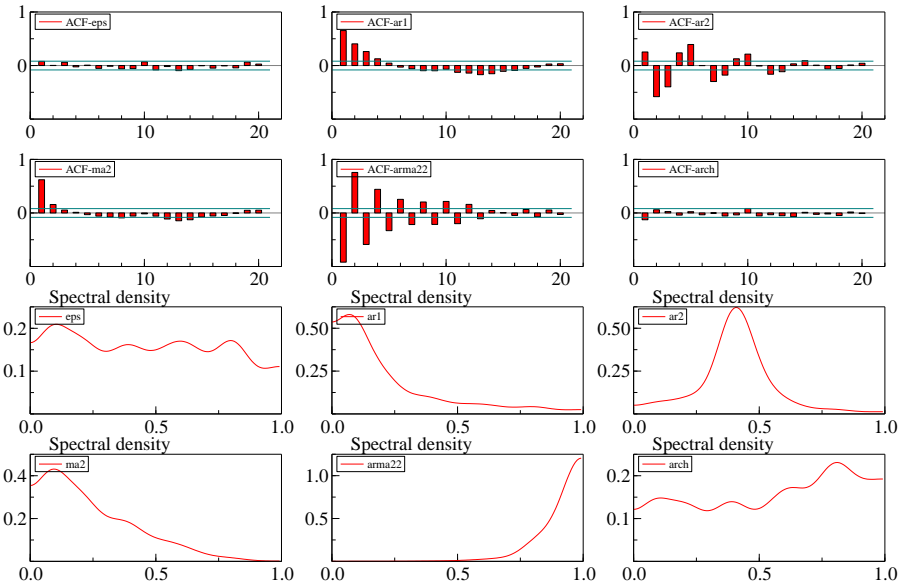


Figure 7.6 Correlograms and spectral densities of six processes

The remaining examples correspond to the equation:

$$y_t = \alpha + \beta y_{t-1} + \mu t + \epsilon_t, \quad \epsilon_t \sim N(0, 1),$$

using parameter values:

z1: $\alpha = 0, \beta = 1, \mu = 0$	z2: $\alpha = 0, \beta = .9, \mu = 0$	z3: $\alpha = 0, \beta = .5, \mu = 0$
m1: $\alpha = .1, \beta = 1, \mu = 0$	m2: $\alpha = .1, \beta = .9, \mu = 0$	m3: $\alpha = .1, \beta = .5, \mu = 0$
t1: $\alpha = .1, \beta = 1, \mu = .001$	t2: $\alpha = .1, \beta = .9, \mu = .001$	t3: $\alpha = .1, \beta = .5, \mu = .001$

The algebra code uses the `eps` variable created above.

```

z1 = year() >= 3 ? 1 * lag(z1,1) + eps : 0;
z2 = year() >= 3 ? .9 * lag(z2,1) + eps : 0;
z3 = year() >= 3 ? .5 * lag(z3,1) + eps : 0;
m1 = year() >= 3 ? .1 + 1 * lag(m1,1) + eps : 0;

```

```

m2 = year() >= 3 ? .1 + .9 * lag(m2,1) + eps : 0;
m3 = year() >= 3 ? .1 + .5 * lag(m3,1) + eps : 0;
t1 = year() >= 3 ? .1 + .001*trend() + 1*lag(t1,1) + eps : 0;
t2 = year() >= 3 ? .1 + .001*trend() + .9*lag(t2,1) + eps : 0;
t3 = year() >= 3 ? .1 + .001*trend() + .5*lag(t3,1) + eps : 0;

```

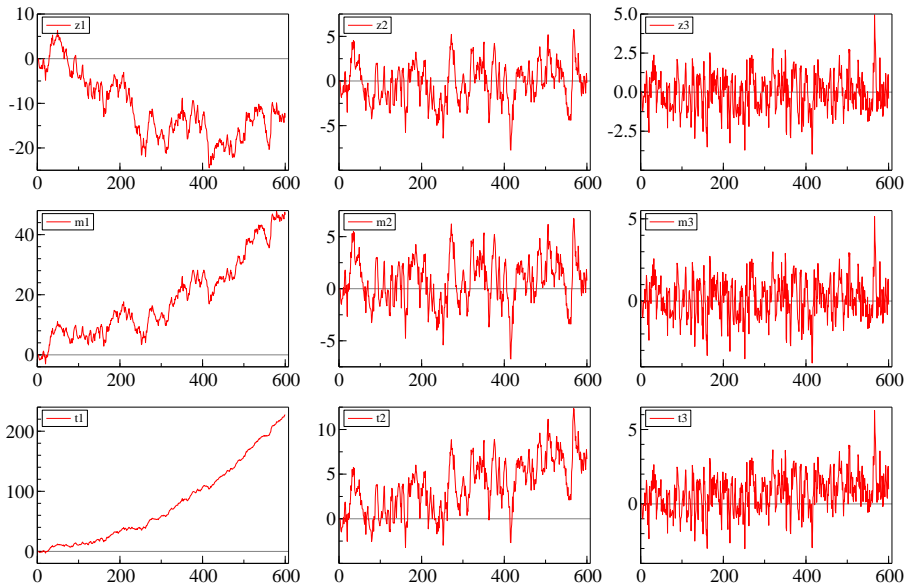


Figure 7.7 AR(1) with various parameter values

Figure 7.7 clearly shows the impact of the different parameter values on the behaviour of the functions.

7.2.6 Smoothing data

As our last tutorial example, we show how the natural cubic spline function can be used to fit missing values. Load the tutorial data set DATA.IN7.

First edit the CONS variable to set the observations for the years 1958, 1968 and 1978 to missing values (for each year: select the four observations, press Enter, set to Missing value, press Enter again, and apply to the whole selection). Afterwards, make sure that you do not save this modified database. An alternative way to set the missing values is to run the following algebra code:

```

CONS = insample(1958,1,1958,4) || insample(1968,1,1968,4)
|| insample(1978,1,1978,4) ? MISSING;

```

Select the Scatter plot graphics dialog, and do a cross plot of CONS against time (so there is no need to choose an X variable), adding a Cubic spline smooth using all three bandwidth settings. The results should be similar to Figure 7.8. The label of the spline line gives the equivalent number of parameters (comparable to the number of variables used in a linear regression). So we see that a bandwidth of 1600 corresponds to nearly 10 parameters (it also corresponds to the Hodrick–Prescott filter, see §8.10.2),

whereas automatic bandwidth selection is much less smooth, with an equivalence to 47 parameters.

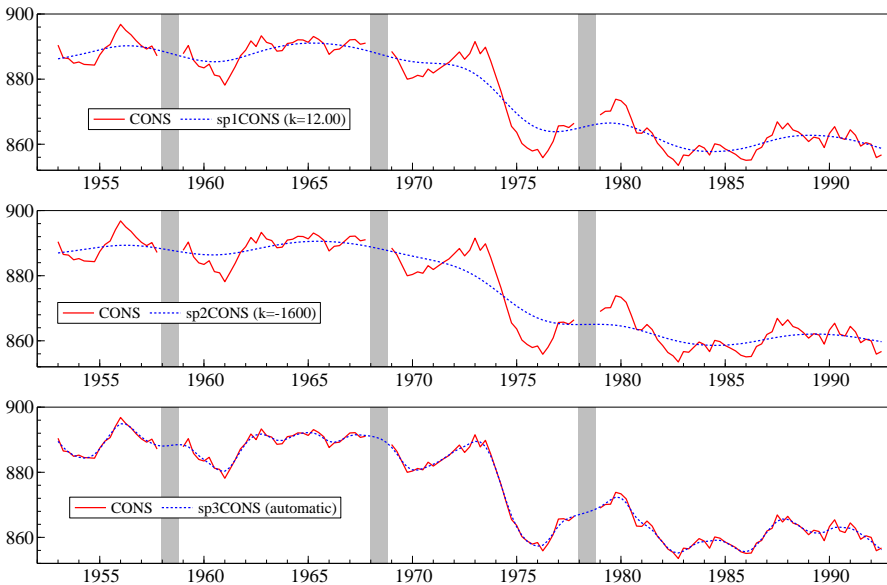


Figure 7.8 Splines with various bandwidth settings

The following algebra code creates the three smooths in the database (see §10.5.5):

```
sp1CONS = smooth_sp(CONS, 12, sp1CONS);
sp2CONS = smooth_sp(CONS, -1600, sp2CONS);
sp3CONS = smooth_sp(CONS, 0, sp3CONS);
```

The first line sets the bandwidth implicitly by specifying an equivalent number of parameters (12), the next sets the bandwidth directly to 1600, and the last uses automatic bandwidth selection. You could try to add the appropriate variable to each graph by selecting the area (left mouse button), and then using Add Variable. When done correctly, each added variable should exactly cover the pre-existing spline.

To add the shading, first create the shading variable:

```
shading = CONS == MISSING;
```

Then add this variable to each graph, changing the line type to shading, and removing the Y label so that it doesn't show in the legend.

OxMetrics Reference

Chapter 8

OxMetrics Statistics

OxMetrics provides many convenient graphical options for prior data analysis. For simplicity, we denote the selected variable by x_t , $t = 1, \dots, T$. This chapter summarizes the underlying formulae, discussing each available graph type in turn.

8.1 Actual series and scatter plots

Actual series gives a graph for each selected variable, showing the variable over time. The All scatter plots option gives scatter plots between all selected variables (omitting redundant scatter plots).

8.2 Mean, standard deviation and variance

These are relevant for the statistics described below. For a series x_t , $t = \dots, T$:

$$\begin{array}{ll}
 \text{sample mean} & \bar{x} = \frac{1}{T} \sum_{t=1}^T x_t, \\
 \text{sample variance} & \tilde{\sigma}_x^2 = \frac{1}{T} \sum_{t=1}^T (x_t - \bar{x})^2, \\
 \text{sample standard deviation} & \tilde{\sigma}_x.
 \end{array} \tag{8.1}$$

Note that an alternative definition of the variance would divide by $T - 1$ instead of T .¹

$$\begin{array}{ll}
 \text{sample variance (OLS)} & \hat{\sigma}_x^2 = \frac{1}{T-1} \sum_{t=1}^T (x_t - \bar{x})^2, \\
 \text{sample standard error} & \hat{\sigma}_x.
 \end{array} \tag{8.2}$$

¹The maximum likelihood estimate for a linear model gives $1/T$, while regression on a constant using OLS produces an unbiased estimate of the variance using $1/(T - 1)$.

8.3 Autocorrelation function (ACF) or covariogram

Define the sample autocovariances $\{\hat{c}_j\}$ of a series $x_t, t = \dots, T$:

$$\hat{c}_j = \frac{1}{T} \sum_{t=j+1}^T (x_t - \bar{x})(x_{t-j} - \bar{x}), \quad j = 0, \dots, T-1, \quad (8.3)$$

using the full sample mean $\bar{x} = \frac{1}{T} \sum_{t=1}^T x_t$. The variance $\tilde{\sigma}_x^2$ corresponds to \hat{c}_0 , see (8.1).

The sample autocorrelation function (ACF) is the series $\{\hat{r}_j\}$ where \hat{r}_j is the sample correlation coefficient between x_t and x_{t-j} . The length of the correlogram is specified by the user, leading to a figure which shows $(\hat{r}_1, \hat{r}_2, \dots, \hat{r}_s)$ plotted against $(1, 2, \dots, s)$ where for any j when x is any chosen variable:

$$\hat{r}_j = \hat{c}_j / \hat{c}_0, \quad j = 0, \dots, T-1. \quad (8.4)$$

The first autocorrelation, $\{\hat{r}_0\}$, is equal to one, and omitted from the graphs.

The asymptotic variance of the autocorrelations is $1/T$, so approximate 95% error bars are indicated at $\pm 2T^{-1/2}$ (see e.g. Harvey, 1993, p.42).

If there are missing values in the data series, the correlogram uses all valid observation as follows: the mean is computed for all valid observations; terms where x_t or x_{t-j} are missing are omitted from the summation in (8.3). Consequently, the same procedure is used in (8.4) and any other derived functions such as the spectrum and periodogram.

8.4 Partial autocorrelation function (PACF)

Given the sample autocorrelation function $\{\hat{r}_j\}$ the partial autocorrelations are computed using Durbin's method as described in Golub and Van Loan (1989, §4.7.2). This corresponds to recursively solving the Yule–Walker equations. For example, with autocorrelations, $\hat{r}_0, \hat{r}_1, \hat{r}_2, \dots$, the first partial correlation is $\hat{\alpha}_0 = 1$ (omitted from the graphs). The second, $\hat{\alpha}_1$, is the solution from

$$\begin{pmatrix} \hat{r}_0 \\ \hat{r}_1 \end{pmatrix} = \begin{pmatrix} \hat{r}_0 & \hat{r}_1 \\ \hat{r}_1 & \hat{r}_0 \end{pmatrix} \begin{pmatrix} \hat{\alpha}_0 \\ \alpha_1 \end{pmatrix},$$

et cetera.

8.5 Correlogram

The sample correlogram plots the series $\{\hat{r}_j^*\}$ where \hat{r}_j^* is the correlation coefficient between x_t and x_{t-j} . The length of the correlogram is specified by the user, leading to a figure which shows $(\hat{r}_1^*, \hat{r}_2^*, \dots, \hat{r}_s^*)$ plotted against $(1, 2, \dots, s)$ where for any j when

x is any chosen variable:

$$\hat{r}_j^* = \frac{\sum_{t=j+1}^T (x_t - \bar{x}_0)(x_{t-j} - \bar{x}_j)}{\sqrt{\sum_{t=j+1}^T (x_t - \bar{x}_0)^2 \sum_{t=j+1}^T (x_{t-j} - \bar{x}_j)^2}}. \quad (8.5)$$

Here $\bar{x}_0 = \frac{1}{T-j} \sum_{t=j+1}^T x_t$ is the sample mean of x_t , $t = j+1, \dots, T$, and $\bar{x}_j = \frac{1}{T-j} \sum_{t=j+1}^T x_{t-j}$ is the sample mean of x_{t-j} , so that \hat{r}_j^* corresponds to a correlation coefficient proper.

Note the difference with the definition of the sample autocorrelations $\{\hat{r}_j\}$ in (8.4) above. This difference tends to be small, and vanishes asymptotically, provided the series are stationary. But, as argued in Nielsen (2006), the correlogram provides a better discrimination between stationary and non-stationary variables: for an autoregressive value of one (or higher), the correlogram declines more slowly than the ACF.

When the correlogram is selected for plotting, the PACF is based on the correlogram values, rather than the ACF. Handling of missing values is the same as for the ACF.

8.6 Cross-correlation function (CCF)

The sample cross-correlation function (CCF) graphs the correlation between a series x_t and the lags of another series y_t :

$$\hat{r}_j^{xy} = \frac{\sum_{t=j+1}^T (x_t - \bar{x})(y_{t-j} - \bar{y})}{\sum_{t=1}^T (x_t - \bar{x}) \sum_{t=1}^T (y_t - \bar{y})} = \frac{\hat{c}_j^{xy}}{(\hat{c}_0^x \hat{c}_0^y)^{1/2}}, \quad j = 0, \dots, T-1, \quad (8.6)$$

using the full sample means \bar{x} and \bar{y} .

The CCF between x_t and y_t is different from that between y_t and x_t : $\hat{r}_j^{xy} \neq \hat{r}_j^{yx}$. Both are plotted in the same graph.

8.7 Periodogram

The periodogram is defined as:

$$\begin{aligned} p(\omega) &= \frac{1}{2\pi} \sum_{j=-T+1}^{T-1} \hat{c}_{|j|} e^{-i\omega j} = \frac{1}{2\pi} \sum_{j=-T+1}^{T-1} \hat{c}_{|j|} \cos(j\omega) \\ &= \frac{\hat{c}_0}{2\pi} \sum_{j=-T+1}^{T-1} \hat{r}_{|j|} \cos(j\omega), \\ &\text{for } \omega = 0, 2\pi/T, 4\pi/T, \dots, (\text{int}(T/2)2\pi)/T. \end{aligned} \quad (8.7)$$

Note that $p(0) = 0$.

When the periodogram is plotted, only frequencies greater than zero and up to π are used. Moreover, the x -axis, with values $0, \dots, \pi$, is represented as $0, \dots, 1$. So, when $T = 4$ the x coordinates are 0.5, 1 corresponding to $\pi/2, \pi$. When $T = 5$, the x coordinates are 0.4, 0.8 corresponding to $2\pi/5, 4\pi/5$.

However, when the periodogram is evaluated using the Algebra function periodogram, it is evaluated at all frequencies up to 2π , resulting in T periodogram values. For $T = 4$ these are evaluated at $0, \pi/2, \pi, 3\pi/2$. For $T = 5$: $0, 2\pi/5, 4\pi/5, 6\pi/5, 8\pi/5$. The frequencies are returned in the last argument of periodogram.

8.8 Spectral density

The estimated spectral density is a smoothed function of the sample autocorrelations $\{\hat{r}_j\}$, defined as in (8.4). The sample spectral density is then defined as:

$$\hat{s}(\omega) = \frac{1}{2\pi} \sum_{j=-(T-1)}^{T-1} K(j) \hat{r}_{|j|} \cos(j\omega), \quad 0 \leq \omega \leq \pi, \quad (8.8)$$

where $|\cdot|$ takes the absolute value, so that, for example, $\hat{r}_{|-1|} = \hat{r}_1$. The $K(\cdot)$ function is called the *lag window*. OxMetrics uses the Parzen window:

$$\begin{aligned} K(j) &= 1 - 6 \left(\frac{j}{m}\right)^2 + 6 \left|\frac{j}{m}\right|^3, & \left|\frac{j}{m}\right| \leq 0.5, \\ &= 2 \left(1 - \left|\frac{j}{m}\right|\right)^3, & 0.5 \leq \left|\frac{j}{m}\right| \leq 1.0, \\ &= 0, & \left|\frac{j}{m}\right| > 1. \end{aligned} \quad (8.9)$$

We have that $K(-j) = K(j)$, so that the sign of j does not matter ($\cos(x) = \cos(-x)$). The \hat{r}_j s are based on fewer observations as j increases. The window function attaches decreasing weights to the autocorrelations, with zero weight for $j > m$. The parameter m is called the *lag truncation parameter*. In OxMetrics, this is taken to be the same as the chosen length of the correlogram. For example, selecting $s = 12$ (the with length setting in the dialog) results in $m = 12$. The larger m , the less smooth the spectrum becomes, but the lower the bias. The spectrum is evaluated at 128 points between 0 and π . For more information see [Priestley \(1981\)](#) and [Granger and Newbold \(1986\)](#).

8.9 Histogram, estimated density and distribution

Given a data set $\{x_t\} = (x_1 \dots x_T)$ which are observations on a random variable X . The range of $\{x_t\}$ is divided into N intervals of length h with h defined below. Then the proportion of x_t in each interval constitutes the histogram; the sum of the proportions is unity on the scaling in OxMetrics. The density can be estimated as a smoothed function of the histogram using a normal or Gaussian kernel. This can then be summed ('integrated') to obtain the estimated cumulative distribution function (CDF).

Denote the actual density of X at x by $f_X(x)$. A non-parametric estimate of the density is obtained from the sample by:

$$\widehat{f_X(x)} = \frac{1}{Th} \sum_{t=1}^T K\left(\frac{x - x_t}{h}\right), \quad (8.10)$$

where h is the *window width* or smoothing parameter, and $K(\cdot)$ is a *kernel* such that:

$$\int_{-\infty}^{\infty} K(z) dz = 1.$$

OxMetrics sets:

$$h = 1.06\tilde{\sigma}_x/T^{0.2}$$

as a default, and uses the standard normal density for $K(\cdot)$:

$$K\left(\frac{x - x_t}{h}\right) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x - x_t}{h}\right)^2\right]. \quad (8.11)$$

The default window width is used when the dialog option is set to Default bars. Alternatively, it is possible to directly choose the number of bars, in which case, the more bars, the less smooth the density estimate will be. The histogram uses the whole sample, skipping over missing values.

$\widehat{f_x(x)}$ is usually calculated for 128 values of x , but since direct evaluation can be somewhat expensive in computer time, a fast Fourier transform is used (we are grateful to Dr. Silverman for permission to use his algorithm). The estimated CDF of X can be derived from $\widehat{f_x(x)}$; this is shown with a standard normal CDF for comparison. An excellent reference on density function estimation is [Silverman \(1986\)](#).

8.10 Regression lines and smooths

The scatter plots option allows for various types of regression lines to be drawn. A single regression line draws the fitted values from the OLS estimates $\hat{\alpha}$ and $\hat{\beta}$ in

$$\hat{y}_t = \hat{\alpha} + \hat{\beta}x_t, \quad t = 1, \dots, T,$$

assuming that y_t and x_t are the selected variables. The distinction between sequential and recursive regression lines is easily explained using three lines. Divide the sample in three parts: $1, \dots, T/3$, $T/3 + 1, \dots, 2T/3$ and $2T/3 + 1, \dots, T$, then:

sequential	line 1	$\hat{y}_t = \hat{\alpha}_1 + \hat{\beta}_1 x_t, \quad t = 1, \dots, T/3,$
	line 2	$\hat{y}_t = \hat{\alpha}_2 + \hat{\beta}_2 x_t, \quad t = T/3 + 1, \dots, 2T/3,$
	line 3	$\hat{y}_t = \hat{\alpha}_3 + \hat{\beta}_3 x_t, \quad t = 2T/3 + 1, \dots, T,$
recursive	line 1	$\hat{y}_t = \hat{\alpha}_1 + \hat{\beta}_1 x_t, \quad t = 1, \dots, T/3,$
	line 2	$\hat{y}_t = \hat{\alpha}_4 + \hat{\beta}_4 x_t, \quad t = 1, \dots, 2T/3,$
	line 3	$\hat{y}_t = \hat{\alpha} + \hat{\beta} x_t, \quad t = 1, \dots, T.$

Some examples were given in [Chapter 5](#). The regression line is fitted over the whole sample, skipping over missing values inside sample.

8.10.1 Kernel smooth

As a first step towards non-parametric regression, we start with a scatter plot of y_t and x_t , with x_t along the horizontal axis. Divide the x -axis in N intervals, and compute the average y -value in each interval. The resulting step function provides a non-parametric regression line: to compute $\hat{y}(x_0)$ we locate the interval on which x_0 falls, and use the mean of y in that interval as \hat{y} . An obvious drawback is that, if x_0 is towards the edge of an interval, most of the observations are on the other side. Solve this by centring the interval in x_0 . This still gives as much weight to points far away as to points nearby. In line with the smoothing functions (kernels) in the spectrum and density estimates, we use a density function to take a weighted average.

The non-parametric estimate of y is obtained from

$$\hat{y}_h(x) = \left(\sum_{t=1}^T K\left(\frac{x - x_t}{h}\right) \right)^{-1} \left(\sum_{t=1}^T K\left(\frac{x - x_t}{h}\right) y_t \right), \quad (8.12)$$

where, as for (8.10), h is the window width, *bandwidth* or smoothing parameter, and $K(\cdot)$ is a kernel which integrates to 1. OxMetrics uses the Epanechnikov kernel:

$$\begin{aligned} K(u) &= \frac{3}{4} (1 - u^2), & |u| \leq 1, \\ &= 0, & |u| > 1, \end{aligned} \quad (8.13)$$

which is optimal in a certain sense. The optimal bandwidth, equivalent to the default chosen for the non-parametric density estimate is:

$$h = 0.75 \tilde{\sigma}_x / T^{0.2}.$$

The function $\hat{y}_h(\cdot)$ is computed at 128 points. As $h \rightarrow 0$, $\hat{y}_h(\cdot) \rightarrow \bar{y}$: the sample mean is the fitted value for any x . On the other hand, if $h \rightarrow \infty$ the interval goes to zero, and we fit the nearest corresponding y -value so that each data point is picked up exactly. Note, however, that the $\hat{y}(\cdot)$ function is evaluated at the T data points x_t (which is time in the absence of x). [Härdle \(1990\)](#) is a general reference on the subject of non-parametric regression.

There are three ways of specifying the bandwidth:

- Number of parameters

This specifies the equivalent number of parameters (approximately comparable to the number of regressors used in a linear regression). The default is

$$\frac{3}{4} \left[\frac{(T-1)}{12} \right]^{1/2} T^{-0.2}.$$

- Set

Sets the bandwidth directly.

- Automatic

Chooses the bandwidth by generalized cross validation (GCV). We find that choosing bandwidth using GCV or cross validation (CV) tends to undersmooth.

Equation (8.14) below indicates how GCV is computed.

The kernel smooth is *not* computed using a Fourier transform, but directly, so can be slow for large T . The smooth is fitted over the whole sample skipping over missing values inside sample, which are estimated by the fit from the smooth.

One drawback of this smooth is that, for trending lines, it behaves counter intuitively at the edges. Consider, for example, the left edge of the kernel smooth of a variable which is upwardly trending. Since the smooth starts as a moving average of points which are only to the right, and hence mainly higher, the left edge will have a J shape, starting above what would be fitted by eye.

8.10.2 Spline smooth

A spline is another method for smoothing a scatter plot. Consider a plot of y_t , against x_t , and sort the data according to x : $a < x_{[1]} < \dots < x_{[T]} < b$. In a spline model, the sum of squared deviations from a function g is minimized, subject to a roughness penalty:

$$\min \sum_{t=1}^T [y_t - g(x_{[t]})]^2 + \alpha \int_a^b [g''(x)]^2 dx.$$

OxMetrics uses a *natural cubic spline*, which is cubic because the function g is chosen as a third degree polynomial, and natural because the smooth is a straight line between a and $x_{[1]}$ and between $x_{[T]}$ and b . This avoids the end-point problem of the kernel smooth of the previous section. Two good references on splines and nonparametric regression are [Green and Silverman \(1994\)](#) and [Hastie and Tibshirani \(1994\)](#).

The α parameter is the bandwidth: the smaller α , the lower the roughness penalty, and hence the closer the smooth will track the actual data.

The spline is evaluated at all the data points, where missing y values are estimated by the fit from the smooth. The spline procedure handles ties in the x variable. The algorithm used to compute the spline is of order T , and discussed extensively in [Green and Silverman \(1994, Chs.2,3\)](#).

For evenly-spaced data (e.g. scatter plot against time), the algorithm involves a Toeplitz matrix. This illustrates the closeness of a natural cubic spline to the Hodrick–Prescott filter which is popular in macro-economics:

	diagonal	2nd diag	3rd diag	remainder
Hodrick–Prescott	$6\alpha + 1$	-4α	α	0
spline	$6\alpha + 2/3$	$-4\alpha + 1/6$	α	0

The only difference is in this Toeplitz matrix. Since the Hodrick–Prescott filter uses $\alpha = 1600$ (for quarterly data), the smoothers from both methods are virtually identical.²

There are three ways of specifying the bandwidth:

²Some articles describe computation of the Hodrick–Prescott filter as involving inversion of a $T \times T$ matrix. This is clearly not the case. It is avoided in the Reinsch algorithm used for the general scatter-plot smooth, and more obviously in the evenly spaced case, where all that is required is the solution of a linear system involving a banded Toeplitz matrix. Alternatively, the Kalman filter may be used, as in [Koopman, Harvey, Doornik, and Shephard \(2013\)](#).

- Number of parameters

This specifies the equivalent number of parameters, k_e , (approximately comparable to the number of regressors used in a linear regression). The default is

$$\frac{3}{4} \left[\frac{(T-1)}{12} \right]^{1/2} T^{-0.2}.$$

- Set

Sets the bandwidth directly, with the default of 1600 corresponding to the Hodrick–Prescott filter for quarterly data (see §10.5.5.1 for other frequencies).

- Automatic

Chooses the bandwidth by generalized cross validation (GCV). We find that choosing bandwidth using GCV or cross validation (CV) tends to undersmooth. The GCV criterion is computed as:

$$GCV(\alpha) = T \left(\frac{RSS}{T - 1.25k_e + 0.5} \right). \quad (8.14)$$

We have adopted GCV instead of CV because a very good fit at one point could dominate the CV criterion.

8.11 QQ plot

Draws a QQ plot. The variable would normally hold critical values which are hypothesized to come from a certain distribution. This function then draws a cross plot of these observed values (sorted), against the theoretical quantiles (the horizontal axis). The 45° line is drawn for reference (the closer the cross plot to this line, the better the match). The QQ plot line is drawn over the whole sample, skipping over missing values.

The following distributions are supported:

- $\chi^2(df_1)$,
- $F(df_1, df_2)$,
- $N(0, 1)$,
- $t(df_1)$,
- Uniform(0, 1), resulting in a Quantile plot.

The normal QQ plot includes the pointwise asymptotic 95% standard error bands, as derived in Engler and Nielsen (2009) for residuals of regression models (possibly autoregressive) with an intercept. Here the normal QQ plot is created for the standardized data, but drawn on the original scale. The standard errors are given by the square root of:

$$T^{-1} \left(\frac{\Phi(z)(1 - \Phi(z))}{\phi(z)^2} - z^2/2 - 1 \right),$$

where z denotes the coordinates of the reference distribution (the standard normal), Φ is the standard normal cdf, and ϕ the density.

8.12 Box plot

A box plot shows the distribution of a variable in terms of its quartiles, labelled Q_1 , Q_2 , Q_3 (the 25%, 50% and 75% quartiles). Define the interquartile range as $IQR = 1.5(Q_3 - Q_1)$. The box plot consists of the following elements:

- a box, with horizontal lines at Q_1 , Q_2 (the median) and Q_3 ;
- a vertical line from $Q_1 - IQR$ to $Q_3 + IQR$ (omitted inside the box);
- individual observations: all observations outside the $(Q_1 - IQR, Q_3 + IQR)$ range, plus the two observations on either end which just fall inside this range.

For an example see [Figure 4.6](#) on page 50. The box plot uses the whole sample, skipping over missing values.

8.13 Exponentially-weighted moving average (EWMA)

A simple exponentially-weighted moving average of a series $y_t, t = 1, \dots, T$, is defined as:

$$m_t = \alpha y_t + (1 - \alpha) m_{t-1}, \quad t = 2, \dots, T, \quad (8.15)$$

using $m_1 = y_1$. The resulting series z_t is $z_t = m_t$, except when any y_{t+h} is missing for $h \geq 1$, in which case it is replaced by forecasts $z_{t+h} = m_t$.

An extended version introduces a slope coefficient β , and is also called Holt's method:

$$\begin{aligned} m_t &= \alpha y_t + (1 - \alpha) (m_{t-1} + b_{t-1}), \\ b_t &= \beta (m_t - m_{t-1}) + (1 - \beta) b_{t-1}. \end{aligned}$$

In this case $m_1 = y_1, m_2 = y_2, b_2 = y_2 - y_1$; when $\beta = 0$ the EWMA (8.15) is used. The resulting series z_t is $z_t = m_t$, except when any y_{t+h} is missing for $h \geq 1$, in which case it is replaced by forecasts $z_{t+h} = m_t + h b_t$.

Back-substitution in the EWMA without slope coefficient shows that:

$$m_t = (1 - \alpha)^{t-1} m_1 + \alpha \sum_{s=0}^{t-1} (1 - \alpha)^s y_{t-s}.$$

See e.g. [Harvey \(1993, Ch. 5\)](#), or [Makridakis, Wheelwright, and Hyndman \(1998, Ch. 4\)](#).

8.14 Exponentially-weighted moving correlation

The exponentially weighted moving sample correlation of two series y_t and x_t , both having expectation zero, is defined as:

$$r_t = \frac{p_t(x, y)}{[p_t(x, x)p_t(y, y)]^{1/2}}, \quad t = 3, \dots, T$$

where

$$p_t(x, y) = \sum_{s=0}^{t-1} \lambda^s x_{t-s} y_{t-s}, \quad t = 1, \dots, T,$$

and r_1, r_2 are set to missing values. When any y_{t+h} or x_{t+h} is missing r_{t+h} is set to r_t .

Premultiplying the numerator and denominator of r_t by $1 - \lambda$, shows that $m_t^* = (1 - \lambda)p_t(x, y)$ is the EWMA of the cross-product with $\alpha = 1 - \lambda$ and $m_1 = 0$. This does not work for $\lambda = 1$.

Chapter 9

OxMetrics file formats

9.1 OxMetrics data files (.oxdata)

This most convenient way of storing data on disk when using OxMetrics: it is compact and fast to load.

An oxdata file is a zip file that holds an .in7 file and a .bn7 file. These can also be handled as two separate files.

Note The oxdata format was introduced with OxMetrics 9.

9.2 OxMetrics data files (.in7/.bn7)

The information file, with the .in7 extension, holds the information about the data set. The information file is accompanied by a binary data file. The binary data file holds the actual data, and has the .bn7 extension. The .in7/.bn7 combination is convenient because information on sample size etc. is automatically stored. The drawback over .oxdata is that the two files need to be kept together.

Loading of very large .in7/.bn7 files is significantly faster than loading .xlsx or .csv files.

Note The .in7/.bn7 format is essentially as it was in PcGive versions 7 and 8. The difference is that spaces are now allowed in variable names, and the introduction of date and choice variables. Hidden variables and generations of variables are no longer supported.

9.2.1 The .in7 file format

The .in7 file is a text file that stores the metadata. As an example, see [Table 9.1](#), which lists the data.in7 file.

The following items are found in the .in7 file:

Table 9.1 data.in7: information file of artificial data set

pcgive 700							
data data.bn7							
;Tutorial Data Set:4 equation model with oil shock for PcGive							
;October 1985							
>CONS	1953	1	1992	3	4	32 data	10-04-1992 13:20:38.33
; Artificial consumption variable							
>INC	1953	1	1992	3	4	1336 data	10-04-1992 13:20:38.33
; Artificial income variable							
>INFLAT	1953	1	1992	3	4	2640 data	10-04-1992 13:20:38.33
; Artificial inflation variable							
>OUTPUT	1953	1	1992	3	4	3944 data	10-04-1992 13:20:38.33
; Artificial output variable							

Identification The first line must be the file identification followed by a version number. Currently the identification is pcgive and the version is 700. This will change if new features are added to the format.

Data file name The second line specifies the type of file (data), and gives the name of the data file corresponding to the information file. Here it is a binary file, called data.bn7. The .bn7 extension indicates that this file is a binary file. If the specified file has the .dat extension, it is assumed to be an ASCII (that is, human-readable) file, ordered by variable, see §9.2.4.

When xxx.in7 is used, and if xxx.bn7 exists in the same directory, the data statement is ignored. Else, the data part must specify the .bn7 file. This behaviour was introduced with version 1.0 of OxMetrics, and allows renaming a .in7/.bn7 pair without having to edit the .in7 file.

Comment Lines starting with a semicolon are comment lines, so the third line in the example is a comment line. All comment lines following the data statement are considered to document the whole file; those following a variable belong to the variable.

Variable definition The remaining entries are variable definitions. The first character (> or -) indicates the status and is described below.

All lines starting with a semicolon that follow a variable definition are comments on the variable. This documentation can be edited in the database window by selecting the variable name. When saving the database in the OxMetrics format, the documentation is preserved.

There is one line defining each variable, which holds the following information:

>CONS	1953	1	1992	3	4	32	data	10-04-1992	13:20
		↓			↓	↓	↓	↓	↓
↓	sample period		frequency		address	group		date & time	type
name, preceded by status: active (>) or deleted (-)									

Status If the status is a minus sign, the variable is deleted (but physically still present).

A deleted variable will not be loaded into OxMetrics.

Variable name Maximum length is 64 characters. The variable name is enclosed in double quotes if it has a space, but leading and trailing spaces, as well as double

quotes in the name are removed. So a name like CONS-INC, which would not be a valid variable name in any programming language, is allowed.

Generation The generation is not explicitly stored in the information file, but derived from the variable name. If there is more than one entry with an identical name (also written in identical case, since Cons and CONS are different), then the first one has generation 0, the second generation 1, etc. Only the highest generation (most recent version) will be loaded into OxMetrics.

Sample period The sample period consists of four numbers: *startyear*, *startperiod*, *endyear* and *endperiod*. Each variable in the data file can have a different sample period. When reading such a file into OxMetrics with a still empty database, the sample containing all data will be used (missing observations will get the missing value).

Frequency Variables with different frequencies can be stored, but only one frequency at a time can be loaded into OxMetrics (although several databases can be open simultaneously with different frequencies).

Address Gives the physical address of the variable in the binary data file. So the first observation of CONS starts at byte 32 in data.bn7, the second at byte 40 etc. (data is stored in double precision, taking eight bytes per observation). The 32 bytes preceding every variable are a separate information block, from which a rudimentary information file can be reconstructed. If necessary, this is a 64-bit integer.

Group Variables in a file can be grouped together by the group name. No spaces are allowed in the group name.

Date and time fields These give the date and time at which the variable was written. No spaces are allowed in either field.

Variable type The optional entry after the date and time fields gives the variable type:

(empty) By default the type field is empty, indicating a normal numerical variable.

date Indicates that this is a variable of type Date.

choice[#] Signals a choice variable. The # is an integer that specifies the number of labels. The next line enumerates the labels, starting with |, and also using | as a separator. The choice list can be written over multiple lines, but each line must start with a |. As a consequence, the | character should not be used in the choice label.

9.2.2 The .bn7 file format

The .bn7 file is a binary file, and thus not human readable. It holds the actual data.

Each variable starts with a 32 bit signature, followed by the data in the form of 8 byte doubles. The doubles have the standard IEEE form and are stored in little endian format (the default on PC platforms).

The 32 bit signature consists of:

bytes	C type	description
0-7	double	marker (double: 9999E99)
8-11	int	number of observations (T)
12-23	char[12]	variable name (truncated or padded with null characters)
24-25	short int	frequency
26-27	short int	starting period
28-29	short int	starting year
30-31	short int	not used

This signature is followed by $8 \times T$ bytes. The signature is always written, but never read, because all the relevant information is obtained from the `.in7` file, including the starting address of each variable. Note that `int` is a 32-bit signed integer, and `short int` a 16-bit signed integer.

Missing values are encoded as a IEEE defined NaN (*not a number*); `-9999.99` is *not* recognized as missing (but was used for this purpose in early versions of PC-GIVE).

9.2.3 Limitations

The amount of data in the `.in7/.bn7` files is limited by available disk space, and, more strictly, by available memory. In a 32-bit operating system, the maximum database size is about $2^{28} \approx 2.68 \times 10^8$ elements, equivalent to 2 Gigabyte of memory space. For a 64-bit operating system, the maximum number of observations in a variable is $2^{31} \approx 2.1 \times 10^9$. This is usually much more than can be loaded at one time in OxMetrics. C source code and object files to read `.in7/.bn7` files are available on request. Ox can read and write `.in7/.bn7` on all computer platforms.

9.2.4 The information and ASCII data files (`.in7/.dat`)

An information file may describe a human-readable data file which is ordered by variable. This option is especially useful for the situation where OxMetrics is routinely used to process output generated by another program which cannot write binary data files. When saving data, OxMetrics will always create a binary data file with accompanying information file. To attach a human-readable file to an information file, the filename following the `data` statement in the `.in7` file should have the `.dat` extension (and the `.bn7` file must not exist, otherwise that file is read instead). In the example of [Table 9.1](#) the second line would read `data data.dat`. Then OxMetrics will assume that the data is stored in the human-readable (ASCII) data file `data.dat`. The data must be ordered by variable, and offending words are treated as missing values (see [§9.4](#)). Apart from the `data` statement, the organization of the `.in7` file is the same as in [§9.2](#). The value in the address field will be ignored, the date and time field may be missing.

9.3 Spreadsheet files (`.xlsx`, `.csv`)

OxMetrics can read and write the following spreadsheet files:

- Excel: .xlsx files;
This is the Excel 2007 (or newer) workbook file (Office Open xml).
- text-based: .csv (comma separated).
The .xls format is no longer supported by OxMetrics 9.

9.3.1 .xlsx files

OxMetrics can read an .xlsx file provided the following conventions are adopted:

- Ordered by observation (that is, variables are in columns).
- The data form a contiguous sample.
- The data are on the first sheet.
- The data matrix is a rectangular array, structured in one of the following formats:
 1. Unlabelled first column with dates, followed by labelled columns, one for each variable.

The first column either consists of proper calendar dates, or strings that represent a date. In the latter case, it should be of the form year–period, where the – can be any single character: for example, 1980–1 (or: 1980Q1 1980P1 1980:1 etc.; this character is ignored).

If the date column can be read as fixed frequency dating of observations, the sample will be set appropriately, and the date column will not appear as a variable.

If the first column cannot be interpreted as defining a fixed-frequency database, the column will be loaded with default variable name Svar1.

2. All columns are labelled.

An attempt will be made to read the first column as the dates. Regardless of the result, the column will appear as a variable in the database.

If string-based dates were used, the date variable is of type choice. Otherwise it will be of type date, in which case the database will appear as dated.

3. No column is labelled.

This is the same as if all columns are labelled with Svar1, Svar2,

If no dates are present, the database will have a fixed frequency of unity, starting from 'year' 1.

If these conventions are not adopted the file can still be read, but you might have to provide some additional information, such as frequency, start date and row of first observation. In the example, these are respectively 4, 1980 1, 2.

Variable types are determined automatically:

1. If a variable contains text fields, it will be interpreted as a choice variable. The ordering of the choice labels is determined from the comment field, see the example below.

If there is no comment field, numbers will be assigned consecutively to unique text fields. This means that these observations will be assigned an integer value that is not yet used in the variable. Identical text fields will have the same integer value.

Fields that have no label associated show their value, while missing values are shown with a dot.

When saving as `.xlsx`, the choice labels are saved to the comment field of the variable name.

- 2. A variable with dates will become a date variable. A date variable will interpret all observations as dates.
- 3. Otherwise, the variable has only numerical values (the default).

You can use `#N/A`, `#NA`, `.NaN`, `.`, `missing` or leave gaps for missing observations in the input spreadsheet file. When saving as a spreadsheet file, missing values are written as `#N/A`.

For example, the format for writing is (this is also the optimal format for reading):

	A	B	C	D
1		CONS	INFL	DUM
2	1980(1)	883	2.7	3
3	1980(2)	884	3.5	5
4	1980(3)	885	3.9	1
5	1980(4)	889	2.6	9
6	1981(1)	900	3.4	2

We shall give some examples to show what happens with slightly different files. The spreadsheet displayed on the left was created in Excel, and then subsequently loaded into OxMetrics, with the result displayed on the right.

In the first example the first column has dates, rather than text. There is also text in the third variable. OxMetrics detects that the dates correspond to a fixed frequency sequence (monthly in this case), and sets the sample correctly. Text fields are assigned unused integers, starting from zero, in the order they are found (separately for each variable). So `'text'` gets value 0, and `'abcd'` value 3.

	A	B	C	D	OxMetrics database			
1		aap	noot	mies		aap	noot	mies
2	Jan-80	1	#N/A	1	1980(1)	1	missing	1
3	Feb-80	2.3	12.436	2	1980(2)	2.3	12.436	2
4	Mar-80	9	13.786	text	1980(3)	9	13.786	text
5	Apr-80	4.125	9.456	abcd	1980(4)	4.125	9.456	abcd
6	May-80	1	#N/A	#N/A	1980(5)	1	missing	.

While this method of reading text works, it doesn't guarantee that the labels always appear in a user-defined order (so that `'text'` always corresponds to value 0). When saving this database from OxMetrics as `.xlsx` file, the choice labels are saved in the comment field of the variable. The description of the variable created in OxMetrics is also preserved there. Using `Choice Variable` for the description and saving, the file appears as follows in Excel:

	A	B	C	D	E	F
1		aap	noot	mies		Choice Variable
2	1980(1)		1 #N/A		1	
3	1980(2)		2.3	12.436	2	
4	1980(3)		9	13.786 text		<choices>
5	1980(4)		4.125	9.456 abcd		text abcd
6	1980(5)		1 #N/A	#N/A		</choices>
7						

In the second example the dates are valid, but column labels (that is, variable names) are missing. Also note that any non-numeric fields are converted to missing values.

	A	B	C	D	OxMetrics database				
1	1980(1)	1	missing	1		Svar1	Svar2	Svar3	Svar4
2	1981(1)	2.3		2	1980	1980(1)	1	missing	1
3	1982(1)	9	13.786	3	1981	1981(1)	2.3	missing	2
4	1983(1)	4.125	9.456	4	1982	1982(1)	9	13.786	3
					1983	1983(1)	4.125	9.456	4

Finally, both dates and column labels are missing:

	A	B	C	OxMetrics database			
1	1	#N/A	1		Svar1	Svar2	Svar3
2	2.3	12.436	2	1-1	1	missing	1
3	9	13.786	3	2-1	2.3	12.436	2
4	4.125	9.456	4	3-1	9	13.786	3
				4-1	4.125	9.456	4

9.3.2 .csv files

A comma-separated values file stores the spreadsheet in a human-readable text file with .csv extension. OxMetrics closely follows RFC 4180 (see the documentation at tools.ietf.org/html/rfc4180):

1. Each line represents a row of the spreadsheet, and all lines must have the same number of elements. OxMetrics allows \r, \r\n, or \n as end-of-line markers.
2. Each field is separated by a comma.
3. A field may be enclosed in double quotes ('quoted', e.g. "1.25") or not ('unquoted', e.g. 1.25). A field with multiple lines, comma or double quotes must be quoted. An embedded double quote is written as two consecutive double quotes. Spaces are considered part of a field, but spaces after a quoted field are skipped.

The following rules apply to the fields:

1. A numerical value must be unquoted and written with a period as the fractional separator.
2. The following unquoted fields are interpreted:

missing values	infinities
.	+.Inf
.NaN	.Inf
#N/A	-.Inf
#NA	
missing	

An empty field is also treated as a missing value.

3. Dates are recognized, provided they are unquoted and follow the same ISO standard as Ox.

4. A quoted field is interpreted as a text field.

These rules identify the spreadsheet, from which variable and sample information is extracted along the lines described in §9.3.1. The only difference is that an unlabelled first column, with other columns labelled, is loaded if it contains numerical data (even when recognized as defining a fixed frequency).

When reading, variables with non-numerical fields are converted to a choice variable, with the text as choice label. A choice variable is not properly preserved in the saved csv file: the label is written, but the numerical value is reconstructed from the order when loading. This may not correspond to the original order.

9.3.3 Zipped .csv files

When a csv file is zipped, say `test.csv` zipped into `test.csv.zip` or `test.zip`, it can be loaded directly into OxMetrics. A database can also be saved as a zipped csv, which is given the `.zip` extension.

9.3.4 Interpreted and uninterpreted .xlsx and .csv files

OxMetrics can use spreadsheet files for econometric modelling, because it imposes structure on the spreadsheet to define variables, their names, and the sample period. Dates are mapped to OxMetrics dates. Text fields, apart from variable names or date labels in the first column, are interpreted as labels for specific values, defining a choice variable. Because the underlying value of a choice is still numerical, it can be used to select an estimation sample by, create dummy variables, or use in a regression.

Another benefit of the adopted handling of choice variables is that they can be easily manipulated in Ox using the Database class. `Database::GetVarChoices` gets the list of choices as an array, while `Database::GetObsLabel` gets the label of an observation on a variable.

The Ox `loadsheet` function can be used to read an entire spreadsheet (`.xlsx` or `.csv`) into an array, without the interpretation that OxMetrics and Ox apply by default.

The Ox `loadmat` function can be used to read an entire numerical spreadsheet (`.xlsx` or `.csv`) into a matrix, either with or without the interpretation that OxMetrics and Ox apply to dates and strings. Without the interpretation all the strings are treated as missing values, and variables that only have missing values are dropped.

9.4 Data by observation (.dat)

As the name suggests, a human-readable (or text) data file is a file that can be read using a file viewer or editor. (A binary file cannot be read in this way.) Initial input of data to OxMetrics is often in this form, and then saved in the OxMetrics format (see §9.2). The default extension is `.dat`.

In this format, there is no information on sample size nor on data frequency. There may be information on variable names at the beginning of the file. The data set will be loaded as undated, which can be changed subsequently inside the program.

Each variable must have the same number of observations. Variables that have too short a sample have to be padded by missing values. Text following ; or // up to the end of the line is considered to be comment, and skipped. Data files must be ordered by observation (first observation on all variables, second observation on all variables, etc.) with one observation per line. The file may start with listing the variable names. Examples are:

//by observation	//with names
891 2.8 //1953 (1)	Cons Inflat
883 2.7 //1953 (2)	891 2.8 //1953 (1)
884 3.5 // etc.	883 2.7 //1953 (2)
891 2.8	884 3.5 // etc.
885 3.9	891 2.8
. 2.6	885 3.9
891 2.8	889 2.6

Missing values can be represented by a dot or .NaN.

9.5 Data with load info (.dat)

This data type can be read without intervention, unlike data by observation, which is unstructured. An example is:

```
>CONS 1953 1 1954 1 4
      890.449584960938      .      886.328796386719
      884.884704589844      885.254089355469

>INC 1953 1 1954 1 4
      908.212280273438      900.678527832031      899.795288085938
      898.481506347656      895.776916503906
```

The variable description starts with a >, followed by the variable name (no spaces are allowed), the sample and finally the frequency. This is similar to the description in the .in7 file. The description is followed by the actual data (so the data are always ordered by variable here). As before, text following ; or // up to the end of the line is considered to be comment, and skipped.

When loading a .dat file, OxMetrics will first see if it is a Gauss data file. If not, it will try to read it as a data file with load information. Finally, it will try a data file by observation.

9.6 Stata data file (.dta)

The .dta file is a format written by Stata. OxMetrics can read files written by Stata versions 4–6 as well as those defined as formats 113 to 117. There is no sample information read from the .dta file, so the frequency will be set to one, and the first observation will be at ‘year’ 1. This can be changed after loading the data, see §6.5.2.

OxMetrics can write a .dta file as a format-114 file, but dates, database comment and sample information is lost. Choice labels and variable comments are preserved.

9.7 Results file (.out)

The contents of text windows are written to the results file, using the default extension .OUT when saving the window content to disk.

9.8 Batch file (.fl)

The batch file stores a set of batch commands, and has the .FL extension. It is a normal text file. The batch language is explained in Chapter 11.

9.9 Algebra file (.alg)

An algebra file holds the algebra code, and will normally have the .ALG extension. It is a text file and can be edited. The algebra syntax is explained in Chapter 10.

9.10 Ox file (.ox)

Is used to store Ox programs (Doornik, 2021). When loaded in OxMetrics, with the Windows version of Ox installed, the file can be run by clicking on the Run button on the toolbar, or using the Modules/Run Default Module command (Ctrl+r).

9.11 Matrix file (.mat)

A matrix file holds a matrix in human readable form. The actual matrix contents is preceded by two integers which specify the matrix dimensions (number of rows and columns). A matrix file normally has the .mat extension. Lines starting with ; or // are treated as comments. An example of a matrix file is:

```
2 3          //<-- dimensions, a 2 by 3 matrix
//comment   //<-- a line of comment
1 0 0        //<-- first row of the matrix
0 1 .5       //<-- second row of the matrix
```

9.12 OxMetrics graphics file (.gwg)

This is the only graphics storage which can be read back into OxMetrics. The file is text, but direct editing will rarely be necessary because all graphics objects can be manipulated on screen. Figure 9.1 corresponds to the following .gwg file:

```
OxMetrics 11 1 1
paper(15000,7500)
box(1,640)
mode(1,0)
```

```

xvec(0,2,2)
{
  "CONS" 8
  < 890.4495849609375 886.5430297851563 886.3287963867188 884.8847045898438
    885.2540893554688 884.528076171875 884.4362182617188 884.3106079101563
  >
  "" 0
  1953.75 0.25;
  freq:4;
  symbol:1 4 160;
}
xvec(0,3,3)
{
  "INC" 8
  < 908.2122802734375 900.6785278320313 899.7952880859375 898.4815063476563
    895.7769165039063 894.830810546875 892.7406005859375 892.7684936523438
  >
  "" 0
  1953.75 0.25;
  freq:4;
  symbol:0 1 90;
}
axis(0,1)
{
  x;
  anchor2: 0;
  label:300 1000 60;
  grid:0 8 1;
  set:0 1 0 0 0 0 0;
  rot:0;
}
axis(0,1)
{
  y;
  anchor2: 0;
  label:300 1000 60;
  grid:0 8 1;
  set:0 1 0 0 0 0 0;
  rot:0;
}
area(0,0)
{
}
legend(0,1)
{
  365 46 0;
  set:300 1 2 1;
}
line(0,11,11)
{
  wset:1954.502949852507 884.0768348623853 1954.99133480826
    905.3899082568807 0;
}
text(0,1)
{
  9128 5619 10$$\theta_i$;
  set:0 390 0;
  wset: 1955.03558259587 904.0137614678901;
}

```

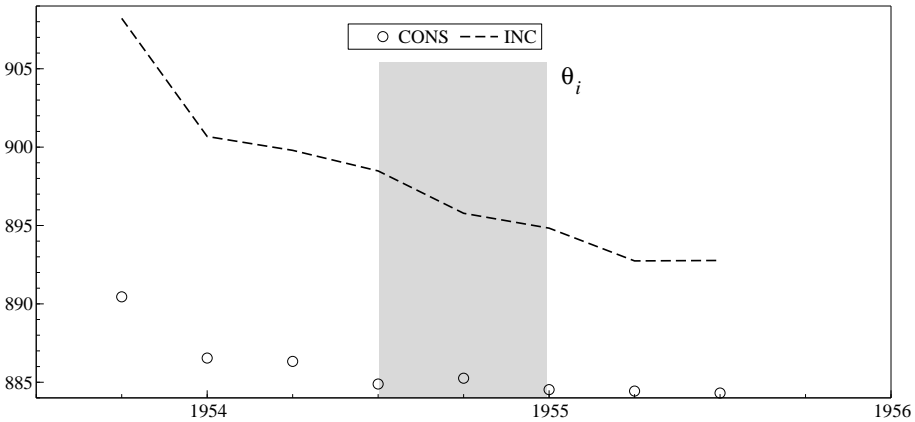


Figure 9.1 A simple graph

9.13 PDF file (.pdf)

A PDF file is a (partially) compressed binary file which can be handled by many PDF compatible programs. This is the preferred format for inserting a graph in LaTeX.

9.14 SVG file (.svg)

This is the Scalable Vector Graphics (SVG) format that is created by the World Wide Web Consortium. It is a vector format that is supported by all modern web browsers, and can easily be inserted in an html page. SVG files are the preferred format for inserted in Excel 2016 and Word 2016 (or newer).

An SVG file is a text file. OxMetrics saves in version 1 format, because Excel and Edge do not fully support higher SVG versions.

9.15 PostScript file (.eps)

A PostScript file is a text file which can be printed on any PostScript compatible printer, or on other printers using GhostScript. GhostScript (and its companion, GhostView for viewing files) can be obtained from the internet.

Direct printing can be achieved simply by copying the file to the PostScript printer. However, since the file is Encapsulated PostScript (EPS), it will not be centred on the page. Most popular word-processors can load EPS files.

Figure 9.1 corresponds to the following .EPS file (part of the actual graph is deleted from the listing):

```

%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 432 216
%%Creator: Ox (C) J.A. Doornik, 1994-2006.
%%Pages: 0

```

```

%%EndComments

%functions deleted ...

%lines
/Linetype0 [ ] def /Linewidth0 10 def
/Linetype1 [ ] def /Linewidth1 6 def
/Linetype2 [ ] def /Linewidth2 10 def
/Linetype3 [50 20] def /Linewidth3 10 def
/Linetype4 [ ] def /Linewidth4 10 def
/Linetype5 [35 35] def /Linewidth5 10 def
/Linetype6 [35 35] def /Linewidth6 20 def
/Linetype7 [35 35] def /Linewidth7 30 def
/Linetype8 [160 80] def /Linewidth8 10 def
/Linetype9 [160 80] def /Linewidth9 20 def
/Linetype10 [160 80] def /Linewidth10 30 def
/Linetype11 [ ] def /Linewidth11 20 def
/Linetype12 [ ] def /Linewidth12 30 def
/Linetype13 [ ] def /Linewidth13 50 def
/Linetype14 [ ] def /Linewidth14 10 def
/Linetype15 [ ] def /Linewidth15 10 def

%fonts
/F0 /Times-Roman ISOLatin1
/F1 /Symbol findfont definefont pop
/F2 /Times-Bold ISOLatin1
/F3 /Times-Italic ISOLatin1
/F4 /Times-BoldItalic ISOLatin1
/F5 /Helvetica ISOLatin1
/F6 /Courier ISOLatin1
/F7 /Times-Roman ISOLatin1
/F8 /Times-Roman ISOLatin1
/F9 /Times-Roman ISOLatin1

%colors
/BWG 16 array def
/Gray 16 array def
/RGB 16 array def
% first set all to black
0 1 15 { dup Gray exch 0 put RGB exch [0 0 0] put } for

% color 0: white <-- background colour
% color 1: black <-- foreground colour
BWG 0 0.00 put Gray 0 1.00 put RGB 0 [1.00 1.00 1.00] put
BWG 1 0.00 put Gray 1 0.00 put RGB 1 [0.00 0.00 0.00] put
BWG 2 0.00 put Gray 2 0.30 put RGB 2 [1.00 0.00 0.00] put
BWG 3 0.00 put Gray 3 0.11 put RGB 3 [0.00 0.00 1.00] put
BWG 4 0.00 put Gray 4 0.35 put RGB 4 [0.00 0.50 0.50] put
BWG 5 0.00 put Gray 5 0.41 put RGB 5 [1.00 0.00 1.00] put
BWG 6 0.00 put Gray 6 0.44 put RGB 6 [0.00 0.75 0.00] put
BWG 7 0.00 put Gray 7 0.45 put RGB 7 [0.50 0.50 0.00] put
BWG 8 0.20 put Gray 8 0.21 put RGB 8 [0.50 0.00 0.50] put
BWG 9 0.95 put Gray 9 0.95 put RGB 9 [1.00 1.00 0.50] put
BWG 10 0.79 put Gray 10 0.80 put RGB 10 [0.50 1.00 0.50] put
BWG 11 0.85 put Gray 11 0.85 put RGB 11 [0.50 1.00 1.00] put
BWG 12 0.70 put Gray 12 0.70 put RGB 12 [0.00 1.00 1.00] put
BWG 13 0.25 put Gray 13 0.25 put RGB 13 [0.25 0.25 0.25] put
BWG 14 0.50 put Gray 14 0.50 put RGB 14 [0.50 0.50 0.50] put
BWG 15 0.75 put Gray 15 0.75 put RGB 15 [0.75 0.75 0.75] put

```

```
%choose a ColorModel: 0=black/white; 1=black/white/gray; 2=gray; 3=color
/ColorModel 1 def

% /SetColor and /FillColor deleted ...

%init
save
1 setlinejoin
280 /F0 Font
10 setlinewidth
/Ltp Linetype1 def
/Lwd Linewidth1 def

/User { /ptX 0 def /ptY 0 def} def
%.PS uses full page printing
% select papersize: A4, Letter, User (in points: 72/inch)
% select orientation: 0 portrait mode, 1 landscape
% full page printing for .PS, else use EpsPage
%.EPS defers scaling/rotation to document

EpsPage
%A4 /Landscape 1 def FullPage

%Graph start
% rest deleted ...
%Graph end

restore
showpage
%%EOF
```

Sometimes minor changes need be made after saving the file, e.g.:

- Changing colour model.
 - 0 Black and white

All non-white colours are translated to black.
 - 1 Black, white, gray

Colour 0 is white, colours 1–8 are black, the remainder are translated to a grey value. One additional change to the line types is made: if line 2 and 3 are both solid, line 3 is changed to a dotted line.
 - 2 Gray levels

Colour 0 is white, colours 1 is black, the remainder are translated to a grey value.
 - 3 Color

Will result in a colour graph on a colour printer, else the result will be translated to grey levels inside the printer.

The statement `/ColorModel 1 def` sets the colour model.

The PostScript mode command on Edit Graph/Graph layout allows choosing a colour model to be used when saving the file.

- Changing line types. For example, to make line two twice as thick, and dashed, change


```
to /Linetype2 [ ] def /Linewidth2 15 def
to /Linetype2 [40 40] def /Linewidth2 20 def
```

The [40 40] defines the pattern: 40 units line, followed by a gap of 40 units. Note that line type 1 is the black line, used for axes. Line type two is the one used for the first vector. Line type 0 is usually invisible (white line on white background)

- Changing font sizes. Whenever a font is chosen, it is preceded by the font size, as in:

```
300 Font0
```

 which selects font 0 (set to Times-Roman) at size 300.
- Full page print.

As was pointed out before, when copying the EPS file to the printer, it will not fill the page. To print the graph in landscape filling a full A4 page, change

```
EpsPage
%A4 /Landscape 1 def FullPage
```

to

```
%EpsPage
A4 /Landscape 1 def FullPage
```

Change A4 to Letter for letter size, and use /Landscape 0 for portrait mode. These settings can be controlled in the PostScript Setup dialog. When saving as a .PS file, the default is full page. When loading the EPS file into a word-processor, that program will be able to resize the graph to any desired size.

9.16 PostScript file (.ps)

The difference with the .EPS file of the previous section is that the default size is full page, with settings controlled via the PostScript Setup dialog.

9.17 Enhanced meta file (.emf)

The Windows metafile format is used for storing the graph as displayed on screen. Since the graphs are drawn in vector form, the metafile will be a vector graph. EMF files can be read by many programs, for example Microsoft Word and Scientific Word, and edited further.

Chapter 10

Algebra Language

10.1 Introduction

OxMetrics is mostly menu-driven for ease of use. To add flexibility, certain functions can be accessed through entering commands. The syntax of these commands, which can be seen as small computer languages, is described in this and the next chapter. In addition, there is the Ox language, which is a more serious computer language, and described in a separate book.

10.2 Executing Algebra code

Algebra is a simple vector language, operating on only one type of object: the variables in the database. This object is manipulated as a whole, although it is possible to limit access to a subsample. The only valid statements are assignments and conditional assignments. Statements have to be terminated with a semicolon.

If an error occurs while executing the algebra code, the processing will be aborted, and control returned to OxMetrics. All successful statements up to that point will have been executed, with corresponding changes to the database. Take this into account when rerunning the corrected code.

There are three ways of running Algebra code, as discussed in the next three sections.

10.2.1 Calculator (Alt+c)

The calculator enables easy manipulation of the variables in the database, and is a convenient way to write algebra expressions.

The aim is to build a valid algebra expression in the expression window (without the assignment part and the terminating semi-colon). All successful transformations are logged in the Results window as Algebra code.

10.2.2 Algebra Editor (Alt+a)

The Tools/Algebra Editor command enables you to transform the database variables by typing mathematical formulae into an editor. The algebra code can be saved and reloaded. The Algebra editor also allows for choosing the database to which the code is applied.

10.2.3 Algebra from Results windows (Ctrl+a)

A text selection containing Algebra code can be run directly from that window using the Edit/Run as Algebra command (or Ctrl+a as short cut). The code is applied to the currently active database. The current database is changed by setting focus to that database, or using any of the drop down boxes (calculator, algebra editor, graphics), or using the `usedata()` batch command.

10.2.4 Algebra from a Batch file

Algebra code can be specified in a Batch program, see Ch. 11, and specifically §11.5.1. An Algebra file can also be loaded into a batch file (§11.5.18).

10.3 Syntax of Algebra language

10.3.1 Variables and variable names

Names are made up of letters and digits. The first character must be a letter or an underscores (.). Subsequent characters must be a letter, underscore, digit, or one of the four symbols \$, #, %, @. A # can be followed by a date, so `II#1990(1)` and `II#2020-12-31` are valid names. If a variable name does not follow these rules, it must be enclosed in double quotes in Algebra expressions.

Valid names are `CONS`, `cons`, `cons_1`, `_a_1_b`, etc. Examples of invalid names are `ΔCONS`, `1-CONS`, `log(X)`, etc. Invalid names may be used in Algebra when enclosed in double quotes, so

```
"log(X)" = log(X);
```

is valid as long as the variable called `X` exists. Algebra is case-sensitive: `CONS` and `cons` refer to different variables. When you create a new variable through an assignment operation, it is immediately added to the database, and initialized to missing values. If necessary the database name will be truncated (longer names are allowed in algebra than in the database).

10.3.2 Comment

Anything between `/*` and `*/` is considered comment. Note that this comment cannot be nested. So

```
aap = CONS + 1; /* comment /* nested comment */ */
```

leads to a syntax error. Everything following `//` up to the end of the line is also comment.

10.3.3 Constants

A constant is a number which is used in an expression. Examples are 1, 1.2, .5, -77000, -.5e-10 and 2.1E-12.

10.3.4 Algebra operators

10.3.4.1 Arithmetic operators

The binary arithmetic operators are `^`, `*`, `/`, `+`, `-`. The `^` is the power operator; `CONS^2` raises the variable `CONS` to the power two. All computations are double precision floating point arithmetic. The precedence order is: `^`, unary `+` and unary `-`, then `*`, `/`, and finally `+`, `-`. An example of unary minus is: $x = -1$. *Any arithmetic operation involving missing values returns a missing value.*

10.3.4.2 Relational and logical operators

The relational operators are `<`, `<=`, `>`, `>=`, standing for ‘less’, ‘less or equal’, ‘greater’, ‘greater or equal’. The equality operators are `==` and `!=`, ‘is equal to’ and is ‘not equal to’. These are ranked below the relational operators in precedence. Ranked yet lower are logical AND (`&&`) and logical OR (`||`). The unary negation operator `!` has the highest precedence. An example of unary negation is:

```
new = !season();
```

`new` will have a 0 where the `season` function returns a 1, and vice versa. Boolean arithmetic is also done in floating point. The numeric result is 1. for an expression that evaluates to true, and 0. for one that is false. An expression with value 0 is false, whereas a value that is not 0 is true.

Missing values are treated as follows:

a op b	result when a and/or b is a missing value
<code><</code> , <code><=</code> , <code>></code> , <code>>=</code>	FALSE if a and/or b is missing
<code>==</code>	TRUE if both a and b are missing
<code>!=</code>	TRUE if either a or b is missing (but not both)
<code>&&</code> , <code> </code>	no special treatment of missing values

10.3.4.3 Algebra operator precedence

Table 10.1 lists the operator precedence, with the highest precedence at the top of the table.

10.3.5 Assignment statements

The assignment operator is the `=` symbol. Assignment statements have to be terminated by a semicolon (`;`). If the variable to which a result is assigned does not yet exist, it is

Table 10.1 Algebra operator precedence

Symbol	Description	Associativity
\wedge	Power	Left to right
!	Logical NOT	Right to left
-	Unary minus	
+	Unary plus	
*	Multiply	Left to right
/	Divide	
+	Add	Left to right
-	Subtract	
<	Less than	Left to right
<=	Less than or equal to	
>	Greater than	
>=	Greater than or equal to	
==	Equal	Left to right
!=	Not equal	
&&	Logical AND	Left to right
	Logical OR	Left to right

created and added to the database. Otherwise the existing variable is overwritten. Note that assignment expressions are vector expressions: all observations will be overwritten (it is possible to restrict assignment to a subsample using the `insample` function). Some valid assignment statements are:

```
cons2 = 2 * CONS - 1;
cons3 = CONS / 3 - (cons2 + 5) * 1.55;
Seasonal = season();
```

The last statement constructs a variable called `Seasonal`, which is 1 in period 1, and 0 otherwise. This variable will be used in further examples in this chapter.

Take care not to confuse the `=` and `==` symbols: `=` assigns, while `==` compares. Parentheses may be used in expressions in the usual way; in the example above `(cons2 + 5) * 1.55` evaluates to `cons2 * 1.55 + 5 * 1.55`.

10.3.6 Conditional assignment statements

The conditional assignment expression has the following form:

TestExpression ? *TrueExpression* : *FalseExpression*

First the *TestExpression* is evaluated; if it is true (not 0.), then *TrueExpression* will be evaluated, else *FalseExpression*. Let us consider an example involving the `Seasonal` variable created above, which is 1 in the first quarter, and 0 in the other quarters. The statement

```
new = Seasonal ? CONS : 0;
```

can be read as follows: the variable `new` will get the value of `CONS` when `Seasonal` is true (that is, when `Seasonal` is not 0., so in the first quarter of each year), else it will get the value 0. In this case, the same result could have been reached with:

```
new = Seasonal * CONS;
```

However, if we had used a seasonal with the value 2 in the first quarter, and the rest zeros, then only the conditional assignment would have given the desired result (since 2 is not false, and hence true).

Note that the ‘: *FalseExpression*’ part is optional. For example:

```
new = Seasonal ? CONS;
```

is a valid conditional assignment statement. Now the variable will have the value of `CONS` for the first quarter, but the other observations of `new` are not touched (so if `new` is created by this expression, it will contain missing values for quarters 2, 3 and 4, making the variable unusable for modelling purposes). Another example:

```
new = (CONS == MISSING) ? 0 : CONS;
```

This new variable takes the values of `CONS`, replacing missing values by zeros.

10.3.7 Indexing

It is possible to index a particular observation in a variable using rectangular brackets. For example:

```
CONS_1 = CONS[-1];
```

There are three methods of indexing:

- Relative to the current observation.

Each Algebra statement can be seen as running within a loop over all observations. The relative index is offset to the current observation in the loop. So

```
CONS_1 = CONS[-1];
```

can be interpreted as:

```
for (t = first observation; t <= last observation; t = t + 1)
  CONS_1[t] = CONS[t - 1];
```

Consequently, this statement has the same effect as

```
CONS_1[0] = CONS[-1];
```

- By absolute date

In this case the index has the form *year(period)*, for example

```
x = INC + CONS[1980(1)];
y[1955(1)] = 12;
```

This type of indexing was used in §4.9.2.

- By absolute index

In this case the index has the form *index(0)*, which references observation *index* (with the first observation having index 0!). For example

```
x = INC + CONS[0(0)]; // add first CONS value
```

10.3.8 Keywords

The following keywords are reserved by algebra:

Table 10.2 Reserved Algebra keywords

keyword	value
FALSE	0.
TRUE	!FALSE
FREQUENCY	data frequency
MISSING	the missing value
PI	Pi (3.1415..)
NOBS	number of observations

10.4 Algebra implementation

Each Algebra statement has an implicit loop over the sample size around it. E.g.

```
LCONS = log(CONS);
meanLCONS = mean(LCONS);
```

is equivalent to the following pseudo code:

```
for (t = first observation; t <= last observation; t = t + 1)
    LCONS[t] = log(CONS[t]);
for (t = first observation; t <= last observation; t = t + 1)
    meanLCONS[t] = mean(LCONS);
```

So meanLCONS will have the same value for every observation.

By default, the loop is over all the observations of the database. However, the `insample` function can restrict the loop to a subsample.

Implementing transformations in Ox is different: there, variables are extracted from the database as column vectors and operations work on the vector as a whole. Implementing the same code in Ox:

```
#include <oxstd.h>
#import <database>
main()
{
    decl db = new Database();

    db.Load("data.oxdata");

    db.Append(log(db.GetVar("CONS")), "LCONS");
    decl mean = meanc(db.GetVar("LCONS"));
    db.Append(constant(mean, db.GetSize(), 1), "meanLCONS");

    db.Info();
    delete db;
}
```

Note that the mean of a vector is just a single number, so creating a vector which has this value for every observation is implemented with the `constant` function in Ox (and adding this to the database is not very useful).

10.5 Algebra Functions

A large set of functions is provided by algebra. Most of these take both variables and constants as arguments. A function name must be followed by parentheses, even if it does not take any arguments. See the Algebra function list of [Table 10.3](#) for function definitions. Some examples of statements involving functions are:

```
lcons = log (CONS);           // takes the natural logarithm of CONS
cons_1 = lag(CONS, 1);        // lag CONS one period
ccons = exp(log(CONS));       // gives back original CONS
dummy = insample(1979, 1, 1983, 4) ? log(CONS) : 0;
           // Dummy will be log(CONS) for the period 1979(1)
           // to 1983(4), and 0 outside it.
trough = (lag(trough,1) == MISSING || CONS < lag(trough,1))
          ? CONS : lag(trough,1);
           // a trough can also be created with the built in
           // algebra function trough()
dummy = (year() == 1979) ? 1 : 0; // creates a dummy variable
```

10.5.1 Differencing and lag functions

A variable is lagged using the `lag` function,¹ and differenced using the `diff` function. For example, for first lag and difference respectively, as well as second lag and difference:

```
Y_1 = lag(Y, 1);
DY = diff(Y, 1);
Y_2 = lag0(Y, 2);
D2Y = diff(Y, 2);
```

This code adopts the naming convention which is used by OxMetrics: append an underscore followed by lag length for a lagged variable, and a prefix ‘D’ for differenced variables. Note that lags should not be created in the database: applications create them during model formulation, so that they can be handled appropriately.

When taking a first difference or lag using `diff`, `dlog`, or `lag`, the first observation becomes missing; for a second difference or lag, the first two observations are missing, etc. The `lag0` function sets the missing lags to zero.

Neither function can have an expression as the argument for differencing. So you cannot write `diff(diff(Y,1),1)` but must write:

```
DY = diff(Y, 1);
DDY = diff(DY, 1);
```

Note that this DDY is not the same as D2Y above.

Finally, the `dlog(.)` function takes the first difference of the (natural) logarithm of the variable.

10.5.2 Implicit functions

A few functions are implicitly defined, so that the corresponding variables do not need to be present in the database. For example, if an expression uses `CSeasonal` as vari-

¹In most cases it is better to avoid storing lags in the database: most estimation modules handle lags explicitly, allowing for dynamic analysis of the model.

able on the right-hand side, but the variable does not exist in the database, it is implicitly translated to the function call `cseasonal(0)`, while `CSeasonal_1` becomes `cseasonal(1)`. This also implies to `Seasonal`.

The implicit call is not made if the variable does exist in the database.

A similar implicit definition is used for indicators of the type `DI#`, `II#`, `SI#`, `TI#`, `QI#`, `AO#`, `LS#`, `LT#`, `MI#`. A date field is expected after the `#`, while the multiplicative indicator has an additional variable name after the `@` symbol. Examples are:

```
ls#1973(1) = LS#1973(1);
lt#1973(1) = LT#1973(1);
my#1973(1) = MI#1973(1)@LY;
```

The functions are defined as:

<code>DI#y(p)</code>	1 in $y(p)$, -1 in next period, 0 otherwise,
<code>II#y(p)</code>	1 in $y(p)$, 0 otherwise (same as <code>AO#</code>),
<code>SI#y(p)</code>	1 up to $y(p)$, 0 afterwards,
<code>LS#y(p)</code>	0 up to $y(p)$, 1 afterwards,
<code>TI#y(p)</code>	..., -4, -3, -3, -1, 0 up to $y(p)$, 0 afterwards,
<code>LT#y(p)</code>	0 up to $y(p)$, 1, 2, 3, ... afterwards,
<code>QI#y(p)</code>	<code>TI#y(p)</code> squared,
<code>MI#y(p)@v</code>	$v \times SI#y(p)$.

`II#1950(1)` is the same as the function call `II(1950, 1)`.

10.5.3 ACF and periodogram functions

The sample autocorrelation function can easily be plotted, see §4.6.1. To compute the exact values of the ACF, use the following algebra code:

```
acf_lag = MISSING;
acf_dcons = MISSING;
acf(DCONS, acf_dcons, acf_lag);
```

The first two lines are required to create the output variables for the `acf` function. The output from `acf` is the actual ACF values (here stored in `ACF_DCONS`), and the lag length (in `ACF_LAG`).

The `periodogram` function has a similar syntax, returning the sample periodogram in the second argument, and the frequencies at which it was computed in the third. Section 8.7 describes this in more detail.

10.5.4 Sorting functions

The sort functions change the order of observations in the database and must be handled with care. Note that the `trend()` function can be used to create a variable which corresponds to the original observation index of the data. So if that variable is sorted together with the variable we wish to sort, it can be used to ‘unsort’ the variable.

```
sort(arg1)
```

Sorts the variable `arg1` in increasing order (`arg1` must be a variable, and cannot be an expression). Returns the value of `arg1` after sorting. Suppose the residuals of a regression have been saved in the variable called `Residual`, then a sorted copy is created as follows:

```

    res = Residual;
    sort(res);
_sortby(arg1, arg2)

```

Sorts the variable *arg1* in increasing order, and sorts *arg2* accordingly. Returns the value of *arg1* after sorting. Both *arg1* and *arg2* must be variables. Suppose the residuals of a regression have been saved in the variable called *Residual*, then a sorted residual is created as follows:

```

index = trend();
tmp = _sortby(Residual, index);
      /* Residual is sorted, and index accordingly */
      /* tmp is a dummy variable, */
      /* at this stage identical to Residual. */
      /* Now it is easy to locate outliers in Residual */

```

The following statement will reset the old ordering (do not rerun the `index = trend();` statement, because that will overwrite the index and lose the information on the original ordering):

```

tmp = _sortby(index, Residual);
      /* Restores index and Residual, */
      /* at this stage index is equal to Trend. */
_sortallby(arg1)

```

Sorts the variable *arg1* in increasing order, and sorts the whole database accordingly. Returns the value of *arg1* after sorting. This function will be most useful with cross-section data; for example, to push missing values to the end:

```

exclude =
    (Var1 == MISSING || Var2 == MISSING || Var3 == MISSING);
/* exclude is 1 for each observation where any of the
   three variables has a missing value, 0 elsewhere */

tmp = _sortallby(exclude);
      /* the observations without any missing values will
         precede those with missing values, making it easy
         to exclude them from the regression.
         An index variable can be used to restore the order */

```

10.5.5 Smoothing functions

The smoothing functions which are available in scatter plots can also be accessed from algebra. [Chapter 8](#) briefly reviews the mathematics underlying the smoothers.

10.5.5.1 Hodrick–Prescott filter

The Hodrick–Prescott is a filter which is popular in macro-economics, and virtually identical to a natural cubic spline (§8.10.2). The syntax is:

```
var_dest = smooth_hp(var, alpha, var_dest);
```

where *var* is the variable to be smoothed, *alpha* is the bandwidth (use 0 for the default of 1600), and *var_dest* is the destination variable (must be different from *var*). For example:

```
hpCONS = smooth_hp(CONS, 0, hpCONS);
```


which creates `hpCONS` through the assignment statement, and uses it as the destination for the fitted values from the filter.

If there are missing values in the data series, `smooth_hp` uses data starting from the first valid observation, and stopping at the first missing value thereafter (this is unlike the smoothing functions of the next section, which will skip over missing values, using the maximum sample). The sample used can be restricted using the `insample` function.

The bandwidth parameter associated with the Hodrick–Prescott filter is $100(\text{freq})^2$:

annual data	100
quarterly data	1600
monthly data	14400

10.5.5.2 Kernel and spline smoothing

The kernel based smoother (using the Epanechnikov kernel, §8.10.1), and the natural cubic spline (§8.10.2) are computed respectively by:

```
var_dest = smooth_np(var,alpha,var_dest);
var_dest = smooth_sp(var,alpha,var_dest);
```

where *var* is the variable to be smoothed, *var_dest* is the destination variable (must be different from *var*), and *alpha* is the bandwidth, used as follows:

- 0 use automatic bandwidth selection based on generalized cross validation;
- < 0 the absolute value is used for the bandwidth;
- > 0 specifies the equivalent number of parameters to be used.

These functions will use all available observations (unless restricted by the `insample` function), and will fill in missing values using the fit from the smoother. Some examples were given in §7.2.6.

10.5.5.3 Exponentially-weighted moving average and correlation

A simple exponentially-weighted moving average (see §8.13) of a series *y* can be created using:

```
ysmo = ewma(y, 0.5, 0, ysmo);
```

Here $\alpha = 0.5$. An extended version introduces a slope coefficient β , and is also called Holt's method, e.g. with $\beta = 0.1$:

```
ysmo = ewma(y, 0.5, 0.1, ysmo);
```

In both cases, in-sample missing values are replaced by forecasts.

The exponentially weighted moving sample correlation (see §8.14) of two series *y* and *x* (both assumed to have expectation zero) can be created by:

```
xysmo = ewmc0(x, y, 0.5, xysmo);
```

Here, λ is set to 0.5.

10.5.5.4 Date and time functions

When a database is dated, there are functions to return components of the current date and time for each observation: `day`, `month`, `week`, `hours`, `minutes`, `seconds`.

The `makedate` and `maketime` functions translate year, month, day and hours, minutes, seconds to date and time respectively. These can be added together to create a date with a time.

The `dayofweek`, `isdayofmonth` and `iseaster` functions are needed when determining holidays, for example to delete all holidays and weekends from a database. The following code, supplied as `holiday_us.alg` in the algebra folder, determines all the official US holidays:

```
hol_us_fix = /* official US holidays: fixed dates */
  (month()==1 && day()==1) /* New Year's Day */
|| (month()==7 && day()==4) /* Independence Day */
|| (month()==11 && day()==11) /* Veterans Day */
|| (month()==12 && day()==25); /* Christmas Day */
hol_us_fix = hol_us_fix /* fixed dates moved to Fri */
|| (month()==12 && day()==31 && dayofweek()==6)
|| (month()==7 && day()==3 && dayofweek()==6)
|| (month()==11 && day()==10 && dayofweek()==6)
|| (month()==12 && day()==24 && dayofweek()==6);
hol_us_fix = hol_us_fix /* fixed dates moved to Mon */
|| (month()==1 && day()==2 && dayofweek()==2)
|| (month()==7 && day()==5 && dayofweek()==2)
|| (month()==11 && day()==12 && dayofweek()==2)
|| (month()==12 && day()==26 && dayofweek()==2);

hol_us_flt = /* official US holidays: floating dates */
  isdayofmonth(1,2,3) /* 3rd Mon in Jan: Martin Luther King day*/
|| isdayofmonth(2,2,3) /* 3rd Mon in Feb: Washington's Birthday */
|| isdayofmonth(5,2,-1) /* last Mon in May: Memorial Day */
|| isdayofmonth(9,2,1) /* 1st Mon in Sep: Labor Day */
|| isdayofmonth(10,2,2) /* 2nd Mon in Oct: Columbus Day */
|| isdayofmonth(11,5,4) /* 4th Thu in Nov: Thanksgiving */
```

UK holidays also require the `iseaster` function:

```
hol_uk = /* Bank holidays for England and Wales */
  (month()==1 && day()==1) /* Jan 1 */
|| (month()==1 && day()==2 && dayofweek() == 2) /* Jan 1 on Sun */
|| (month()==1 && day()==3 && dayofweek() == 2) /* Jan 1 on Sat */
|| iseaster(-2) /* Good Friday */
|| iseaster(1) /* Easter Monday */
|| isdayofmonth(5,2,1) /* Early May Bank Holiday */
|| isdayofmonth(5,2,-1) /* Spring Bank Holiday */
|| isdayofmonth(8,2,-1) /* Summer Bank Holiday */
|| (month()==12 && day()==25) /* Christmas day */
|| (month()==12 && day()==26) /* Boxing day */
|| (month()==12 && day()==27 && dayofweek() == 3) /* Xmas on Sun */
|| (month()==12 && day()==27 && dayofweek() == 2) /* Xmas on Sat */
|| (month()==12 && day()==28 && dayofweek() == 3) /* Xmas on Sat */
```

10.6 Algebra function summary

Available functions are listed in [Table 10.3](#). Any function operating on missing values returns a missing value. Any function which fails also returns a missing value. Where the argument is *var*, it must be just a variable name, e.g. `one = 1; cum(one);`

is allowed, but `cum(1)` is not. Similarly: `lag(CONS+1,1)` is not allowed, while `lag(CONS,1)+1` is.

The indicator functions `DI()`, ..., `QI()` for fixed frequency are, e.g. `II(2020,3)`, and for a dated database: `II(2020-03-23, 0)`. The date refers to the point at which the underlying process changes.

Several examples of Algebra code were given above; for convenience these are summarized here:

```
LX = log(X);
DX = diff(X, 1);
"log(X)" = log(X);
aap = CONS + 1; /* comment /* nested comment */ */
new1 = !season();
cons2 = 2 * CONS - 1;
cons3 = CONS / 3 - (cons2 + 5) * 1.55;
Seasonal = season();
new2 = Seasonal ? CONS : 0;
new3 = Seasonal * CONS;
new4 = Seasonal ? CONS;
new5 = (CONS == MISSING) ? 0 : CONS;
CONS1ag1 = CONS[-1];
X = INC + CONS[1980(1)];
y[1955(1)] = 12;
X = INC + CONS[0(0)]; // add first CONS value
```

Table 10.3 Algebra functions

Function	Returns
abs (<i>arg</i>)	absolute value of <i>arg</i> (same as fabs)
acf (<i>var,dest,lag</i>)	autocorrelation function (all arguments must be a variable)
acos (<i>arg</i>)	arccosine of <i>arg</i> , or 0 if $ arg > 1$
almon (<i>var,arg,power</i>)	almon lag of <i>var</i> $\text{almon}(x, c_1, c_2) = \sum_{k=0}^{c_1} (c_1 + 1 - i)^{c_2} x_{t-i} / \sum_{k=0}^{c_1} (c_1 + 1 - i)^{c_2}$
asin (<i>arg</i>)	arcsine of <i>arg</i> , or 0 if $ arg > 1$
atan (<i>arg</i>)	arctangent of <i>arg</i>
ceil (<i>arg</i>)	ceiling of <i>arg</i> (the smallest integer $\geq arg$)
cos (<i>arg</i>)	cosine of <i>arg</i>
cseasonal (<i>lag</i>)	centred seasonal with <i>lag</i>
cum (<i>var</i>)	cumulative sum of <i>var</i>
date ()	date value (dated database) or the observation index
day ()	day (dated database)
dayofweek ()	day of the week (dated database, 1=Sunday, ..., 7=Saturday)
delete (<i>var,...</i>)	delete the list of variables when this Algebra run completes
denschi (<i>arg</i>)	$\chi^2(arg)$ density
DI (<i>year,period</i>)	difference indicator for <i>year(period)</i>
DI (<i>date,0</i>)	difference indicator for calendar <i>date</i>
densf (<i>arg1,arg2</i>)	$F(arg1, arg2)$ density
densn ()	$N(0, 1)$ density
denst (<i>arg1</i>)	student-t(<i>arg</i>) density
diff (<i>var,arg</i>)	arg^{th} difference of <i>var</i> , or the missing value if <i>var</i> is not a variable, or the difference cannot be taken
div (<i>arg1,arg2</i>)	returns the result of the (32 bit) integer division
dlog (<i>var</i>)	first difference of natural logarithm of <i>var</i>
dummy (<i>yr1,per1,yr2,per2</i>)	1 inside the sample <i>yr1 (per1) – yr2 (per2)</i> , 0 outside it
dummydates (<i>date1,date2</i>)	as dummy for a dated database
ewma (<i>var,alpha,beta,dest</i>)	see §10.5.5.3, §8.13
ewmc0 (<i>varx,vary,lambda,dest</i>)	see §10.5.5.3, §8.14
exp (<i>arg</i>)	exponential function of <i>arg</i>
fabs (<i>arg</i>)	absolute value of <i>arg</i>
floor (<i>arg</i>)	floor of <i>arg</i> (the largest integer $\leq arg$)
fmod (<i>arg1,arg2</i>)	floating point remainder of <i>arg1/arg2</i>
hours ()	
II (<i>year,period</i>)	impulse indicator for <i>year(period)</i>
II (<i>date,0</i>)	difference impulse for calendar <i>date</i>
indates (<i>date1,date2</i>)	as insample for a dated database
insample (<i>yr1,per1,yr2,per2</i>)	true if inside the sample <i>yr1(per1) to yr2(per2)</i> , false outside
isdayofmonth (<i>month,dow,nth</i>)	returns 1 for n^{th} day of week in month; <i>dow</i> : 1=Sunday, etc. <i>nth</i> < 0: from end
iseaster (<i>daysafter</i>)	returns 1 for <i>daysafter</i> easter

(Continued on next page)

Function	Returns	(Continued)
<code>lag(var,arg)</code>	arg^{th} lag of <i>var</i> , or the missing value if <i>arg1</i> is not a variable, or the lag cannot be taken; use $arg < 0$ for leads	
<code>lag0(var,arg)</code>	arg^{th} lag of <i>var</i> , or zero if <i>arg1</i> is not a variable, or the lag cannot be taken; use $arg < 0$ for leads	
<code>lead(var,arg)</code>	arg^{th} lead of <i>var</i> , or the missing value if <i>arg1</i> is not a variable, or the lead cannot be taken	
<code>log(var)</code>	natural logarithm of <i>var</i>	
<code>log10(var)</code>	base 10 logarithm of <i>var</i>	
<code>loggamma(var)</code>	logarithm of the gamma function at <i>var</i>	
<code>LS(year,period)</code>	level shift indicator for <i>year(period)</i>	
<code>LS(date,0)</code>	difference level shift indicator for calendar <i>date</i>	
<code>LT(year,period)</code>	local trend indicator for <i>year(period)</i>	
<code>LT(date,0)</code>	difference local trend indicator for calendar <i>date</i>	
<code>makedate(year,month,day)</code>	make date values	
<code>maketime(hour,min,sec)</code>	make time values	
<code>max(arg1, arg2)</code>	maximum of <i>arg1</i> and <i>arg2</i>	
<code>mean(var)</code>	mean of <i>var</i>	
<code>MI(arg,year,period)</code>	multiplicative indicator for variable <i>arg</i> at <i>year(period)</i>	
<code>MI(arg,date,0)</code>	multiplicative indicator for variable <i>arg</i> at calendar <i>date</i>	
<code>min(arg1,arg2)</code>	minimum of <i>arg1</i> and <i>arg2</i>	
<code>minutes()</code>	minutes (dated database)	
<code>month()</code>	month (dated database)	
<code>movingavg(var,lag,lead)</code>	the moving average of <i>var</i>	
	$MAV(x, c_1, c_2) = \sum_{k=t-c_1}^{t+c_2} x_k / (c_2 + c_1 + 1)$	
<code>movingSD(var,lag,lead,mvar)</code>	the moving standard deviation of <i>var</i> around <i>mvar</i>	
	$MSD(x, c_1, c_2, z) = \left[\sum_{k=t-c_1}^{t+c_2} \frac{(x_k - z_k)^2}{(c_2 + c_1 + 1)} \right]^{1/2}$	
<code>pacf(var,dest,lag)</code>	PACF (all arguments must be a variable)	
<code>peak(var)</code>	maximum value of <i>var</i> up to current observation	
<code>period()</code>	current period	
<code>periodogram(var,dest,freq)</code>	periodogram (all arguments must be a variable)	
<code>probn(arg1)</code>	$P(X \leq arg1 X \sim N(0, 1))$	
<code>quanchi(p,arg)</code>	$\chi^2(arg)$ quantiles at <i>p</i>	
<code>quanf(p,arg1,arg2)</code>	$F(arg1, arg2)$ quantiles at <i>arg1</i>	
<code>quann(p)</code>	$N(0, 1)$ quantiles at <i>p</i>	
<code>quant(p,arg1)</code>	student-t(<i>arg</i>) quantiles at <i>p</i>	
<code>QI(year,period)</code>	quadratic trend indicator for <i>year(period)</i>	
<code>QI(date,0)</code>	difference quadratic trend for calendar <i>date</i>	
<code>ranu()</code>	uniform random numbers	
<code>ranchi(arg)</code>	$\chi^2(arg)$ random numbers	
<code>ranf(arg1,arg2)</code>	$F(arg1, arg2)$ random numbers	
<code>rann()</code>	$N(0, 1)$ random numbers	
<code>rant(arg1)</code>	student-t(<i>arg</i>) random numbers	
<code>ranseed(arg)</code>	sets the random number seed to <i>arg</i> , returns <i>arg</i>	
<code>round(arg)</code>	rounded value of <i>arg</i>	
<code>season()</code>	1 in period 1, 0 otherwise	
<code>seasonal(lag)</code>	seasonal with <i>lag</i>	
<code>seconds()</code>	seconds (dated database)	

(Continued on next page)

Function	Returns	(Continued)
SI(<i>year,period</i>)	step indicator for <i>year(period)</i>	
SI(<i>date</i> ,0)	difference step for calendar <i>date</i>	
sin(<i>arg</i>)	sine of <i>arg</i>	
smooth_hp(<i>var,alpha,dest</i>)	see §10.5.5	
smooth_np(<i>var,alpha,dest</i>)	see §10.5.5	
smooth_sp(<i>var,alpha,dest</i>)	see §10.5.5	
sort(<i>var</i>)	see §10.5.4	
_sortallby(<i>var1</i>)	see §10.5.4	
_sortby(<i>var1, var2</i>)	see §10.5.4	
sqrt(<i>arg</i>)	square root of <i>arg</i>	
stock(<i>var,arvalue,init</i>)	integrates <i>var</i>	
	$\text{stock}(x, c_1, c_2) = (1 - c_1)y_{t-1} + x_t$, given $y_0 = c_2$	
stockv(<i>var,arvar,init</i>)	integrates <i>var</i>	
	$\text{stockv}(x, z, c_2) = (1 - z_t)y_{t-1} + x_t$, given $y_0 = c_2$	
tailchi(<i>arg1,arg2</i>)	$P(X \geq \text{arg1} X \sim \chi^2(\text{arg2}))$	
tailf(<i>arg1,arg2,arg3</i>)	$P(X \geq \text{arg1} X \sim F(\text{arg2}, \text{arg3}))$	
tailn(<i>arg1</i>)	$P(X \geq \text{arg1} X \sim N(0, 1))$	
tailt(<i>arg1,arg2</i>)	$P(X \geq \text{arg1} X \sim t(\text{arg2}))$	
tan(<i>arg</i>)	tangent of <i>arg</i>	
TI(<i>year,period</i>)	trend indicator for <i>year(period)</i>	
TI(<i>date</i> ,0)	difference trend for calendar <i>date</i>	
time()	time value (dated database)	
trend()	1 for the first observation, 2 for the second, etc.	
trough(<i>var</i>)	minimum value of <i>var</i> up to current observation	
variance(<i>var</i>)	variance of <i>var</i>	
year()	current year.	

Chapter 11

Batch Language

11.1 Introduction

The Batch language gives some control over OxMetrics through a command language. This allows for automating repetitive tasks, or as a quick way to get to a certain point in your analysis. Other programs (such as PcGive, STAMP and X12arima) extend this batch language with their own module-specific commands.

OxMetrics allows you to save the most recent model from the current module as a batch file (if supported by the module). If a model has been created interactively, it can be saved as a batch file for further editing or easy recall in a later session. This is also the most convenient way to create a batch file.

11.2 Executing Batch commands

If an error occurs while executing the Batch commands, the processing will be aborted, and control returned to OxMetrics. All successful statements up to that point will have been executed.

There are five ways of running Batch commands, as discussed in the next four sections.

11.2.1 Batch Editor (Alt+b)

The Tools/Batch Editor command activates the edit window in which you can edit/load/save a set of batch commands. The file extension used for batch files is .FL.

11.2.2 Batch from Results windows (Ctrl+b)

A text selection containing Batch commands can be run directly from that window using the Edit/Run as Batch command (or Ctrl+b as short cut).

11.2.3 Batch from the File/Open command

If you use File/Open, and set the file type to Run Batch file (*.fl), then the batch file is run immediately.

11.2.4 Batch from the Windows Explorer

You can double click on a .FL file in the Windows Explorer to run the file directly. If OxMetrics is not active yet, it will be started automatically.

11.2.5 Batch from a Batch file

A batch file can be called from another batch file, see §11.5.19.

11.3 Batch files and default folders

When a batch file is run, the working folder is set to that of the batch file. This allows data and algebra files which reside in the folder of the batch file to be found, even if the batch file is run from somewhere else. In addition the `chdir` command allows specifying a new default directory. Finally, paths to data files can be hard coded, but that is to be avoided if possible.

When a batch file is loaded in the batch editor, and then run, there will not be an associated default folder. However, the process of opening the batch file would set the default folder to that of the batch file.

11.4 General Batch command summary

Table 11.1 gives an alphabetical list of the OxMetrics batch language statements.

11.4.1 Comment

Anything between `/*` and `*/` is considered comment. Note that this comment cannot be nested. Everything following `//` up to the end of the line is also comment.

11.4.2 Command types

There are two types of batch commands:

- function calls (with or without arguments) terminated by a semicolon. An example is the `loaddata` command.
- block commands, which are followed by statements between curly brackets. An example is the `algebra` command.

Table 11.1 Batch command summary

```

algebra { ... }
appenddata("filename", "group"="");
appresresults("filename");
break;
chdir("path");
closedata("databasename");
closedrawwindow();
command("command_line");
database("name", year1, period1, year2, period2, frequency);
draw(area, "y", "mode"="");
drawf(area, "y", "function", d1=0, d2=0);
drawptext(area, "text", "x", "y", size=300, rot=0);
drawtext(area, "text", "x", "y", size=300);
drawtitle(area, "text", "pos", size=300);
drawx(area, "y", "x", "mode"="");
drawz(area, "y", "x", "mode"="");
exit;
loadalgebra("filename");
loadbatch("filename");
loadcommand("filename");
loaddata("filename");
loadfile("filename");
loadgraph("filename");
module("name");
package("packagename", "modeltype"="");
print("text");
printdate;
println("text");
savedata("filename");
savedrawwindow("filename", "window"="");
saveresults("filename");
setdraw("option", i1=0, i2=0, i3=0, i4=0, i5=0);
setdrawwindow("name");
show;
usedata("databasename", i1=0);

```

11.4.3 Default arguments

Table 11.1 shows two instances of default arguments, in the `appenddata` command, and in the `setdraw` command. For example, when omitting the last argument as in

```
appenddata("data.oxdata");
```

the command is actually interpreted as:

```
appenddata("data.oxdata", "");
```

11.5 Batch commands

In the following list, function arguments are indicated by *words*, whereas the areas where statement blocks are expected are indicated by Examples follow the list of descriptions. For terms in double quotes, the desired term must be substituted and provided together with the quotes. A command summary is given in [Table 11.1](#). Module specific commands are documented with each module (so, for *PcGive* batch commands consult Volume I of the *PcGive* books).

11.5.1 algebra { ... }

Contains the algebra code to execute. The block of algebra should be enclosed in curly braces, for example:

```
algebra
{
    // Create SAVINGSGL in database test
    SAVINGSGL = lag(INC,1) - lag(CONS, 1);
    y = log(Y);
}
```

11.5.2 appenddata("filename", "group"="");

Appends the data from the named data file to the existing in-memory database. If no database exists yet, the named data file will be opened as in the `loaddata()` command. If the second argument is omitted, or equal to "" (that is, no group name is specified) the whole file will be appended, otherwise only the named group.

11.5.3 appresults("filename");

Append the contents of the Results window to the named file and *clear the Results window*.

11.5.4 break;

Stop the batch file, return to OxMetrics menus. While a batch file is running, there is no way of stopping it other than this batch command.

11.5.5 chdir("path");

Sets the current working folder to the specified path. There are three special forms of this command:

1. `chdir("#batch")` sets the working folder to that of the batch file;
2. `chdir("#home")` sets the working folder to that of the OxMetrics.exe executable file (but if it ends in \bin that will be deleted). If OxMetrics.exe is in its default folder \Program Files\OxMetrics\bin, then this command will set the working folder to \Program Files\OxMetrics.
3. `chdir("")` prints the current working folder.

11.5.6 `closedata("databasename");`

Closes the database, which is currently loaded in OxMetrics. The database is closed without saving, so any changes which have been made since the last save are lost.

11.5.7 `closedrawwindow();`

Closes the batch draw window.

11.5.8 `command("command_line");`

Sends the line of text directly to the active module. The module must be able to accept commands (either via batch, or as an input console module).

11.5.9 `database("name", year1, period1, year2, period2, freq);`

Creates a database with the specified name and frequency, and sample period *year1* (*period1*) to *year2* (*period2*).

11.5.10 `draw(area, "y", "mode=");`

Graphs variable *y* from the current database in the requested area (the first area has number 0). If the second argument is missing, the graph is a standard time-series plot. For other types of graphs, mode can be one of:

"bar"	draw bars
"diff"	take first differences of <i>y</i>
"dlog"	growth rates (first difference of natural logarithm)
"index"	draw index line
"log"	take natural logarithms of <i>y</i>
"logscale"	take natural logarithms and show on (natural) log scale
"match"	match in variables area by means and ranges to first variable
"shading"	draw shading
"startat100"	scale the variable to start at 100.
"symbol"	draw symbols as well as the line
"twoscale"	as "match", but showing second variable in area on right-hand axis
"seasonal"	draw seasonal sub-plot

11.5.11 `drawf(area, "y", "function", d1=0, d2=0);`

Graphs a function of the variable *y* from the current database in the requested area (the first area has number 0). The function argument can be one of:

"acf"	autocorrelation function with specified lag length
"boxplot"	boxplot
"cdf"	estimated distribution (QQ plot against normal)
"chisq"	QQ plot against $\text{Chi}^2(d1)$
"cumfreq"	cumulative frequency count (for non-standard bars, specify the bar count)
"density"	estimated density (for non-standard bars, specify the bar count)
"f"	QQ plot against $F(d1,d2)$
"frequency"	frequency count (for non-standard bars, specify the bar count)
"histogram"	histogram (for non-standard bars, specify the bar count)
"normal"	QQ plot against normal with same mean and variance
"pacf"	partial autocorrelation function with specified lag length
"periodogram"	periodogram
"spectrum"	spectral density using specified lag length
"t"	QQ plot against student-t($d1$)
"uniform"	Quantile plot (QQ against uniform)

11.5.12 `drawptext(area, "text", "x", "y", size=300, rot=0);`

11.5.13 `drawtext(area, "text", "x", "y", size=300);`

11.5.14 `drawtitle(area, "text", "pos", size=300);`

Adds text to the specified area.

"drawptext"	draws at pixel coordinates
"drawtext"	draws at world coordinates
"drawtitle"	draws a title at position "" (top), "X", "Y", or "Z"

11.5.15 `drawx(area, "y", "x", "mode"="");`

Graphs variable y against x from the current database in the requested area (the first area has number 0). This is a standard scatter plot, but if the second argument equals "", the graph is a standard time-series plot. For other types of graphs, mode can be one of:

"alt"	use alternate style: x,y labels along axes
"project"	add regression line with projections
"regression"	add a regression line
"smooth"	add a cubic spline (automatic bandwidth)

11.5.16 `drawz(area, "y", "x", "mode"="");`

Graphs variable y against x by z from the current database in the requested area (the first area has number 0). This is a standard scatter plot, but the second argument may equal "" for a time-series plot. The mode argument can be one of:

"bar"	z is error bar
"band"	z is error band
"fan"	z is error fan
"hilo"	high-low
"symbol"	z is symbol size
"value"	print values of z

For 3-dimensional graphs, the mode argument can be one of:

"contour"	contour from scattered data; x, y, z are vectors
"contourx"	contour from tabular data; x, y are vectors, $z=z(x, y)$
"contoury"	contour from tabular data; x, y are vectors, $z=z(y, x)$
"points"	sequence of 3D points
"surface"	surface from scattered data; x, y, z are vectors
"surfacex"	surface from tabular data; x, y are vectors, $z=z(x, y)$
"surfacey"	surface from tabular data; x, y are vectors, $z=z(y, x)$
"trisurface"	triangulated surface from scattered data; x, y, z are vectors

11.5.17 `exit;`

Stops the batch file and *exits OxMetrics*.

11.5.18 `loadalgebra("filename");`

Loads and executes the algebra code from the named .ALG file. The extension must be provided. See §11.3 on the default folder.

11.5.19 `loadbatch("filename");`

Loads and executes the batch code from the named .FL file. The extension must be provided. See §11.3 on the default folder.

11.5.20 `loadcommand("filename");`

Sends the contents of the named file to the active module. The module must be able to accept commands (either via batch, or as an input console module).

11.5.21 `loaddata("filename");`

Load the data from the named .IN7 file. To load, a full pathname would sometimes have to be specified, e.g. "c:\mydata\data.oxdata". The extension indicates the data type and must be provided. Inside OxMetrics, the database will then be known as data.oxdata, and will be the current default database. See §11.3 on the default folder.

If the database is already opened, the command works as `usedata()`.

11.5.22 `loadgraph("filename");`

Load the graph from the named file. To load, a full pathname would sometimes have to be specified, e.g. "c:\mydata\data.oxdata". The extension indicates the file type and must be provided (but OxMetrics will always require the .gwg file for loading, so when using `loadgraph("fig1.eps")`, OxMetrics will look for `fig1.gwg`).

11.5.23 `module("name");`

Starts the specified module. When the module is already active, it becomes the focus for subsequent module-specific batch commands. Otherwise the module is started first.

11.5.24 `package("packagename", "modeltype"="");`

Used to specify the package and model type for modelling:

```
package("PcGive", "Single-equation");
```

11.5.25 `print("text");`**11.5.26** `println("text");`

Prints the specified text in the Results window. This can be useful to introduce explanatory notes. The `println` commands adds a new line after printing the text.

11.5.27 `printdate;`

Prints the current date and time in the Results window. The date and time are pasted into the text in ISO format `yyyy-mm-dd hh:mm:ss`, so putting the year first and using a 24-hour clock.

11.5.28 `savedata("filename");`

Save the current database to the named .IN7 file. The .BN7 file will get the same base name. If files with these names already exist, they will be overwritten!

11.5.29 `savedrawwindow("filename", "window"="");`

If no window is specified, the command saves the current Batch Graphics window to the named file (or gives an error message 'Cannot save to' if there is no Batch Graphics window or the file cannot be saved). Otherwise it saves the named graphics window.

11.5.30 `saveresults("filename");`

Save the contents of the results to the named file and *clear the Results window*. If a file with that name already exists, it will be overwritten!

11.5.31 `setdraw("option", i1=0, i2=0, i3=0, i4=0, i5=0);`

This command changes the default settings used in graphics. The easiest way to use this command is to design the layout using the interactive dialogs (Graphics Setup and Postscript Setup). Then use the Write as batch command on both pages of the Graphics Setup dialog to record the batch code in the results window. This can then be pasted to the batch editor, and saved to a file. Or edited, and run directly from the results window.

<i>option</i>	changes	<i>option</i>	changes
"axis"	axis fonts/ticks	"legendfontsize"	legend font size
"axisline"	axis options	"legendhide"	legend hiding
"axisformat"	format of labels	"legendresize"	legend resizing
"box"	box and grid	"line"	colour line settings
"bw"	black&white settings	"linebw"	b&w line settings
"color"	colour settings	"margin"	paper margins
"colormodel"	PostScript model	"palette_max"	palette light colour
"default"	reset default	"palette_min"	palette dark colour
"font"	font	"papercolor"	Paper colour
"grid"	grid style	"printpage"	PostScript paper
"histogram"	bar colours	"symbol"	symbol settings
"legend"	legend style	"xystyle"	cross-plot style

Use of the default option is as follows:

<code>setdraw("default")</code>	reset all graphics defaults,
<code>setdraw("default", 0)</code>	reset all graphics defaults,
<code>setdraw("default", 1)</code>	reset layout, leave line types and colours,
<code>setdraw("default", 2)</code>	reset line types and colours, leave layout.

The following table lists the integer arguments for each option, with the range of possible values. If no range is given, the argument is a size in pixel coordinates (see Ch. 12). The default width and precision for `axisformat` is 8 and 6 respectively.

<i>linetype</i>	<i>papertype</i>	<i>colormodel</i>
solid	0 A4	0 black & white
dotted	1 Letter	1 black, white, gray
dashed	2 user defined	2 gray
long dashes		3 color
user defined		
<i>symboltype</i>		
0 solid box	5 none	10 diamond
1 open box	6 line	11 solid diamond
2 plus	7 solid circle	12 cross
3 dash	8 triangle	
4 circle	9 solid triangle	

<i>option</i>	<i>i1</i>	<i>i2</i>	<i>i3</i>	<i>i4</i>	<i>i5</i>
"axis"	<i>fontsize</i>	<i>step</i>	<i>tick</i>		
"axisline"	<i>no X-line:0-1</i>	<i>no Y-line:0-1</i>	<i>center dates:0-1</i>	<i>no small Y:0-1</i>	
"axisformat"	<i>width</i>	<i>precision</i>	<i>same precision:0-1</i>	<i>leading zeros:0-1</i>	
"box"	<i>box:0-1</i>	<i>X-grid:0-1</i>	<i>Y-grid:0-1</i>		
"bw"	<i>lineno:0-15</i>	<i>red:0-255</i>	<i>green:0-255</i>	<i>blue:0-255</i>	
"color"	<i>lineno:0-15</i>	<i>red:0-255</i>	<i>green:0-255</i>	<i>blue:0-255</i>	
"colormodel"	<i>model:0-3</i>				
"default"	<i>0,1,2</i>				
"font"	<i>fontno:0-3</i>	<i>fontsize</i>			
"grid"	<i>color:0-15</i>	<i>type:0-15</i>			
"histogram"	<i>inside:0-15</i>	<i>outside:0-15</i>			
"legend"	<i>boxed:0-1</i>	<i>columns</i>			
"legendfontsize"	<i>fontsize</i>				
"legendhide"	<i>hide:0-1</i>				
"legendresize"	<i>resize:0-1</i>				
"line"	<i>lineno:0-15</i>	<i>linetype:0-4</i>	<i>width</i>	<i>on</i>	<i>off</i>
"linebw"	<i>lineno:0-15</i>	<i>linetype:0-4</i>	<i>width</i>	<i>on</i>	<i>off</i>
"margin"	<i>left</i>	<i>top</i>			
"palette_max"	<i>index:0-7</i>	<i>red:0-255</i>	<i>green:0-255</i>	<i>blue:0-255</i>	
"palette_min"	<i>index:0-7</i>	<i>red:0-255</i>	<i>green:0-255</i>	<i>blue:0-255</i>	
"papercolor"	<i>red:0-255</i>	<i>green:0-255</i>	<i>blue:0-255</i>		
"printpage"	<i>orientation:0-1</i>	<i>papertype:0-2</i>	<i>X-size</i>	<i>Y-size</i>	
"symbol"	<i>lineno:0-15</i>	<i>symtype:0-4</i>	<i>size</i>		
"xystyle"	<i>2d along axes:0-1</i>	<i>3d along axes:0-1</i>			

11.5.32 `setdrawwindow("name");`

Sets the name of the graphics window in which the graphs appear. The default is "Batch Graphics".

11.5.33 `show;`

Shows the graph created in batch. Note that graphs from `draw/drawf/drawx/drawz` commands are not displayed until the `show` command is issued.

11.5.34 `usedata("databasename", i1=0);`

Sets the default database to *databasename*. The database must already be loaded into OxMetrics. Normally, only the one-argument function would be used, e.g., `usedata("ukm1.in7")`. If the second argument is 1, the database is marked clean: when it is closed there is no prompt for saving it. This can be useful after a section of algebra, which implements basic transformation for a dataset. This can be used as follows:

```
loaddata("ukm1.in7");
loadalgebra("ukm1.alg");
usedata("ukm1.in7", 1);
```


11.6 Examples

We finish with two example batch files. The first uses most non-graphics Batch commands, as well as some commands from the PcGive module.

```

database("test", 1950, 1, 2000, 4, 4); // Create the database
// Append the tutorial data set to test
// note that test has a longer sample than data.oxdata
chdir("C:\Program Files\OxMetrics9"); // default install folder
// it is better to use chdir("#home");
appenddata("data.oxdata");
loaddata("data.oxdata"); // Load the tutorial data. Now there are
// two databases, with data.oxdata the default database
usedata("test"); // make test the default database

algebra
{
    // Create SAVINGSGL in database test
    SAVINGSGL = lag(INC,1) - lag(CONS, 1);
}
package("PcGive", "Single-equation");
system
{
    Y = CONS; // The endogenous variable
    Z = Constant, CONS_1, CONS_2; // the regressors
}
estimate("OLS", 1953, 3, 1992, 3, 8);
// Estimate the system by OLS over 1953(2)-1992(3)
// withhold 8 forecasts

println("Saving results to test.out");
saveresults("test.out"); // Save the contents of the Results
// window to test.out and clear the window
testsummary; // Do the test summary
println("Appending results to test.out");
appresults("test.out"); // Append the contents of the Results
// window to test.out, and clear the window

break; // stop the batch run, remaining
// commands will not be executed
savedata("newtest.in7");// save test to newtest.in7/newtest.bn7
exit; // Exit OxMetrics and PcGive

```

The final example uses many graphics batch commands, and is supplied as `plots.fl` with OxMetrics:

```

chdir("#home");
loaddata("data.oxdata");
setdrawwindow("Time plots");
draw(0, "CONS");
draw(0, "INC", "match");
draw(1, "CONS");
draw(1, "INC", "twoscale");
draw(2, "CONS", "startat100");
draw(2, "INC", "startat100");
draw(3, "CONS", "log");
draw(4, "CONS", "logscale");
draw(5, "CONS", "dlog");
show;

setdrawwindow("Scatter plots");

```

```

drawx(0, "CONS","INC","alt");
drawx(1, "CONS","INC","regression");
drawx(2, "CONS","INC","project");
drawx(3, "CONS","INC","smooth");
show;

setdrawwindow("Function plots");
drawf(0, "CONS","acf", 5);
drawf(0, "CONS","pacf", 5);
drawf(1, "CONS","periodogram");
drawf(2, "CONS","spectrum", 5);
drawf(3, "CONS","density");
drawf(3, "CONS","histogram");
drawf(4, "CONS","cdf");
drawf(5, "CONS","frequency");
drawf(6, "CONS","cumfreq", 20);
drawf(7, "CONS","boxplot");
drawf(8, "CONS","normal");
show;

algebra
{
    z = fabs(INC - 890) * 10;
}
usedata("data.oxdata", 1); // mark as clean

setdrawwindow("Z plots");
drawz(0, "CONS","", "z", "bar");
drawz(1, "CONS","", "z", "band");
drawz(2, "CONS","", "INC", "hilo");
drawz(3, "CONS","INC", "z", "value");
drawz(4, "CONS","INC", "z", "symbol");
show;

database("3d_scatter", 1, 1, 625, 1, 1);
algebra
{
    x = (ranu() - 0.5) * 5;
    y = (ranu() - 0.5) * 5;
    z = exp(-x^2-y^2);
}
setdrawwindow("3D contour plots");
drawz(0, "x","y","z", "contour");
show;

setdrawwindow("3D surface plots");
drawz(0, "x","y","z", "surface");
drawz(1, "x","y","z", "trisurface");
show;

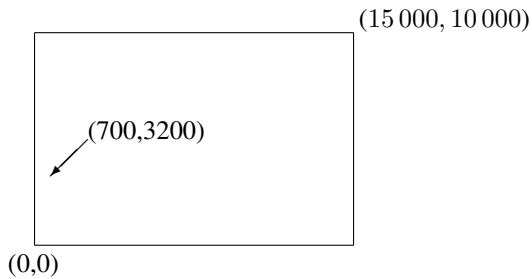
```

Chapter 12

OxMetrics Graphics

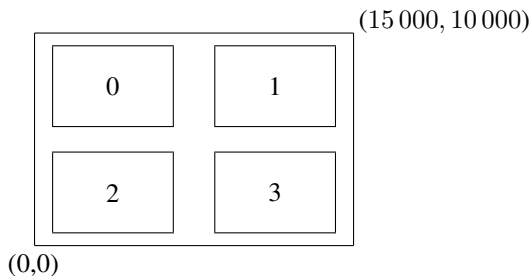
12.1 Graphics paper

Graphs in OxMetrics are drawn on a graphics worksheet, consisting of 10 000 by 15 000 pixels, with (0,0) in the bottom left corner:



These pixels are virtual and different from screen pixels: the paper is always $10\,000 \times 15\,000$, regardless of the screen resolution or the size on screen.

Positions can be specified in pixel coordinates, as for example $(p_x, p_y) = (700, 3200)$. More often it is convenient to use real-world coordinates to map the pixel coordinates into real data values. This is done by specifying an area on the graphics worksheet, and attaching real-world coordinates to it. These areas are allowed to overlap, but need not:



The areas are numbered from left to right and top to bottom, *counting starts at zero*.

Suppose we have set up all areas as being from $(x, y) = (0.0, 0.0)$ to $(x, y) = (1.0, 1.0)$ (again within each area the origin is the lower left corner). Then we can draw a line through area 2 in two ways:

1. in real coordinates within an area
 - step 1: select area 2;
 - step 2: move to (0.0, 0.0);
 - step 3: draw a line to (1.0,1.0).
2. using pixel coordinates on the worksheet
 - step 1: move to pixel coordinates (600,600);
 - step 2: draw a line to pixel coordinates (3600, 3600).

where we assume that (600,600) to (3600,3600) are the pixel coordinates chosen for area 2. Drawing in real-world coordinates has the advantage that it corresponds more closely to our data.

12.2 Creating graphs

A graph is created from data in a database using the Graphics toolbar button, or Model/Graphics. These graphs appear in the window labelled OxMetrics Graphics. Subsequent graphs are added to this window. To start from scratch, close the window, or select and delete each area. To keep the existing OxMetrics graphics window and open a new one for subsequent graphs, use File/New Data Plot Window (note that the menu contents change according to the type of window which is active).

Modules such as PcGive will put their graphs in a window with a different name. Examples of the various types of OxMetrics graphs are given in Chapter 4.

12.2.1 Actual series with optional transformations

plots the actual values of all selected variables against time (or the observation index for undated series), together or each in a separate graph. If there are missing values, these show up as gaps in the line.

- Create separate Plots

This creates as many graphs as there are series.
- 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: $\Delta y_t = y_t - y_{t-1}$
- Seasonal growth rates: $\log(y_t - y_{t-s})$, $s = 4$ for quarterly data, $s = 12$ for monthly data,
- Seasonal differences: $\Delta_s y_t = y_t - y_{t-s}$
- Use log scale: assumes the data are in logs and powers up the axis to reflect the log scale.

12.2.2 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

12.2.3 Scatter plots

The available scatter-plot types are:

- Y against X
- Y against X, labels along the axes
- Scatter plot with regression line
- With cubic spline smooth, automatic bandwidth
- All scatter plots

The Style, Regression and Smoothing sections allow additional options and combinations.

If there are missing values, these are omitted from the graphs.

12.2.4 Distribution

- Estimated density and histogram, optionally with normal reference

The histogram is a simple graph: the range of x_t is divided into intervals, and the number of observations in each interval is counted. The height of each bar records the number of entries in that interval. In OxMetrics, these are divided by the total number of observations to show the relative frequency (use bar to keep the count). OxMetrics sets a default number of intervals dependent on the sample size, but this can be changed by the user by clicking on Smoothing and Use custom bar count to set a different value.
- 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

12.2.5 Time-series: ACF etc.

- Autocorrelation function

The sample autocorrelation function (ACF), or correlogram, plots the correlations \hat{r}_j between x_t and successive x_{t-j} . Also see §8.3. The length of the ACF can be set by the user. Since the correlation between x_t and x_t is always unity, it is not drawn in the graphs.

- Partial autocorrelation function

The partial autocorrelation coefficients correct the autocorrelation for the effects of previous lags. So the first partial autocorrelation coefficient equals the first normal autocorrelation coefficient.

- Cross-correlation function

The sample cross-correlation function (CCF) graphs the correlation between a series and the lags of another series.

- Periodogram

The periodogram and the spectrum (or more accurately here: spectral density) graph the series in the frequency domain. The sample periodogram is based on the coefficients of the Fourier decomposition of the sample autocorrelations (that is, the correlations $\{\hat{r}_j\}$ in §4.6.1 between x_t and x_{t-j}).

- Spectral density

The sample spectral density is a smoothed (and scaled) function of the periodogram. It is symmetric between $-\pi$ and π , and so is only graphed for $[0, \pi]$; 1 on the horizontal axis stands for π , 0.5 for 0.5π , etc. Peaks at certain frequencies can indicate regular (cyclical or seasonal) behaviour in the series.

- Seasonal sub-plot

In a seasonal sub-plot, the data are displayed by season. For example, for quarterly data, first the quarter 1 data are graphed, then quarter 2, etc. This helps detecting changing seasonal patterns.

12.2.6 QQ plots

QQ (quantile-quantile), or cross-probability, plots:

- Quantile plot: against uniform
- QQ plot against normal (same mean, variance)
- QQ plot with choice of distribution

As discussed in the companion book on PcGive, statisticians view variables as being described by probability distributions. If X is the variable, and x a value it could take, then

$$P_x(X > x)$$

is the probability that the value is in fact greater than x . For example, if the variable is the height of a child, when x is one metre, $P_x(X > 1)$ is the probability that the child is taller than a metre. The values of x can be any number, but $P_x(\cdot)$ cannot be negative or exceed unity. Plotting $P_x(X > x)$ against x generates an \int -shaped curve (but more stretched out horizontally), which is hard to interpret.

When X has a uniform distribution over $(0, 1)$, if $P_x(\cdot)$ is plotted against x in a unit square, the result is a straight line. A similar idea applies to all distributions and QQ plots can be selected so some reference distribution is a straight line with the empirical distribution compared to it.

The QQ plot options page on the Graphics dialog allows for a choice of reference distributions. For the t , F and χ^2 distribution it is necessary to also supply the degrees of freedom arguments. Drawing a variable against a uniform results in a so-called quantile plot.

12.2.7 Two series by a third

- Error bars
- Error bands
- Error fans
- High-low
- Show Z values
- As symbol size: Bubble chart

A bubble chart consists of a scatter plot, where the symbols are circles, and the size of the circle indicates a third dimension (e.g. market share).

12.2.8 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

With 3-dimensional plots there are two ways of presenting the data: as a scatter of points in three-dimensional space, or as a table with the X and Y values along the side, and the Z values in the cells. The tabular format specifies the 3D surface directly, whereas for the scatter it is left to OxMetrics to work out the surface. Although the tabular format leads to better results, the scatter is easier to use, as it requires only three variables.

To create a surface from a scatter, OxMetrics derives a triangulation, and from the triangulation a smooth surface. This method works best when the surface is smooth.

There are a few additional edit actions available for 3-dimensional graphs:

- Rotation

The Edit menu has an entry for rotation: left/right (azimuth, keyboard short-cuts are left-arrow, right-arrow), up/down (elevation, keyboard short-cuts are up-arrow, down-arrow) and side ways (twist). This can be done from the menu for the currently selected 3-d area, or for all graphs. The keyboard short-cuts only work when a 3D area is selected.
- Palette

The Style entry for the 3D graph on Edit Graphs allows for a choice of palette.

- Contour lines

The same page allows for the addition of contour lines to a surface (but not a triangulation).

12.3 Printing graphs

Select File/Print to print to an attached printer and File/Print Preview to first view the results. The following control of headers and footers is available for printing from File/Print Page Setup:

- Orientation: portrait or landscape;
- Column and scale are ignored, as the graph will fill the page;
- Draw a box;
- The size of margin to use;
- Draw a header for the graph. This is controlled via the Header and Footer boxes.

There are five specials:

- `&[date]` prints the date,
- `&[time]` prints the time,
- `&[page]` prints the page number,
- `&[pages]` prints the total number of pages,
- `&[file]` prints the file name.

Other text in the edit field is printed directly.

Other print setup options are printer specific.

Use File/Print Preview to view the result before printing. Note that colours are shown in the preview, but will come out in grey levels (unless you have a colour printer of course).

12.4 Graphics formats

Graphs can be saved to disk in various formats:

- PDF file (**.pdf**), which is the format used to produce all the graphs in this book;
- Scalable Vector Graphics (**.svg**), which can be inserted into web pages, as well as Microsoft Word and Excel using Insert/Pictures;
- OxMetrics Graphics File (**.gwg**);
- Enhanced metafile (**.emf**); These can easily be inserted into Microsoft Word using Insert/Pictures.
- Encapsulated PostScript (**.eps**);
- PostScript (**.ps**), this is like EPS, but defaulting to a full page print;
- Portable Network Graphics (**.png**), which is a bitmap format that may be useful for insertion in web pages if SVG cannot be used;
- Ox (**.ox** and **.ox.gwg**) code to regenerate the graph can be created by right-clicking on a graph in the document listing and selecting Save Ox code. This can be useful if, after manual adjustments, the graph is to be used as a template for other graphs.

The GWG format is particular to OxMetrics; no other program can read it and no printer can handle it. OxMetrics requires the GWG format, because it needs to be able to allow re-editing of the graph when it is reloaded; the other formats do not store sufficient information.

When you save a graph in any format, the GWG file is automatically saved alongside it. Then, when loading a previously saved PDF file (say), OxMetrics can use the GWG file to reload the graph.

Graphics file formats are discussed in more detail in §9.12–§9.17.

12.5 Saving and loading graphs

Use File/Save As or the diskette toolbar button to save the graph to a file. At the bottom of the dialog is a dropdown listbox in which you select the type of file.

When you save a graph in any format, the GWG file is automatically saved alongside it. Then, when loading a previously saved EPS file (say), OxMetrics can use the GWG file to reload the graph.

Choose File/Open, to open a previously saved graphics file type.

12.6 Saving Ox code of a graph

Right click on the name of a graphics window in the work space gives the option to Save Ox code. This first saves the .gwg file, then uses the gwg2ox package to write the Ox code that recreates the graph.

For Figure 5.2 the code is (with the data cut off for reasons of space):

```
// File generated by gwg2ox
// gwg file   : D:\Waste\gwtutg2.ox.gwg
// gwg status : OK
// gwg objects: 25

#include <oxstd.oxh>
#include <oxdraw.oxh>

DoDrawing()
{
    SetDrawWindow("gwtutg2.ox.gwg");
    decl data = { // cut
        "y1":<890.4495849609375,886.5430297851562,886.3287963867188 >,
        "y2":<908.2122802734375,900.6785278320312,899.7952880859375 >,
        "x3":<908.2122802734375,900.6785278320312,899.7952880859375 >,
        "y3":<890.4495849609375,886.5430297851562,886.3287963867188 >,
        "y4":<.,-3.90655517578125,-0.2142333984375,-1.444091796875 >,
        "y5":<.,-7.53375244140625,-0.88323974609375,-1.31378173828125>,
        "x6":<.,-7.53375244140625,-0.88323974609375,-1.31378173828125>,
        "y6":<.,-3.90655517578125,-0.2142333984375,-1.444091796875 >,
        "end": <>
    };
    DrawAdjust(ADJ_AREAMATRIX, -1, -1, 1, 640);
    DrawTMatrix(0,data["y1"],"CONS",1953,1,4,3,2);
    DrawAdjust(ADJ_INDEX, 1, 0);
```

```

DrawTMatrix(0,data["y2"],"INC",1953,1,4,0,3);
DrawAdjust(ADJ_REGRESSION, 1, 1, 4, 4);
DrawAxisAuto(0,1);
DrawAxisAuto(0,0);
DrawLegend(0,20,20,0);
DrawXMatrix(1,data["y3"],"CONS",data["x3"],"income",1,2);
DrawAdjust(ADJ_SYMBOL, PL_FILLBOX, 100);
DrawAdjust(ADJ_REGRESSION, 4, 1, 4, 4);
DrawAxisAuto(1,1);
DrawAxisAuto(1,0);
DrawLegend(1,20,20,0);
DrawTMatrix(2,data["y4"],"DCONS",1953,1,4,0,2);
DrawAdjust(ADJ_COLOR, -1, 12);
DrawTMatrix(2,data["y5"],"DINC",1953,1,4,0,3);
DrawAxisAuto(2,1);
DrawAdjust(ADJ_AXISGRID, 1, 14, 14);
DrawAxisAuto(2,0);
DrawAdjust(ADJ_AXISGRID, 1, 14, 14);
DrawLegend(2,457,870,0);
DrawXMatrix(3,data["y6"],"DCONS",data["x6"],"DINC",1,2);
DrawAdjust(ADJ_SYMBOL, PL_CIRCLE, 90);
DrawAdjust(ADJ_REGRESSION, 20, 0, 4, 4);
DrawAxisAuto(3,1);
DrawAxisAuto(3,0);
DrawLegend(3,20,20,0);
DrawAdjust(ADJ_AREAMATRIX, 2, 2);
ShowDrawWindow();
}
main()
{
    DoDrawing();
}

```

12.7 Editing graphs

Each graph consists of a collection of objects, which in most cases can be manipulated, moved or deleted. The main one, the area, was introduced in §12.1.

12.7.1 Graph layout

An area defines a rectangle on the paper which has world coordinates attached to it. The concept of areas enables multiple graphs on the paper, up to 36. Areas are allowed to overlap.

The Graph Layout page of the Edit Graphics dialog allows for making adjustments that are relevant for the whole graph. Activate this through Edit/Edit Graph, or by double clicking in an area. You can:

- Change the Areas layout, e.g. for example from 2×1 to 1×2 .
- Box all areas in the current graphics window.
- Change the aspect ratio, for example to half-height graphs as used occasionally in this book.

The aspect ratio of the ‘graph paper’ defaults to 1.5 (15 000 along the x -axis and 10 000 along the y -axis). This usually works very well, but occasionally it may

need changing, for example to make the graphs square on paper. Half-size graphs can also look good in publications. Examples are in Figures 4.9 and 4.8.

- Change margins.
- Adjust all styles to make all lines in all the areas thicker or thinner, or
- to Increase all font sizes.
- Set the PostScript mode: usually plots on screen are colour, but required in gray levels for publication purposes.
- Display mode can be changed from colour to black & white. Occasionally, this can be useful when pasting to another application for printing. OxMetrics always starts in colour mode.
- Paper colour allows the colour to be set for the (normally) white space around the graphics areas, e.g. to a light grey.

12.7.2 Area layout

By default, an area has ‘automatic’ pixel coordinates, which means that, when an area is added, the pre-existing area will automatically shrink to make space. An area can be selected with the mouse, and moved around the paper. This changes the automatic paper positioning into a fixed position. Real world coordinates are also selected automatically, but can be set to fixed values.

The Area Layout page of the Edit Graphics dialog contains the settings that relate to positioning and layout of each individual area. You can:

- change world coordinates: select the desired area and select the Set checkbox for the X or Y coordinates, then adjust the settings. Press All to apply this to all areas.
- reset pixel coordinates to automatic or set specific pixel coordinates.
- change layout, e.g. for example from 2×1 to 1×2 .
- Box the current area, or all areas in the current graphics window.

The Edit menu allows you to rotate 3-dimensional graphs. This applies to the currently selected 3D area, or all 3D areas if none is selected:

- Rotate left or right (from the keyboard use $\text{Alt} <$ or $\text{Alt} >$; this does not require the shift key, so just hold Alt pressed while typing the $<$ or $>$ key). This changes the *azimuth*.
- Rotate up or down (from the keyboard use $\text{Alt} -$ or $\text{Alt} +$). This changes the *elevation*.
- Twist left or right.
- Reset the rotation and twist back to the original values.

12.7.3 Variable against time, scatter, or 3D

The title of this section is derived from the variables being plotted, such as $\text{CONS} \times (\text{time}) \times z$ or $\text{INC} \times (\text{time})$.

The variable (or vector) is the main graphics object: it consists of a sequence of observations, graphed against time or another variable. A variable cannot be moved, nor can observations be moved on screen. There are essentially four types of vectors:

- variable against time;

- variable against another variable;
- variable against another variable by a third variable. The third variable can indicate symbol size, have its value printed, or specify error bars.
- 3D graphs: surface, contour or scatter.

The Lines page of the Graphics Property dialog controls how the vector is displayed. The line type joining the observations can be:

- Line: linked,
- Symbols: not linked, but showing symbols,
- Line and symbols,
- Index: this draws a line from the point to the zero axis (if part of the area), or the edge of the area (towards zero). A non zero base can be specified in the Base for index/bars edit field.
- Index and symbols,
- Bars: this draws a bar to the zero axis (if part of the area), or the edge of the area (towards zero), with the bar centred on the current point. A non zero base can be specified in the Base for index/bars edit field.

Style allows a choice of the colour or pattern for the currently selected entry; the type choice is repeated in the drop-down box. The colour with which lines and symbols are drawn is set in the first column; the line width and pattern is set in the second combo box. In both cases an entry is selected from one of the 16 defined line types. These can be changed in Model/Preferences/Graphics Setup

The observations can be marked by a symbol, and the style and size of that symbol can be set in the third column.

For 3D graphs it is possible to change the colour palette, or add contour lines (but not for a triangulation). The global palette definition is set in Model/Graphics Setup

The entries used in the legend can be changed by renaming the labels of the variables in the X, Y, Z label boxes. If there is no name, that component will be omitted from the legend label for the vector: if all labels are empty, the legend entry will be omitted altogether.

12.7.3.1 Error bars

Error bar options are only shown if the line object has error bars. Then you can switch between bars, bands and fans, choose a color for the bars/bands. A multiplication factor may also be set for error bars and bands. For example, if the data in the Z variable are standard errors, then a factor 2 will show the data plus/minus 2 standard errors (which would be a 95confidence interval if the data come from a normal distribution).

To rename a label for the selected vector, click on Rename, or double-click on the vector in the list box.

12.7.3.2 Regression, Scale

The Regression, Scale page of the Edit Graphics dialog allows adding regression lines to variables, and adjustment of the scaling (relative to the selected variable):

- None removes all scaling.
- Means matches the means of all variables to that of the selected variable: the shift factor will be set to make the means of the variables equal.
- Ranges: the scale factor will be set to give all the variables the same range. Maximum – minimum will now be the same for every variable. Adjustment is made relative to the selected variable.
- Means and Ranges matches means and ranges. Both the shift and scale factors are set.

The scale and shift factor can also be set directly.

The regression options in this page are:

- Number of least squares lines.
By default no least squares lines are drawn; you can select up to 40 lines.
- Sequential versus Recursive regression lines.
The default is sequential regression: the X-axis is divided in s sections (where s is the number of regression lines). Each section gets its own regression line. Recursive regression lines will lead to each subsequent section added on to the previous.
- With or without Projections; projections shows the deviation between the variable and the regression line.
- Style sets the line colour and pattern.

12.7.4 Axes

An X and Y axis (and Z for 3D graphs) is added automatically to each graph. Usually the default suffices, but a large number of adjustments can be made in the Axis entries of the Edit Graphics dialog.

12.7.4.1 Settings for non-default labelling

Each axis consists of (values are in world coordinates):

- Minimum and Maximum;
- First large tick: the value at which the first large tick mark appears;
- Intervals for large ticks: the gap between large tick marks;
- Intervals for small ticks: the gap between small tick marks;

These values are all set to a default, but can be changed when Default labelling is unchecked.

12.7.4.2 Location and transformation

The location and transformation settings further position and define the axis:

- Anchor: determines where the axis is anchored, e.g. for an X -axis this is a Y value (changing the anchor would move the X -axis up or down). Minimum anchors at the left (bottom) of an area, Maximum at the right (top). Use specified anchor allows anchoring anywhere in the graph. For 3D graphs, the X axis is anchored at an Y and Z value, Y at X , Z and Z at X , Y .
- Transformation

- The Axis type:
Normal the default axis without transformation;
Log (data is in logs) when the data is in natural logarithms, use this to show the original values on the axis;
Log10 (data is in log10) when the data is in logarithms of base 10, use this to show the original values on the axis;
Use specified shift and scale allows the shift and scale factors to be set;
Dates when the values on the axis are to be interpreted as date values.

12.7.4.3 Style

The following adjustments can be made to an axis (all adjustments apply to the current axis only):

- hide an axis by checking Hide. (Axes can also be deleted, but when an area has no *X* or *Y* axis and is redrawn, that axis will be added.)
- No base line removes the line, leaving only tick marks.
- No small ticks removes the small tick marks.
- Line at $y=0$ is only for *X*-axes, and uses the same line colour and type as the grid. By default a line showing zero is added when both negative and positive data occur.
- Grid attaches a grid, perpendicular to the selected axis. The grid is at the large tick marks. The grid colour and type is set in Graphics Setup, the default is that of line colour and type 14.
- The Tick Size. This sets the small tick size, the large tick size is twice the small tick size.
- The Font Size sets the size of the axis labels.

12.7.4.4 Label style

- Labels other side puts the labels on the other side of the axis.
- Rotate labels writes text of labels along the *Y*-axis (*Y* axis), see e.g. [Figure 5.4](#), or at a right angle (*X*-axis).
- Centre dates puts the labels in between the tick marks. For example for annual data, the label 1960 is centred at 1960 by default. With Centre dates it is set half way between 1960 and 1961. This option is for *X* axes only. This does not work when the axis is dated.

Note that you may have as many axes as you wish. Axes can be selected on screen with the mouse.

12.7.5 Legend style

Legends are added automatically in an appropriate size, unless the legend font gets too small. In that case (normally with more than sixteen areas), the legend will be omitted. In addition, a legend can be hidden, and subsequently unhidden.

Legends can be selected with the mouse, and moved around. Deleting the legend results in it being hidden. The available options in the Legend style page of the Edit Graphics dialog are:

- Hide, legends are never completely deleted.
- Auto resize, by default, legends are reduced when the area gets smaller. When the text gets too small, the legend is hidden.
- Boxed draws a box around the legends.
- Default box size will be unchecked if the box is resized with the mouse. When it is checked, the box size is left to OXMetrics.
- Columns sets the number of columns for the labels; the default is two.
- Font size determines the size of the label. This is reduced further when Auto resize is set.
- Transparent is the default. In that case, legends are drawn below data plots, with the exception of shading and separately drawn symbols. Opaque legends, on the other hand, are drawn on top of everything else.

12.7.6 Histogram

The only available options for histograms is to select a line color/pattern for the outside line and for the area inside the boxes.

12.7.7 Copy properties to other areas

Some aspects of one area can be copied to all other areas in the graph:

- Pixel coordinates: width and height or reset the default;
- World coordinates: X or Y;
- Axes: labelling, scale type or label style;
- Legend style;
- Histogram style.

12.7.8 Text

Text can be added from the Edit menu using Add Text. Text belongs to an area, in which case it is deleted if the area is deleted (or copied when the area is copied). Text entered outside any area will be part of area 0. More than one line of text can be entered at a time. Text can also be rotated, by specifying the angle in degrees. This does not work so well for multiple line text blocks. Text entered immediately above the graph will have the title property, which means that it is moved with the area when the area is moved.

Text uses a \LaTeX -style of formatting, see §12.10.

12.7.9 Lines and symbols

Lines can be added from the Edit menu using Draw. Line coordinates may be specified in world coordinates (the default when the line is drawn inside an area), in which case

the line is deleted if the area is deleted (or copied when the area is copied). Alternatively, lines may be specified in pixel coordinates. The appearance of a line can be changed to a box or a solid box.

12.7.10 Adding, moving and deleting objects

Objects are added from the Edit menu. Most objects can be selected with the mouse, and moved and deleted subsequently.

12.7.11 Pointing

The status bar shows the graphics coordinates of the the mouse.

12.7.12 Graphics setup

Persistent setup for graphics is set through the Model/Preferences/Graphics Setup dialog.

12.8 Copy and paste

There is a distinction between use of the clipboard inside OxMetrics, and that for external use. There are three copy commands:

1. Copy is for internal use.
When no area is selected, the whole graphics window is copied. When an area is selected, only that is copied to the clipboard. These clipboard items are not recognized by other programs.
When a OxMetrics graphics format is on the clipboard, it can be added to an existing graph (no area selected) or to an existing area (which must be selected). Copying in the internal format is delayed, which means that the full copying is not actually performed until a paste is requested. This implies that when a graphics window is closed it is not available for pasting anymore.
2. Copy Bitmap to Clipboard copies the whole graphics window in bitmap format. This can be pasted into other programs that recognize bitmaps.
3. Copy Metafile to Clipboard copies the whole graphics window in EMF format, from which is a vector format that can be pasted into other programs.

It is not possible to paste from other programs into OxMetrics graphs, with the exception of text.

12.9 Graphs and sample selection

All types of graphs will use all valid observations, just skipping over missing values.

12.10 Text formatting

Text uses a \LaTeX -style of formatting, which allows for complex mathematics be used to annotate graphs. OxMetrics implements a subset of the normal \LaTeX commands, and mathematics is typeset using the Symbol font.

Mathematics is enclosed in $\$ \dots \$$; superscripts and subscripts are only allowed in math mode. A superscript is written as $\text{\textasciitilde}\{ \dots \}$, a subscript as $\text{_}\{ \dots \}$. For example, $\$ \text{\hat} \text{\epsilon} \text{_}\{i-1\} \text{\textasciitilde}\{k+1\} \$$ gives ϵ_{i-1}^{k+1} . Tables 12.1 to 12.6 list the mathematics symbols which are available in OxMetrics.

The entries in Tables 12.7 to 12.8 do not require math mode. Note that, to write $\$ \text{_} \text{_} \{ \} \%$ you must enter $\$ \text{_} \text{_} \text{_} \text{_} \{ \} \%$ respectively ($\text{_}\text{_}$ for a backslash is not standard in \LaTeX , where it denotes a line break).

Table 12.1 Greek symbols

α	<code>\alpha</code>	θ	<code>\thetaeta</code>	o	<code>o</code>	v	<code>\upsilon</code>
β	<code>\betaeta</code>	ϑ	<code>\varthetaeta</code>	π	<code>\pi</code>	ϕ	<code>\phi</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	φ	<code>\varphi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	χ	<code>\chi</code>
ε	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	ψ	<code>\psi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ω	<code>\omega</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>		
η	<code>\eta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>		
Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		

Table 12.2 Arrows

\rightarrow	<code>\arrowext*</code>	\Leftarrow	<code>\Leftarrow</code>	\Rightarrow	<code>\Rightarrow</code>
\downarrow	<code>\downarrow</code>	\leftarrow	<code>\leftarrow</code>	\rightarrow	<code>\rightarrow</code>
\Downarrow	<code>\Downarrow</code>	\Leftrightarrow	<code>\Leftrightarrow</code>	\Uparrow	<code>\Uparrow</code>
		\leftrightarrow	<code>\leftrightarrow</code>	\Uparrow	<code>\Uparrow</code>
		\leftarrow	<code>\leftarrow\arrowext</code>		

* indicates that the command is not a standard \LaTeX command.

Table 12.3 Mathematics accents

\acute{a}	<code>\acute{a}</code>	\dot{a}	<code>\dot{a}</code>	\grave{a}	<code>\grave{a}</code>	\tilde{a}	<code>\tilde{a}</code>
\bar{a}	<code>\bar{a}</code>	\ddot{a}	<code>\ddot{a}</code>	\hat{a}	<code>\hat{a}</code>		

Table 12.4 Foreign and accented characters

å	\aa	æ	\ae	ø	\o	ß	\ss
Å	\AA	Æ	\AE	Ø	\O		
á	\'a	â	\^a	ã	\~a	à	\.a
à	\'a	ä	\"a	ā	\=a		

Table 12.5 Mathematical symbols

\aleph	\aleph	\exists	\exists	\mid	\mid	\rfloor	\rfloor
\angle	\angle	\forall	\forall	\neg	\neg	\sim	\sim
\approx	\approx	\geq	\geq	\neq	\neq	\spadesuit	\spadesuit
\ast	\ast	$>$	\gt* or >	\ni	\ni	\subset	\subset
\bullet	\bullet	\heartsuit	\heartsuit	\oplus	\oplus	\subseteq	\subseteq
\cap	\cap	\Im	\Im	\otimes	\otimes	\sum	\sum
\cdot	\cdot	\in	\in	∂	\partial	\supset	\supset
\clubsuit	\clubsuit	∞	\infty	\perp	\perp	\supseteq	\supseteq
\cong	\cong	\int	\int	\pm	\pm	$\sqrt{}$	\sqrt{}
\cup	\cup	\langle	\langle	\prime	\prime	\therefore	\therefore
$^\circ$	\degree	\lceil	\lceil	\prod	\prod	\times	\times
\diamond	\Diamond	\dots	\ldots	\propto	\propto	∇	\nabla
\diamondsuit	\diamondsuit	\leq	\leq	\rangle	\rangle	\vee	\vee
\div	\div	\lfloor	\lfloor	\rceil	\rceil	\wedge	\wedge
\emptyset	\emptyset	$<$	\lt* or <	\Re	\Re	\wp	\wp
\equiv	\equiv						

* indicates that the command is not a standard L^AT_EX command.

Table 12.6 Log-like symbols

arccos	\arccos	csc	\csc	ker	\ker	min	\min
arcsin	\arcsin	deg	\deg	lg	\lg	Pr	\Pr
arctan	\arctan	det	\det	lim	\lim	sec	\sec
arg	\arg	dim	\dim	lim inf	\liminf	sin	\sin
cos	\cos	exp	\exp	lim sup	\limsup	sinh	\sinh
cosh	\cosh	gcd	\gcd	ln	\ln	sup	\sup
cot	\cot	hom	\hom	log	\log	tan	\tan
coth	\coth	inf	\inf	max	\max	tanh	\tanh

Table 12.7 Miscellaneous symbols

©	\copyright	¶	\P	§	\S
Euro*	\euro	£	\pounds	TM	\trademark*
f	\florin*	®	\registered*		
\$	\\$	\	\	-	_
{	\{	}	\}	%	\%

* indicates that the command is not a standard L^AT_EX command.

Table 12.8 Fonts and sizes

bold	<code>\bf</code>	roman	<code>\rm</code>	small	<code>\small</code>
<i>bold italic</i>	<code>\bi</code>	sans serif	<code>\sf</code>	large	<code>\large</code>
font 0 to 9	<code>\f0 – \f9</code>	typewriter	<code>\tt</code>	normalsize	<code>\normalsize</code>
<i>italic</i>	<code>\it</code>				

Chapter 13

OxMetrics Data Management

13.1 Creating data

File/New allows for the creation of a new database. All variables in a database have the same frequency. A sample period is required, but can be extended or changed later.

13.2 Database font

View/Increase Text Size and View/Decrease Text Size are useful to select a smaller or larger font size for displaying the database on screen.

13.3 Database description

A database can have a description for documentation purposes, and separate from the variable description. Only .in7/.bn7 data files can preserve the description.

13.4 Printing data

Use File/Print when a database has the focus to print the database. File/Print Preview is available to view the result before printing.

Select File/Print to print to an attached printer and File/Print Preview to first view the results. The following control of headers and footers is available for printing from File/Print Page Setup:

- Orientation: portrait or landscape;
- Number of columns on the printed page;
- Scale for printing: 100 is full scale, 90 smaller, etc.;
- Draw a box;
- The size of margin to use;

- Write a header and footer. This is controlled via the Header and Footer boxes. There are five specials:
 - `&[date]` prints the date,
 - `&[time]` prints the time,
 - `&[page]` prints the page number,
 - `&[pages]` prints the total number of pages,
 - `&[file]` prints the file name.Other text in the edit field is printed directly.

13.5 Data formats

The data format specific to OxMetrics has the **.oxdata** extension. In addition, OxMetrics can read and write human-readable files, **Excel** spreadsheet files, comma-separated files, and Stata files.

See Chapter 9 for detailed information on the file formats supported by OxMetrics.

13.6 Summary statistics

This command prints the name of the database, the sample period and number of variables to the Results window. For each variable in the database the name is printed as well as:

leading sample – the first valid sample period without missing values,
#obs – number of valid observations,
#miss – number of observations with missing values,
minimum – minimum value,
mean – mean of valid observations, see (8.1),
maximum – maximum value,
std.dev – standard deviation of valid observations, see (8.1).

13.7 Saving data

Use File/Save As or the diskette toolbar button to save the data to a disk file. At the bottom of the dialog is a dropdown listbox in which you select the type of file.

13.8 Navigation and editing

A database cursor indicates which is the current observation. The cursor can be moved from the keyboard or using the mouse. A block of observations can be selected (highlighted) for:

- copying to the clipboard;
- as a destination for a block from the clipboard;

- to set to a single value: press **Enter** to specify the value, whence you will be prompted whether the value should be applied to the whole selection.

Press **Enter** or double click on an observation to edit the value. Single click on a cell with the cursor will give access to the inline editor. Algebra and the Calculator allows manipulation of variables through mathematical expressions.

13.9 Renaming variables

Press **Enter** or double click on the variable name in the database to change the name or documentation. When in the Calculator or the Algebra Editor, click **Ren** when a variable is selected in the list box to rename.

13.10 Deleting variables

Click on the variable name in the database to select the variable (or a group of variables), then press the **Del** key to delete (confirmation of the deletion will be required). When in the Calculator, press the **Del** button when a variable is selected in the list box to delete, confirmation will be required again.

13.11 Reordering variables

The variables listbox in the Calculator and Algebra Editor allows variables to be picked up with the mouse and moved down or up in the listbox.

13.12 Adding variables

Double click on an empty field in the database next to an existing variable to create a new variable. Alternatively use **Edit/New Variable**.

13.13 Extending or reducing the sample period

Double click on an empty field in the database below an existing variable to activate the **Extend Database** sample dialog. Alternatively use **Edit/Extend sample**. Enter a negative value to reduce the sample size.

The **Change Sample** option on the **Edit** menu allows the following actions:

- Add observations at the start;
- Add observations at the end;
- Delete observations at the start;
- Delete observations at the end.

13.13.1 Changing the sample period

The Change Sample option on the Edit menu allows changing the start year and period, as well as the frequency. The number of observations remains unchanged, and no data will be lost.

13.14 Copy and paste

The standard copy and paste facilities are available, also see §13.8.

13.15 Appending data

A database can be appended from disk to an open database, using Append Data File when right clicking on a database in the workspace. This requires the frequencies to match. OxMetrics will make additional adjustments when these are required:

- the sample of the open database will be extended to encompass that of the appended database;
- if a variable from the appended database already exists in the open database, then the existing variable is updated, except where the appended variable has missing values (for those observations the existing variable is not changed).

13.16 Daily, weekly and timed data

The calendar¹ index used by OxMetrics is the Julian day number. For example, Julian day 2453402 corresponds to 2005-01-31. An optional fractional part specifies the fraction of the day: 2453402.75 corresponds to 2005-01-01T18:00. If the day number is zero, it is interpreted as a time only, so 0.75 is just 18:00 (6 PM).² OxMetrics uses the ISO representation yyyy-mm-dd, and the 24 hour clock for time, hh:mm:ss. hhh. A date with a time are joined by the letter T, as shown above.

In many fields it is possible to enter a date or time, which is then automatically translated to the internal representation. Years with week number are then also recognised, e.g. 1976-W3 returns the calendar index for the Monday of week 3 in 1976. (Week 1 is the first week that contains the first Thursday; or equivalently, the week that contains 4 January.)

OxMetrics uses two methods for dating a database:

Fixed frequency Denoted by a start year, start period and frequency. From this, the date of any subsequent observation can be worked out. For example, if the start year is 2005, the start period 1, and the frequency 4 (quarterly), then the sixth observations is for 2006(2), the second quarter. This works well for monthly, quarterly

¹The calendar is Gregorian from 15 October 1582 onwards, and Julian before (so there is no year 0: year -1 precedes year 1; day 0 is on Julian date 1 January -4713).

²This is similar to how Excel stores dates and times.

and annual data. For undated data we simply set the start year and period to one, and the frequency as well.

Dated The fixed frequency method does not work for weekly or daily data: not every year has the same numbers of observations. The solution adopted in OxMetrics is as follows: an integer value represents a date. A database variable can be classified as of type date, in which case the date is displayed, rather than the underlying value.

The following criteria must be satisfied for a database to be dated:

- the first column must be of type Date,
- the first column holds date values,
- the optional fractional part of this indicates time as a fraction of 24 hours; e.g. 2453372.75 is 18:00 on 2005-1-1 (2005-1-1T18:00 in ISO notation),
- the first and last observation must be valid, i.e. cannot be missing.

Note that the fixed sample period is still available for a dated database, but this is usually set as for an undated database (start year, start period, frequency all set to one).

Also note that the first variable in a dated database (which defines the calendar dates of the observations) is a variable as any other: it can be renamed, transformed, moved, deleted, etc. But some of these will make the database undated again.

See §3.3 and §6.7 for more examples.

References

- Davidson, J. E. H., D. F. Hendry, F. Srba, and J. S. Yeo (1978). Econometric modelling of the aggregate time-series relationship between consumers' expenditure and income in the United Kingdom. *Economic Journal* 88, 661–692. Reprinted in Hendry, D. F., *Econometrics: Alchemy or Science?* Oxford: Blackwell Publishers, 1993, and Oxford University Press, 2000; and in Campos, J., Ericsson, N.R. and Hendry, D.F. (eds.), *General to Specific Modelling*. Edward Elgar, 2005.
- Doornik, J. A. (2006). The role of simulation in econometrics. In T. Mills and K. Patterson (Eds.), *Palgrave Handbook of Econometrics*, pp. 787–811. Basingstoke: Palgrave MacMillan.
- Doornik, J. A. (2021). *Object-Oriented Matrix Programming using Ox* (9th ed.). London: Timberlake Consultants Press.
- Doornik, J. A. and D. F. Hendry (2021). *Interactive Monte Carlo Experimentation in Econometrics Using PcNaive* (5th ed.). London: Timberlake Consultants Press.
- Doornik, J. A. and K. Juselius (2018). *CATS 3: Cointegration Analysis of Time Series in OxMetrics*. London: Timberlake Consultants Press.
- Doornik, J. A. and M. Ooms (2005). Outlier detection in GARCH models. Discussion paper 2005-w24, Nuffield College.
- Doornik, J. A. and M. Ooms (2008). Multimodality in the GARCH regression model. *International Journal of Forecasting* 24, 432–448.
- Engler, E. and B. Nielsen (2009). The empirical process of autoregressive residuals. *Econometrics Journal* 12, 367–381.
- Findley, D. F., B. C. Monsell, W. R. Bell, W. R. Otto, and B.-C. Chen (1998). New capabilities and methods of the X-12-ARIMA seasonal-adjustment program (with discussion). *Journal of Business and Economic Statistics* 16, 127–177.
- Golub, G. H. and C. F. Van Loan (1989). *Matrix Computations*. Baltimore: The Johns Hopkins University Press.
- Granger, C. W. J. (1966). The typical spectral shape of an economic variable. *Econometrica* 34, 150–161.
- Granger, C. W. J. and P. Newbold (1986). *Forecasting Economic Time Series*, (2nd ed.). New York: Academic Press.
- Green, P. J. and B. W. Silverman (1994). *Nonparametric Regression and Generalized Linear Models. A Roughness Penalty Approach*. London: Chapman and Hall.
- Härdle, W. (1990). *Applied Nonparametric Regression*. Cambridge: Cambridge University Press.
- Harvey, A. C. (1993). *Time Series Models*, (2nd ed.). Hemel Hempstead: Harvester Wheatsheaf.

- Hastie, T. J. and R. J. Tibshirani (1994). *Generalized Additive Models*. London: Chapman and Hall.
- Hendry, D. F. and J. A. Doornik (2021). *Empirical Econometric Modelling – PcGive 16 Volume I*. London: Timberlake Consultants Press.
- Koopman, S. J., A. C. Harvey, J. A. Doornik, and N. Shephard (2013). *Structural Time Series Analysis, Modelling, and Prediction using STAMP* (5th ed.). London: Timberlake Consultants Press.
- Laurent, S. (2021). *G@RCH Professional 9* (7th ed.). London: Timberlake Consultants Press.
- Makridakis, S., S. C. Wheelwright, and R. C. Hyndman (1998). *Forecasting: Methods and Applications* (3rd ed.). New York: John Wiley and Sons.
- Nielsen, B. (2006). Correlograms for non-stationary autoregressions. *Journal of the Royal Statistical Society B* 68, 707–720.
- Priestley, M. B. (1981). *Spectral Analysis and Time Series*. London: Academic Press.
- Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Author Index

Bell, W. R. 27

Chen, B.-C. 27

Davidson, J. E. H. 54

Doornik, J. A. 27, 36, 95, 109, 122

Engler, E. 110

Findley, D. F. 27

Golub, G. H. 104

Granger, C. W. J. 52, 106

Green, P. J. 109

Harvey, A. C. 27, 97, 104, 109, 111

Hastie, T. J. 109

Hendry, D. F. 27, 54

Hyndman, R. C. 111

Härdle, W. 108

Juselius, K. 27

Koopman, S. J. 27, 109

Laurent, S. 27

Makridakis, S. 111

Monsell, B. C. 27

Newbold, P. 106

Nielsen, B. 51, 105, 110

Ooms, M. 36

Otto, W. R. 27

Priestley, M. B. 97, 106

Shephard, N. 27, 109

Silverman, B. W. 107, 109

Srba, F. 54

Tibshirani, R. J. 109

Van Loan, C. F. 104

Wheelwright, S. C. 111

Yeo, J. S. 54

Subject Index

- 3-dimensional plots 57, 159
- 3D
 - contour 59, 160, 164
 - palette 59, 159, 164
- Actual series 16, 42, 103, 156
- Add Graph 69
- Algebra 7, 23, 92, 128–142
 - editor 92
 - file (.alg) 122
 - functions 134, 138
 - indexing 132
 - keywords 132
 - operator precedence 130
- Executing code 128, 149
- Implementation 133
- Indicators 135
- Syntax 129
- Variable names 129
- Algebra Constants
 - FALSE 24, 92, 133
 - FREQUENCY 133
 - MISSING 99, 133–135
 - NOBS 93, 133
 - TRUE 24, 92, 133
- Algebra Functions
 - _sortallby() 136, 142
 - _sortby() 96, 142
 - abs() 140
 - acf() 50, 135, 140
 - acos() 140
 - almon() 91, 140
 - asin() 140
 - atan() 140
 - ceil() 140
 - cos() 140
 - cseasonal() 140
 - cum() 91, 94, 140
 - date() 137, 140
 - day() 137, 140
 - dayofweek() 137, 140
 - delete() 140
 - denschi() 93, 140
 - densf() 93, 140
 - densn() 93, 140
 - denst() 93, 140
 - DI() 140
 - diff() 23, 134, 140
 - div() 140
 - dlog() 140
 - dummy() 23, 140
 - dummydates() 140
 - ewma() 137, 140
 - ewmc0() 137, 140
 - exp() 140
 - fabs() 140
 - floor() 140
 - fmod() 140
 - hours() 137, 140
 - II() 140
 - implicit 134
 - indates() 140
 - insample() 24, 92, 98, 99, 137, 140
 - isdayofmonth() 137, 140
 - iseaster() 140
 - lag() 134, 139, 141
 - lag0() 141
 - log() 141
 - log10() 141
 - loggamma() 94, 141
 - LS() 141
 - LT() 141
 - makedate() 141
 - maketime() 141
 - max() 141
 - mean() 141
 - MI() 141
 - min() 141
 - minutes() 137, 141
 - month() 137, 141
 - movingavg() 91, 141
 - movingSD() 141
 - pacf() 141
 - peak() 141

- period() 54, 141
- periodogram() 106, 135, 141
- probn() 141
- QI() 142
- quanchi() 141
- quantf() 141
- quann() 141
- quant() 141
- ranchi() 95, 141
- ranf() 95, 141
- rann() 95, 141
- ranseed() 98, 141
- rant() 95, 141
- ranu() 95, 141
- round() 141
- season() 141
- seasonal() 141
- seconds() 137, 141
- SI() 142
- sin() 141
- smooth_hp() 136, 142
- smooth_np() 137, 142
- smooth_sp() 100, 137, 142
- sort() 135, 142
- _sortallby() 136, 142
- _sortby() 96, 136, 142
- sqr() 142
- stock() 91, 142
- stockv() 142
- tailchi() 95, 142
- tailf() 95, 142
- tailn() 95, 142
- tailt() 95, 142
- tan() 142
- TI() 142
- time() 137, 142
- trend() 98, 135, 142
- trough() 142
- usedata() 129
- variance() 142
- year() 98, 137, 142
- Almon lag 91, 140
- Appending
 - data 83, 146, 175
 - results 146
- ARCH 97
- Area 61
 - Layout 163
- Aspect ratio 63, 162
- Autocorrelation 42
 - function 50, 98, 104, 135, 158
- Autocovariance 104
- Autoregressive
 - moving average (ARMA) 97
 - process 97
- Availability 5
- Axis 74, 165
 - anchor 74, 165
 - font size 166
 - labels 166
 - scaling 165
 - tick marks 165
 - transformation 165
- Labelling 165
- Location 165
 - Rotate — labels 74, 166
- Azimuth 58, 159, 163
- Bandwidth 109
 - selection 100, 107–109, 136, 137
- Batch 7, 143–154
 - editor 7
 - file (.fl) 7, 122
 - running 7
 - syntax 143
- Executing code 143, 149
- Graphics settings 69, 151
- Batch Commands
 - appenddata 146
 - appresults 146
 - break 146
 - chdir 146
 - closedata 147
 - closedrawwindow 147
 - command 147
 - database 147
 - draw 147
 - drawf 147
 - drawptext 148
 - drawtext 148
 - drawtitle 148
 - drawx 148
 - drawz 148
 - exit 149
 - loadalgebra 149
 - loadbatch 149
 - loadcommand 149
 - loaddata 149
 - loadgraph 150
 - module 150
 - package 150
 - print 150
 - printdate 150
 - println 150
 - savedata 150
 - savedrawwindow 150

- saveresults 150
- setdraw 69, 151
- setdrawwindow 152
- show 152
- usedata 152
- Black and white 126
- Black, white, gray 126
- Box plot 50, 111
- Boxed 160, 167, 172
- Boxed graph 17
- Bubble chart 56, 159
- Calculator 21, 90, 128, 174
- Changing sample 82, 83, 175
- χ^2 -distribution 93, 110
- χ^2 -random numbers 96
- Choice variables 87
- Clipboard 20, 24, 47, 63, 78, 83, 168, 173, 175
- Colour 67, 69, 126
- Comma separated file 116
- Conditional statement (Algebra) 24, 92, 131
- Copy properties to other areas 167
- Correlation
 - coefficient 104
- Correlation coefficient 104
- Correlogram 51, 104
- Creating a new variable 76
- Cross-correlation function 51, 105, 158
- Cross-section analysis 136
- CSV (comma-separated) file 75
- Cumulation 142
- Cut and Paste 63, 78, 168, 173, 175
- Daily data 29, 84, 175
- Dark mode 25
- Data
 - accuracy 13
 - appending 82
 - closing 147
 - copying to the clipboard 79
 - description 39
 - documentation 114
 - input 76–80
 - loading 81–82, 149
 - ordered by observation 80, 120
 - ordered by variable 81
 - pasting from the clipboard 79
 - printing 172
 - saving 80–81, 150
 - storage 8
 - with load info 121
- Human-readable — 81, 82, 120, 121
- data.oxdata 13, 90, 113
- Database 7, 76
 - description 172
 - editing 173
 - with dates 84
- Appending data 83, 146, 175
- Changing the sample 83, 175
- Extending the sample 83, 174
- Reset starting date 83, 175
- Undo 24
- Date
 - in database 31, 84, 175
- Algebra functions 137
- Representation 8, 85, 175
- Default bars 107
- Deleting a variable 22, 174
- Density
 - estimation 106
- Density estimation 48
- DHSY 54
- Dialogs
 - Algebra 24, 92
 - Calculator 21, 90, 174
 - Dummy 22
- Create a New Database 76
- Edit Graphics 64
 - Area Layout 62, 163
 - Aspect ratio 52
 - Axes 67, 165
 - Copy properties to other areas 167
 - Error bars 164
 - Graph Layout 63, 162
 - Hide axis 74
 - Histogram 49, 167
 - Label 164
 - Legend style 166
 - Legends 72
 - Lines (Error Bar) 56
 - Match by 73
 - Regression 65
 - Regression, Scale 46, 164
 - Rename 67
 - Scaling 72
 - Style 58, 66, 164
 - Type 65, 164
- Formulate 32
- Graphics 16, 40
 - 3-dimensional plots 58
 - Autocorrelation function (ACF) 41
 - Distribution 41, 43
 - Distribution Options 48
 - Scatter Plot 99
 - Scatter Plot Options 45, 47
 - Select type of graph 40

- Two series by a third 55
- Graphics Setup 69, 168
 - Write as batch 151
- Graphics Text 71, 167
- Lines and symbols 71
- Modelling
 - Formulate 32
- New 76
- Open File 81, 82
- Preferences 25
- Settings GARCH 34
- Tail Probability 95
- Test 35
- Variable Description 14, 77
- Directory structure 4
- Dummy variable 22, 91, 134
- Durbin's method 104
- Editor
 - Algebra — 92
 - Batch — 7
- Elevation 58, 159, 163
- Enhanced meta file (.emf) 127, 168
- Error
 - bands 164
 - bars 164
 - fans 57, 164
- Error messages 92
- Exponentially-weighted moving average (EWMA) 111, 137
- Exponentially-weighted moving correlation 111, 137
- Extending sample 83, 174
- F-distribution 93, 110
- F-random numbers 96
- File
 - extensions 8
 - extensions in Windows Explorer 8, 13
 - names 8
- Folder structure 4
- Font 127, 166, 167, 172
- Fourier decomposition 52, 158
- Frequencies 49
- Frequency 115, 133
- Gamma function 93
- Generalized cross validation 108, 110
- Graph 61
 - Ox code 161
 - areas 61
 - axes 165
 - coordinates 18, 61, 155
 - copying 20, 63, 168
 - layout 52, 126, 162
 - paper 61, 155
 - pasting 20, 63, 168
 - printing 20, 160
 - saving 20, 150, 161
 - scale 52, 162
 - scaling 164
 - text 71, 167
 - view 63
 - ics window 156
- 3-dimensional plots 57, 159
- Actual series 16, 42, 103, 156
- Batch control 151
- Boxed — 17
- Distribution 47, 157
- Grid lines 67, 166
- Lines 167
- Multiple —s 18
- Multiple scales 74
- Multiple series 44, 157
- QQ plots 53, 158
- Renaming variables 66
- Scatter plot 45, 103, 107–109, 157
- Shading 43
- Time-series: ACF etc. 50, 158
- Transformed series 42, 156
- Two series by a third 54, 159
- Undo changes 20
- Graphics
 - Adding to — 69
 - Context menu 70
 - Copy properties to other areas 167
 - Delete From Graph 70
 - Drawing 70
 - Removing from — 69
- Grey levels 126, 160
- Grid 166
- Help 5
- Histogram 48, 106, 167
- Hodrick–Prescott filter 99, 109, 110, 136
- Holt's method 111
- Human-readable file 81, 82, 120, 121
- Integration 91, 142
- Keep Graph 63, 156
- Kernel 107, 108
 - smooth 47, 108, 137
 - Epanechnikov — 108
 - Normal — 106
- Kurtosis 48
- Label

- Names used in — 164
- Lags 21, 22, 141
- Latex 169
- Leads 141
- Legend 72
 - style 166
 - Rename variable 164
 - Renaming variables 66
- Licensing code 12
- Line 163
 - colour 64, 67, 69
 - drawing 167
 - style 164
 - type 64, 69, 164
- Long memory 50
- Lotus 116
- Matching
 - means 164
 - means and ranges 44, 72
- Matrix file (.mat) 122
- Menus
 - Edit menu
 - Add Text 71, 167
 - Change Sample 82, 83
 - Change sample 174
 - Copy 82
 - Database Description 172
 - Draw 70
 - Draw Line 70
 - Edit Field 173
 - Edit Graph 162, 163
 - Lines and symbols 167
 - New Variable 76, 174
 - Reset Starting Date 83, 175
 - File menu 13
 - New 76, 172
 - New Data Plot Window 45, 63, 156
 - Open 81, 82, 161
 - Print 160, 172
 - Print Page Setup 160, 172
 - Print Preview 160, 172
 - Save As 80, 161, 173
 - Model menu
 - Algebra 92
 - Calculator 21, 90
 - Graphics 15, 156
 - Model 31
 - Preferences 68, 164, 168
 - Tail probability 95
 - Test 35
 - View menu
 - Increase Text Size 172
 - Point 168
 - Summary statistics 173
- Microsoft Excel 20, 75, 80, 81, 160
 - data file 81, 116
 - Copying from — 82
- Microsoft Word 20, 127, 160
- Missing value 42, 47, 77, 104, 107, 109–111, 115, 116, 118, 121, 129, 130, 133, 137, 156, 157, 168
- Module 27–36, 150
- Modules
 - Model 31
- Moving
 - average 91, 141
 - standard deviation 141
- Multiple series 44, 157
- New Data Plot Window 63
- Non-parametric
 - density estimation 106
 - regression 108, 109, 137
- Normal
 - distribution 92, 93, 107, 110
 - kernel 106
 - random numbers 96
- Numerical
 - accuracy 95
 - integration 94
- Output storage 122
- Ox 6
 - file (.ox) 122
- OxMetrics data file (.in7/.bn7) 8, 113
 - Binary file (.bn7) 115
 - Generation 115
 - Group 115
 - Information file (.in7) 80, 113, 116
- OxMetrics data file (.oxdata) 113
- OxMetrics data file (.oxdata)) 4, 80
- OxMetrics graphics file (.gwg) 122, 161
- OxPack 6
- OxRun 6
- Package 150
- Paper colour 163
- Partial autocorrelation function 51, 104, 158
- Parzen window 106
- Paths in Batch files 146
- PcGive 12, 31, 48, 50, 153
- PDF
 - file (.pdf) 124
- Periodogram 52, 104, 105, 158
- Pixel coordinates 151, 155, 163, 168
- Pointing 168

- PostScript
 - colour model 126
 - file (.eps) 124
 - file (.ps) 127
 - mode 126, 163
 - Full page graph 127
- Precedence (Algebra) 130
- Preferences 25
- Print
 - Orientation 160, 172
- Printing 160, 172
- Probability functions 142
- Projections 165
- QQ plot 48, 53, 110, 158
- Quantile plot 53, 110, 159
- Random number generators 95, 141
- Recession shading 43
- Register OxMetrics 6, 12
- Regression 165
 - lines 45, 65, 107, 165
 - Recursive — lines 107, 165
 - Sequential — lines 107, 165
- Rename variable in legend 164
- Renaming variables 14, 22, 174
- Reordering variables 22, 174
- Results file (.OUT) 122
- Sample period 8, 115
 - for graphs 168
- Scaling 44, 72, 164
- Scatter plot 45, 99, 103, 107–109, 157, 163
- Scientific Word 127
- Seasonal 55, 131, 141
 - sub-plot 50, 55, 158
- Smoothing 99, 136
- Sorting 135
- Spectral density 52, 104, 106, 158
 - Bias 106
- Spline smooth 46, 99, 109, 137
- Spreadsheet files 81, 116
- Stata data file (.dta) 121
- Status bar 9, 14, 18
- Stock 91, 142
- Sub-sample evaluation 90
- Summary statistics 173
- Surface
 - from scatter 57
 - from table 59
- SVG
 - file (.svg) 124
- Swap
 - variables in a database 22, 174
- Symbol
 - size 164
 - type 164
- t-distribution 93, 110
- t-random numbers 96
- Text 71, 167
 - LaTeX-style — 169
 - Rotate — 167
- Time
 - Algebra functions 137
 - Representation 8, 85, 175, 176
- Time-series: ACF etc. 50, 158
- Toeplitz matrix 109
- Tool bars 9
- Toolbar
 - Model 31
- Transformed series 42, 156
- Trend 135, 142
- Tutorial data set (data.oxdata) 13
- Twist 58, 159, 163
- Two series by a third 54, 159
- Uniform
 - distribution 110
 - random numbers 96
- Upgrades 7
- Value labels 87
- Variable
 - creation 76
 - names 114
 - names in algebra 129
 - Sorting a — 135, 136
- Variables
 - Adding — 82, 174
 - Copying — 82, 175
 - Deleting — 22, 174
 - Renaming — 14, 22, 174
 - Reordering — 22, 174
- Vector 163
- Weekly data 29, 84, 175
- Windows 13
- Windows bitmap 168
- Windows Explorer 7, 8, 13
- WKS,WK1 files 116
- Workspace 25
- World coordinates 155, 163, 167
- XLS files 116
- XLSX files 81, 116
- Yule–Walker equations 104