8 The Software JMulTi

Markus Krätzig

8.1 Introduction to JMulTi

8.1.1 Software Concept

This chapter gives a general overview of the software by which the examples in this book can be reproduced; it is freely available via the Internet.[1] The information given here covers general issues and concepts of JMulTi. Detailed descriptions on how to use certain methods in the program are left to the help system installed with the software.

JMulTi is an interactive JAVA application designed for the specific needs of time series analysis. It does not compute the results of the statistical calculations itself but delegates this part to a computational engine via a communications layer. The range of its own computing functions is limited and is only meant to support data transformations to provide input for the various statistical routines.

Like other software packages, JMulTi contains graphical user interface (GUI) components that simplify tasks common to empirical analysis – especially reading in data, transforming variables, creating new variables, editing data, and saving data sets. Most of its functions are accessible by simple mouse interaction.

Originally the software was designed as an easy-to-use GUI for complex and difficult-to-use econometric procedures written in GAUSS that were not available in other packages. Because this concept has proved to be quite fruitful, JMulTi has now evolved into a comprehensive modeling environment for multiple time series analysis. The underlying general functionality has been bundled in the software framework JStatCom, which is designed as a ready-made platform for the creation of various statistical applications by developers.

[1] The homepage is http://www.jmulti.de

8.1.2 Operating `JMulTi`

Much effort has been invested in making the use of `JMulTi` as intuitive as possible. Therefore it is not necessary to read this chapter before using the program. Some options are explained in detail, however. Thus it may be a good strategy to read this chapter after a first acquaintance with the software.

`JMulTi` is a `JAVA` program and consequently relies on the availability of an appropriate `JAVA` Runtime Environment (JRE)[2] that needs to be installed as well. Users new to `JAVA` programs should be aware that the user interface looks slightly different from the one of native programs and is sometimes a bit less responsive. This does not mean that the speed of the statistical calculations is slower than with other software packages. The difference results from the way the `JAVA` GUI components are painted on the screen. The speed of the calculations is only lowered by the overhead imposed by the communications interface between `JMulTi` and the computational engine. This is a tolerable fixed cost that does not affect the performance of the algorithms.

In the following sections some conventions for numbers, dates and variable names, data input, and data handling will be discussed. Section 8.4 describes how various time series operations can be carried out, and Section 8.6 gives a very general overview of the underlying software framework. This is especially meant for developers who consider writing graphical user interfaces for their algorithms.

8.2 Numbers, Dates, and Variables in `JMulTi`

8.2.1 Numbers

`JMulTi` stores numbers as 64-bit, double-precision, floating point values as defined in IEEE (754–1985). Special values are `NaN`, `Infinity` and `-Infinity`. `NaN` stands for Not-a-Number and is used to code missing values in data files. Infinities are not allowed for data points but occur in the definition of intervals or for coding special values.

To input numbers that are used by a procedure, `JMulTi` provides special text fields with input-validating capabilities. Usually these text fields display a reasonable default value. The decimal delimiter is a point. The number displayed is rounded to the precision required for the respective purpose. Very often the number of fraction digits is zero, which allows only for integer input. If the given value cannot be parsed to a number or if it is not in the allowed range, a descriptive message will pop up and any other action is canceled. It is always possible to input values coded in exponential notation (e.g., `1.234e-10`). The respective text field formats the input value in a way that makes sense for the given context.

[2] The Sun JRE can be downloaded from http://www.java.sun.com.

Range of numbers. Sometimes it is necessary to input a range defined by two numbers. `JMulTi` also provides special text fields for this purpose that make sure that the given range is valid and within the enclosing interval. Text fields with this behavior are marked with angular brackets. The two numbers defining the range must be separated by a comma.

8.2.2 Numbers in Tables

`JMulTi` often uses tables to display an array of numbers. The tables frequently have a right mouse pop-up menu for useful tasks on that table. The default menu for number tables allows the precision and the notation of the numbers displayed to be changed. If a table is editable, the same rules hold as for number text fields.

In certain contexts, such as for the specification of subset restrictions, the number table renders certain values with special symbols (e.g., `'*'` or '`---`'). Usually these tables can be edited by just clicking on a cell such that the value switches to the next valid one and the symbol changes accordingly.

8.2.3 Dates

In `JMulTi` dates are objects defined by a main period, a frequency, and a subperiod. They can be specified in various ways with the respective text fields for date input. All three pieces of information must be retrieved from the input. The following formats are recognized, where `D` stands for the main period, `S` for the subperiod, and `F` for the frequency or periodicity:

- `D S/F`, for example `1960 1/6`. This format is always possible and is used as default format if no special identifier for the frequency is defined.
- `D 'Q' S`, for example `1960 Q 1`. This format is used for quarterly data.
- `D 'M' S`, for example `1960 M 11`. This format is used for monthly data.
- `D`, for example `1960`. This format is used for annual data.
- `D ['I','II']`, for example `1960 II`. This format is used for half-yearly data.
- `D.S`, for example `1960.1`. This format is just for input convenience and can be used when the program already knows the frequency from the context, which is often the case.

Date input must conform to the given rules and is validated against a range defined by two dates. If the input is wrong in any way, the previous value is restored and a descriptive message is shown.

Range of dates. Sometimes it is necessary to define a time range by specifying its first and the last date. `JMulTi` provides special text fields for that purpose as well. Like the ones for number ranges, they are marked with angular brackets and validated against an enclosing date range. The two dates must be separated by a comma.

8.2.4 Variable Names

As a general convention throughout the program, variable names can contain letters, numbers, and '_' but must start with a nonnumber. For example, `_invest`, `invest` and `i2` would be valid names, whereas `2i`, `gov exp` or `cons+inv` would be invalid. This is enforced wherever variable names can be set within the program but must also be followed when specifying variable names in a data file. Variable names are case insensitive, meaning that two variable names differing only in the case of one or more letters are considered equal.

8.3 Handling Data Sets

8.3.1 Importing Data

`JMulTi` is meant to be used for empirical data analysis. Therefore the first step is always to read in a data set that is stored in a file. In case the available data is in a format that is not directly readable by `JMulTi`, it should always be possible to transform the data easily into one of the accessible formats.

When the data are read in, `JMulTi` automatically identifies dummy and trend variables as deterministic. In general, reading in variables that are either an intercept, a deterministic trend, or seasonal dummy variables is not recommended because `JMulTi` offers automatic generation of these variables wherever it is necessary for a certain procedure.

8.3.2 Excel Format

`JMulTi` can read in Microsoft® Excel 97 files. Earlier versions are not supported, whereas the file format did not change with later versions of Excel. There is an appropriate file filter for `*.xls` files. When the file is opened, a dialog asks for some additional information. Only one Excel table can be read in at a time.

The Excel file should have the variable names in the first row and the numeric values starting in one of the next rows. The parser also recognizes cells that contain formulas and evaluates them. If `JMulTi` finds no variable names, it creates defaults from the filename and an index. Other cells with nonnumbers will be treated as missing values and coded as `NaN`. Whether the decimal delimiter is a comma or a point depends on the local settings. A number is recognized if Excel has stored the content of the cell as a number.

8.3.3 ASCII Format

An ASCII data file example with an optional description looks like this:

```
/*seasonally adjusted, West Germany:
fixed investment, disposable income, consumption expen-
ditures*/
180 451 415
179 465 421
...
```

The file should contain the data of each variable in a column. Missing values may be coded with `NaN`. It makes no difference whether the numbers are coded with a decimal comma or a decimal point. The exponential notation (e.g., `1.23e-4`) is recognized as well. When the file is opened, a dialog asks for additional information. It is possible to add a description enclosed by `/* ... */` to the data set somewhere in the file.

8.3.4 `JMulTi` .dat Format

`JMulTi` has a file format that is a slight extension of the ASCII format and allows for easy data recognition without further user interaction. The following is an example of a .dat file with an optional description:

```
/*seasonally adjusted, West Germany:
fixed investment, disposable income, consumption expen-
ditures*/
<1960 Q1>
invest income cons
180 451 415
179 465 421
...
```

where the start date enclosed by <. . . > must conform to the rules described in Section 8.2.3.

8.4 Selecting, Transforming, and Creating Time Series

8.4.1 Time Series Selector

Once a data set has been read in, the single variables can be accessed via the time series selector. All time series appear in a list and can easily be combined even if they stem from different files. It is not possible, however, to select time series together for some operation that have a different periodicity. Time series of the same periodicity must have at least two overlapping observations in a common time range to be eligible for common operations. Various tasks on the selected variables can be accessed by a menu that pops up when the user

P1: IML
CB698-Driver CB698-LUTKEPOHI CB698-LUTKEPOHI-Sample.cls August 25, 2006 17:44

right clicks over selected variables in the time series selector. The tasks in the pop-up menu do not take the selected range into account but always operate on the whole range of the selected time series.

The time series selector is used to select the set of variables, the order of the variables, and the time range for the various econometric methods and models. The respective selection is only valid within its context (e.g., for unit root tests). The selection made there has no influence on the model selection – for instance for a VAR model. The selection mechanism is also adjusted to its context, which means that only one variable can be selected for unit root tests, whereas for VAR modeling the number of variables is not restricted.

Sometimes the ordering of the variables is important for a model or for analysis. The selector uses the order in which the variables have been clicked on. For multiple selection, it is necessary to keep either the `Shift` or the `Ctrl` button depressed while the mouse is clicked over a series. The selected variables are displayed in the correct order in the control area of the selector after the selection has been confirmed.

The time range of the selection can be adjusted by editing the respective text field. The valid range is the largest common range of all selected time series, where `NaN` missing values have been automatically truncated from the beginning and the end of each series. The smallest legal range must contain two observations. Once a range selection is made, this selection is kept as long as it is valid. The maximum possible range can easily be set via a button. This mechanism enables the user to keep an edited time range as long as possible but allows a quick switch back to the maximum range as well.

In general, all variable names in `JMulTi` are case insensitive, which means that there is no distinction between, say, `GNP` and `gnp`. However, the variables in the selector are displayed as they have been read in or named with the case of the characters unchanged.

The time series selector displays variables with their names and with a symbol tagging their property. Within the context of time series analysis the three possible properties are `endogenous`, `exogenous`, and `deterministic`. Through these properties variables can be grouped and treated differently by a certain procedure or model. It is possible to change the property of a variable via the right mouse menu. The following tasks are available through this pop-up menu as well:

Delete. All selected time series can be deleted. This operation is also accessible by pressing the `Del` key.

Rename. The first selected time series can be renamed. The new name must be unique among all time series in the list. The rules for variable names described earlier apply.

Creating dummies. For dummy creation, first a variable needs to be selected as a reference for the time range and the periodicity. A dialog allows specification of the date of the impulse or the range of a shift. As already mentioned, seasonal dummies should usually not be added to the data set explicitly because they can be created for each specific analysis where they are needed.

Transforming time series. Some common transformations are also accessible via the transform dialog. New variables are generated with a descriptive suffix. If `logarithm` is selected, it is always applied first. The selected original variables do not change.

Editing time series. The selected time series can be edited as well. Changes are only stored if the dialog is confirmed with `OK`. If series with different ranges are selected, the missing values at the beginning or the end are displayed with `NaN`.

Exporting data. It is possible to combine the selected time series with a new data set that will be saved in the .dat format described earlier. The data set can then be read in again without further interaction.

8.4.2 Time Series Calculator

The time series calculator is a very flexible tool to create new variables by combining existing time series with arithmetic operations and functions. It provides the minilanguage `TSCalc` that operates with one-dimensional arrays and scalars. The operation is as follows:

1. First one has to select the variables to be combined in the time series selector.
2. The names appear in the list `Available Variables` and are put into the variable space of the calculator.
3. One can write one or more commands to the command area and execute them with `Execute` or `Ctrl+E`. Newly created variables appear in the list of available variables.
4. Finally, the selected variables in `Available Variables` can be added to the workspace with `Add Selected`.

The syntax of `TSCalc` is very easy, and only a few simple rules apply:

- New variables can be defined with
  `newvariable = some expression;`.
- Several commands can be executed at once by separating them with `';'`.
- The content of a variable can be printed out by just writing the variable name to the command line.
- The conventions for variable names hold as described in Section 8.2.4.

- All array operations are applied elementwise.
- `TSCalc` understands exponential notation; for example, `1.234e-3`.

By double clicking on a variable in the list, the name appears in the command window and can be combined with commands from the calculator. Apart from basic arithmetic operations, `TSCalc` provides a range of other functions like `sin`, `cos`, `tan`, `min`, `max`, `lagn`, `log`, `stdc`, `meanc`, `rndn`, `ones`, `trend`. For a complete list of possible operators and functions, consult the help system. In case there are syntax errors, a descriptive message is printed to the output window. If the selection in the time series selector is changed, the workspace of the calculator is overwritten. Variables that have not been added to the workspace are lost.

8.5 Managing Variables in `JMulTi`

`JMulTi` uses a sophisticated system to share variables between different components. This makes the internal design of the program much clearer, but it also has the benefit for the user that there is a great deal of transparency over almost all variables that are used by the system. It is possible to check input parameters as well as results being read back after a procedure has finished.

The tool to access the variable space is the `Symbol Control Frame`. A tree structure allows browsing through the variables by their names. Each variable can be displayed in a table, and a description is usually available on what is stored in it and what type and dimension it has. The variables in the tree always belong to a certain scope that is defined by the location of the variable in the component hierarchy. For example, a variable stored for the VEC model is not accessible from the VAR model. This structure is reflected in the tree.

As a special feature it is possible to save every selected variable to file to reuse it in another program. Various file formats are supported. This feature can also be used for exporting output and using it in another program for creating graphics for a publication. The main idea of this facility is to provide as much flexibility as possible and to offer a way of overcoming the limitations of `JMulTi` by using its export functions.

8.6 Notes for Econometric Software Developers

8.6.1 General Remark

The following short overview should serve as a motivation for interested researchers to use the software described in this chapter not only as a ready-made tool but also as a platform to quickly create new applications for various purposes in econometrics. For example, with the proper tools it would be possible to create a user interface for a new test or estimation routine in a few hours without a deep knowledge in object-oriented programming techniques. On the other hand, the system is flexible enough to serve as a platform for more demanding applications as well. The technical details of how to implement this capability are not described here, but the general idea of the underlying framework is presented. For more information the interested reader is referred to the homepage[3] of the project.

8.6.2 The JStatCom Framework

As mentioned in the introduction, JMulTi is based on a software framework called JStatCom. A software framework is a set of reusable classes that make up a reusable design for a class of software [see Johnson & Foote (1988), Deutsch (1989)]. This means that it already provides a structure as well as key functionality for applications in a certain problem domain. The designer of an application can reuse not only classes but the whole design of the framework and concentrate on specific aspects of his or her implementation [Gamma, Helm, Johnson & Vlissides (1995)]. The problem domain for JStatCom is time series analysis, but it could well be extended to statistical modeling in general or simulation, such as for macroeconomic models.

JStatCom is a set of JAVA classes together with additional native libraries necessary to communicate to certain computational engines. It is organized into several packages as shown in Figure 8.1. The dashed lines indicate dependencies between packages, and it can easily be seen that the model package contains the core classes on which all other packages depend. For time series analysis the ts package defines all relevant functionality. The other packages are largely independent of each other. All classes and components within that framework can be accessed by a well-documented application programming interface (API). A special feature of JStatCom is that it makes heavy use of the JavaBeans component architecture as defined by Sun Microsystems (1.01-1997). Components can be configured and plugged together in a standardized way, which can speed up development significantly.

8.6.3 Component Structure

To understand the underlying general design better, Figure 8.2 summarizes the relationships between JStatCom, JMulTi, the computational engine, and statistical procedures written for a certain engine. All components together make a runnable application. The main building blocks of the software are clearly separated from each other and can be developed independently.

The application developer needs to implement the statistical procedures for a specific engine, and he or she must create the user interface for the application. The first task should be familiar to researchers because it means just using a programming language to implement the desired algorithm.

[3] The homepage is http://www.jstatcom.com.

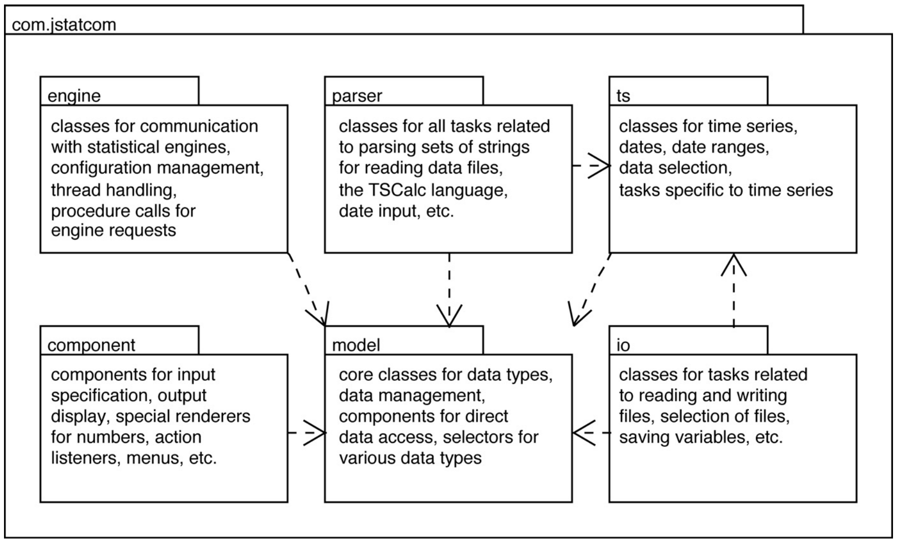
com.jstatcom
engine
classes for communication with statistical engines, configuration management, thread handling, procedure calls for engine requests
parser
classes for all tasks related to parsing sets of strings for reading data files, the TSCalc language, date input, etc.
ts
classes for time series, dates, date ranges, data selection, tasks specific to time series
component
components for input specification, output display, special renderers for numbers, action listeners, menus, etc.
model
core classes for data types, data management, components for direct data access, selectors for various data types
io
classes for tasks related to reading and writing files, selection of files, saving variables, etc.

Figure 8.1. JStatCom JAVA Package Structure.

The task of creating the JAVA application on the other hand is greatly simplified by the framework design of JStatCom. Many sophisticated classes and components are available and can easily be reused. The general functionality described earlier in this chapter is immediately available if the respective components are used. As a developer, one only needs to think about the special needs of a certain procedure, which basically means input and output specification. Although some programming in JAVA is always needed, most steps can be done in a standardized way.

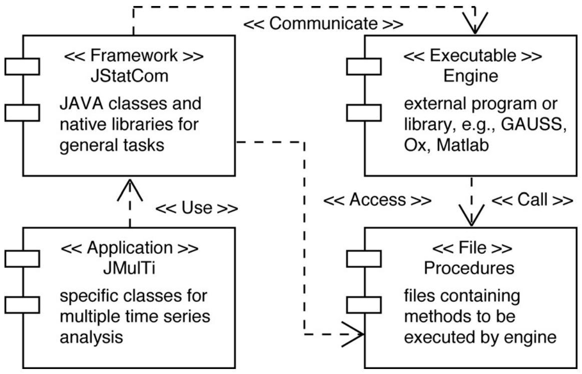
<< Communicate >>
<< Framework >>
JStatCom
JAVA classes and native libraries for general tasks
<< Executable >>
Engine
external program or library, e.g., GAUSS, Ox, Matlab
<< Use >>
<< Access >>
<< Call >>
<< Application >>
JMulTi
specific classes for multiple time series analysis
<< File >>
Procedures
files containing methods to be executed by engine

Figure 8.2. Collaboration of components.

The design splits the complex task of creating an application for time series analysis into smaller units, which can be solved separately. The problems that occur in literally every analysis have been solved in the JAVA framework already. The task of creating the GUI and the task of programming a complex statistical procedure are almost completely separated and can even be done by different people. That way it is possible to reuse already written code efficiently and to enhance its value by integrating it with a graphical user interface.

8.7 Conclusion

JMulTi uses the functionality of JStatCom to implement interesting, new, and otherwise difficult-to-use methods in time series econometrics. It is freely available, using free code provided by many authors.

On the other hand, the development of JStatCom would not at all have been possible without the experiences gained from creating, using, and extending JMulTi. The framework is an extract of all general solutions that have been found to be useful and are not related to a specific model but occur frequently when implementing new methods. Both pieces of software will, it is hoped, continue to be improved, maintained, and extended with new features in the future. It is likely that the software will change much faster than this book; therefore, users are advised to check the respective websites for the latest developments.