RATS

VERSION 11

INTRODUCTION

RATS

VERSION 11

INTRODUCTION

Estima
2717 Harrison St
Evanston, IL 60201

Sales, Orders	+1 847–864–8772	Web:	www.estima.com
Technical Support	+1 847–864–1910	Sales:	sales@estima.com
Fax:	+1 847–447–0174	Technical Support:	support@estima.com

Estima
2717 Harrison St
Evanston, IL 60201

Published in the United States of America

Preface

Welcome to Version 11 of RATS. With this version, we have completed our transition to providing all the documentation in the form of HTML files, which are included both with the distribution of the software, and also are kept up-to-date on our web site (https://estima.com/webhelp). We are still providing an important subset of the information in a three-book manual set (this *Introduction*, the *User's Guide* and the *Reference Manual*) in PDF format for those who prefer that. Note, however, that the HTML files have almost triple the information that's provided in the PDF's as they include detailed descriptions of the many examples and procedures.

Because the HTML help is much more extensive, we removed some relatively obsolete instructions and statistical methods that are no longer topical from the main documentation to try to keep the size of the main manuals within reason. These are still documented in the on-line help. A number of secondary "documents" that we used to keep in PDF form are now much more usefully provided in the help. Among these are the list of procedures, list of examples, list of paper replications, list of functions, list of reserved variables, and list of error messages.

In 2002, we began a project of implementing (all) the worked examples from a wide range of textbooks. We now number more than thirty texts in the collection, including everything from introductory econometrics and time series books to graduate level books covering almost the full range of modern econometrics. In addition, we have an ever-growing number of replication files for important papers. Our experience with this (over 1000 running examples) has led to great improvements in the program. We're grateful to Kit Baum, Tim Bollerslev, Chris Brooks, Kai Carstensen, Richard Davis, Steve De Lurgio, Frank Diebold, Jordi Gali, John Geweke, Bill Greene, Jim Hamilton, Fumio Hayashi, Andrew Mountford, Timo Teräsvirta, Ruey Tsay, Harald Uhlig, Marno Verbeek, Mark Watson and Jeff Wooldridge for their help in providing data and programs, and checking results. A special tip of the hat goes to Bruce Hansen and Mark Watson, who not only post their code and data, but are very quick to act on questions about them.

Thanks to all those in the RATS community who have contributed ideas, suggestions and procedures for others to use. Special thanks go to Walter Enders for co-writing the popular e-book, *The RATS Programming Language*. The second edition of that is included in the RATS distribution. Also to Paco Goerlich and Norman Morin for the numerous procedures they've written. Jonathan Dennis from the University of Copenhagen (the principal programmer of the CATS program) had a major role in designing and refining the **REPORT** and **DBOX** instructions. Chris Sims (now Nobel Laureate), as always, had a special role in influencing the direction of our coverage of time series.

My thanks to my wife Robin for her infinite patience with me and my laptop and mobile hot-spot as I worked on the program and documentation updates and to my daughters Jessica and Sarah.

Thomas A. Doan
May 2025

Table of Contents

1. Getting Started—A Tutorial

The object of this chapter is to get you up and running as quickly as possible. The first thing to do is to get your program installed. If you haven't done that, and need some help with it, see page Int–158 in this book.

Once you're ready to go, you should start working through this book. It introduces you to how RATS gets information from you and supplies results to you. We provide several examples designed to give you a taste of what you can do with the program. To allow you to find your own pace, we have kept the examples fairly short; just long enough for each to introduce a few key concepts. The main discussion in each is intended to be followed in sequence—what's discussed in one is assumed in the next. However, each of the examples is followed by one or more "Learn More" subsections. These you can skip on first reading, particularly if you're never used a statistical package like RATS before.

These "Learn More" segments *do* include information that you will probably need eventually, so you probably will want to come back to them eventually. If you are very proficient at working with statistical languages, then you might want to go through these the first time, since many of the ideas are shared with other such programs.

1.1 Using the RATS Manuals

There are three main manuals: this, the *User's Guide* and the *Reference Manual*. RATS is a program with a very broad range of uses. The idea behind the *Introduction to RATS* is to demonstrate the most important concepts of the program: the different window types and what you can do with them, the wizards, the data handling and graphics. If you are new to RATS, we strongly recommend that you take the time to work through the examples in this chapter. Many of the examples are followed by "Learn More" sections—these are less critical on first read for someone new to the software, but you should eventually come back to them to become more proficient.

The *User's Guide* describes how to use RATS for various types of analysis (with extensive examples). In many cases, you can skip directly to the chapters in that which are of greatest interest. The *Reference Manual* provides a detailed description of each instruction and is designed to be used as the name describes: as a reference in case you need to know more about an instruction. However, all the information (and much more) in the *Reference Manual* is now included in the on-line help, both on our web site at https://estima.com/ratshelp and also installed with your software (*Help* menu). There is a combined index covering the *Introduction*, *User's Guide*, and *Reference Manual* at the end of the *User's Guide*.

Conventions Used in the Manuals

We use the following font and style conventions in the RATS manuals:

Examples	Examples that can executed as written are presented in bold Courier font. For example: **linreg(define=req) rate 1960:1 1995:8 resids_rate** **# constant ip grm2 grppi{1}**
Instructions	Instructions are the fundamental building blocks of the RATS language—they instruct RATS to perform an action. In the example above, the instruction **LINREG** performs a linear regression. In body text, instruction names are presented in uppercase bold Courier font. For example **BOXJENK**, and **DATA**.
Parameters	Parameters are used to provide additional information to an instruction, such as the series names RATE and RESIDS_RATE in the example above. When describing instructions, the names of the available parameters are presented in lowercase italicized Courier. For example, *start*, *end*, *series*.
Options	Options are used to modify how instructions behave. In the sample above, DEFINE defines an equation. In body text, options are presented in uppercase Courier font (such as DATES and PRINT). Bold is used in option lists in the *Reference Manual*.
Variable Names	Variables (series, matrices, scalars, and so on) are generally presented in uppercase Courier, such as RATE and IP. In some cases, mixed-case names are used for readability.

1.2 Example One: The RATS Editor

To introduce you to RATS, we will take you step-by-step through several sample RATS sessions. We'll describe the key menu operations and RATS instructions, and give you a feel for using the program.

If you have not already installed RATS on your computer, see page Int–158 for installation instructions.

The RATS Editor

First, start the program in interactive mode. For Windows, open the WinRATS folder on the *Start* menu or desktop and click on the WinRATS icon. (The folder and icon will generally be a longer name which includes the current version number). For MacRATS, click on the MacRATS program icon. For UNIX, click on the "RATSx" file, or type `ratsx` at the command prompt.

This loads the RATS interactive interface, which we call the RATS Editor. You will see the RATS menu bar, the toolbar, and an empty worksheet window called `NONAME00.TXT`. (If you don't see this, select *New...Editor/Text Window* from the *File* menu or use the `<Ctrl>+N` keystroke).

This environment allows you to type in and execute commands; use menu-driven wizards to perform tasks; save or print program files and output; export information to various types of files; display, save, and print graphs; and more.

If you look at the *File* and *Edit* menus, you'll see familiar operations, like *Save* and *Print* on *File*, and *Cut*, *Copy* and *Paste* on *Edit*; and they all work as you would expect (short-cut keys as well). What you *won't* find are operations for Bold, Italics, Centering, etc. The RATS editor isn't intended to be used to produce a final document—its job is to help you process numerical information and get it into your final document as accurately as possible.

While you do have some choice of font, it's from the very limited number of "monospaced" fonts (Courier is by far the most common of these). In a monospaced or typewriter font, all characters are the same width, making it easy to line up different types of output. RATS also produces output which is in specially formatted tables, but those are separate from the main editor window.

Input and Output Windows

RATS allows you to have multiple windows open at one time, but you can only execute instructions from the window that is designated as the Input Window. This window will have `{i}` appended to the window title, both on the window's title bar, and on the window list in the *Window* menu.

Similarly, any output generated goes to the window designated as the Output Window, which will have `{o}` appended the name. This can be a separate window, or it can be the same as the Input Window. When you first start the program, the `NONAME00.TXT` window should be set as both input and output, shown by `{io}` in the window title bar.

Executing Instructions

In this tutorial, much of the work is done using "point-and-click" operations with menus and dialog boxes. However, we will also show you how to type in and execute instructions directly, which is often the fastest way to accomplish many tasks once you have some familiarity with the language.

In general, you can execute an instruction by just typing it into the Input Window and hitting the <Enter> key. You can also execute a *block* of instructions by selecting (highlighting) the lines you want to execute using the mouse or the keyboard, and then hitting <Enter> or clicking on the "Run" icon  . RATS will execute all of the instructions, in order, from top to bottom.

What Should You Type In?

We put large pointers (⇨) in the margin next to all the instructions (or groups of instructions) that we want you to enter and execute. We will also sometimes show you other sample uses of these instructions. Some of these will work with the sample data sets, but others will not. For now, we recommend that you only type in the instructions marked with a pointer.

The instructions we discuss in this section are provided on a file called `ExampleOne.rpf` (RPF stands for RATS Program File). If you encounter any difficulties, you can open that file (using *File–Open*) to see exactly how the instructions should look.

The DISPLAY Instruction

To get started, type the line below into the blank input window and then hit the <Enter> key (which tells RATS to execute the line you just typed):

⇨
```
display "Hello, World"
```

RATS should display the text "Hello, World" on the next line (or to your output window if using separate input and output windows). As you can see, **DISPLAY** does just what its name implies—it displays information.

You can also use it to display the results of computations. For example, a popular econometrics textbook includes the following expression to compute "by hand" the least squares estimator for the intercept in a regression (Hill et al, 2008, page 22):

(1) $b_1 = \bar{y} - b_2\bar{x} = 283.5737 - (10.2096)(19.6408) = 83.4160$

You can do this computation easily using **DISPLAY**. Try typing in the line below (with no spaces inside the expression), and hitting <Enter> to execute it:

⇨
```
display 283.5737-10.2096*19.6048
```

RATS will do the computation and display the result: `83.41653`

Note that the * symbol is the multiplication operator. The addition and subtraction operators are the usual + and – . Other operators include / for division and ^ for

exponentiation. We describe the full set of operators in Section 1.4.9. Multiplication is *never* implied, the way it is in the algebraic expression above, so you *must* use the * to multiply two terms.

Rounding and Using Full Precision

The textbook actually reports a slightly different result (83.4160). That's because the numbers presented in the book for equation (1) (and used as input in our computation) were themselves rounded versions of the numbers actually used to produce the result shown in the book. RATS (and most other statistical software) does its calculations in "double precision", which gives roughly fifteen significant digits. Now no practical data set records numbers to anything like that level of accuracy, so if the data going in are only good for four digits, it's pointless to report results at fifteen digits. However, whenever possible you should keep all the *intermediate* calculations in full precision: in short, don't round intermediate values. The software is already rounding at fifteen, which sometimes isn't even enough for very poorly-behaved data.

DISPLAY picks the format itself, and usually errs on the side of too many digits. If you want this number rounded to four decimal places, you can use "picture codes", which provide a template for numerical output. Insert the picture code before the number or expression. For example, edit your original instruction by inserting ##.#### (spaces on either side) before the expression:

```
display ##.#### 283.5737-10.2096*19.6048
```

Hit <Enter> to re-execute this line. The cursor can be anywhere in the line—it doesn't have to be at the end of the line (you're executing the line, not inserting a line break).

As we said before, the job of the RATS editor is to help you get information into your final document as accurately as possible. RATS comes with over 1000 examples out of major textbooks. The most common type of error in those textbooks comes from taking a number computed by the software and manually re-typing it (incorrectly) into the manuscript. If you want a value, let RATS do the rounding, and use copy and paste to get it into your document. Select the text you need, and do a copy operation. The text copy is so important, there are four ways to do it: the keyboard shortcut (<Ctrl>+C), the copy toolbar icon  , the standard menu operation *Edit—Copy* and "Copy" on the contextual (right click) menu.

DISPLAY can also show more than one calculation, and more than one field (which is why we wanted you to be careful about spaces). For instance, you can edit your line to add a description and hit <Enter> to execute:

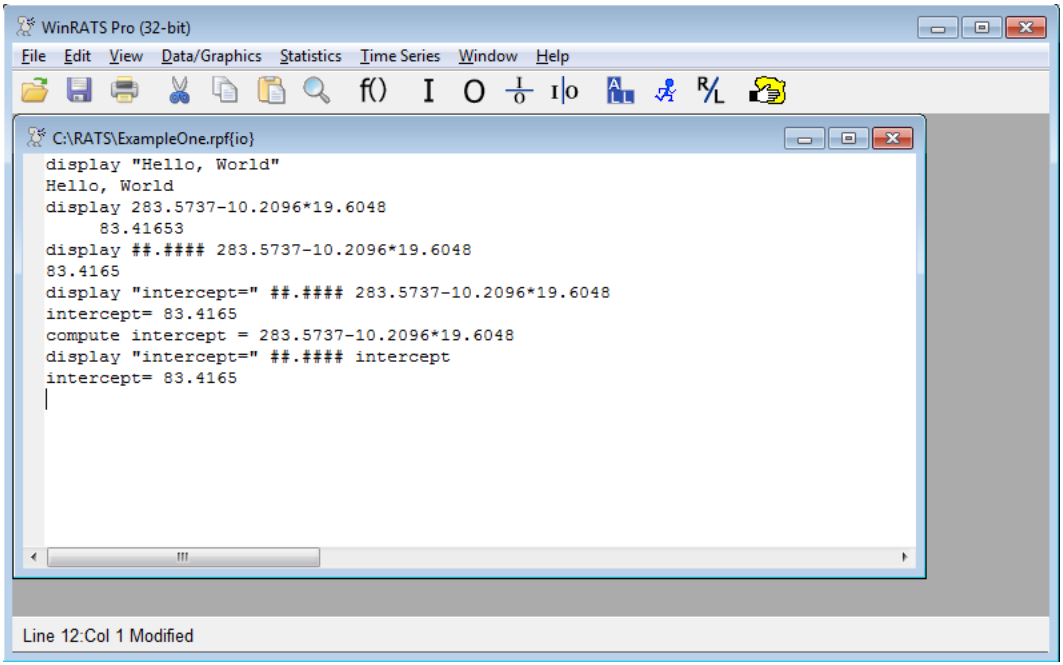
```
display "intercept=" ##.#### 283.5737-10.2096*19.6048
```

The COMPUTE Instruction

COMPUTE is another important instruction. If you need to do an algebraic calculation, but don't need to see the result, you will probably do that using **COMPUTE**. An alternative to doing our computation with **DISPLAY** would be (execute both lines):

```
compute intercept = 283.5737-10.2096*19.6048
display "intercept=" ##.#### intercept
```

This does the calculation and saves the result into a new variable called INTERCEPT, and then displays that. Your screen should now look something like this:



It's important to note a major difference between RATS and "math packages" like Gauss™ and MATLAB™. With either of those, the first "display" would be done with

```
283.5737-10.2096*19.6048;
```

("display" is implicit since there is no other place for result), and the **COMPUTE** with

```
intercept=283.5737-10.2096*19.6048;
```

With math packages, the basic unit of input is an expression. With RATS, it's an instruction. **DISPLAY** and **COMPUTE** are two examples—there are roughly 200 others, many of which do *very* sophisticated calculations requiring (internally) thousands of separate sub-calculations. Many of those have point-and-click "wizards" on the *Data/Graphics*, *Statistics* and *Time Series* menus to help you apply them to your data.

1.2.1 Learn More: Input and Output Windows

The Run/Stop Buttons

While RATS is executing instructions, the "Run" icon changes to the "Stop" icon. "Run" returns as soon as the task is complete and RATS is ready to accept more instructions. In many cases, you won't even notice this because it finishes so quickly. However, if you are doing something which does take a long time, and you decide that you don't really want it to continue, you can click on "Stop" to interrupt the execution.

Editing Without Executing (Ready vs Local Mode)

Hitting the <Enter> key in the Input Window normally executes the current line. If you need to insert a line break *without* executing the line (perhaps you want to input several lines before executing them), do the following:

- On Windows and UNIX/Linux systems, you can edit lines without executing them by putting RATS into "local" mode, either by clicking on the R/L (Ready/Local) button or by hitting <Ctrl>+L. Clicking on the Ready/Local button again (or hitting <Ctrl>+L again) puts RATS back into "ready" mode. "Ready" means that RATS is ready to accept instructions for execution.
- On a Macintosh, if you have a keyboard with separate <Return> and <Enter> keys, you can use <Return> to insert a line break without executing a command. If not, use the same procedure as for Windows.

Configuring Input and Output Windows

You can switch the input or output functions to a particular editor window using the menu items *Window–Use for Input* and *Window–Use for Output* or the corresponding I and O toolbar icons. We tend to use the editor windows in one of two ways:

1. For quick work, we simply start up RATS and work with the NONAME window the way we just did in the example. If you aren't really interested in developing a set of RATS instructions to execute later and just want some quick answers, this is the simplest setup.
2. To develop a new (or existing) program, we designate one window as the Input Window, and a second window as the Output Window. With input and output in separate windows, it is much easier to keep a copy of the instructions that we decide we like. The I/O and I|O toolbar icons provide easy ways to get this split-window setup—they automatically open a second window, designate it as the output window, and tile the two windows horizontally or vertically, respectively.

If you decide that you really like the second setup, you can set RATS to default to that on the Editor tab of the *Preferences*—set the "Open New Output Window at Start" box.

1.3 Example Two: Working With Data

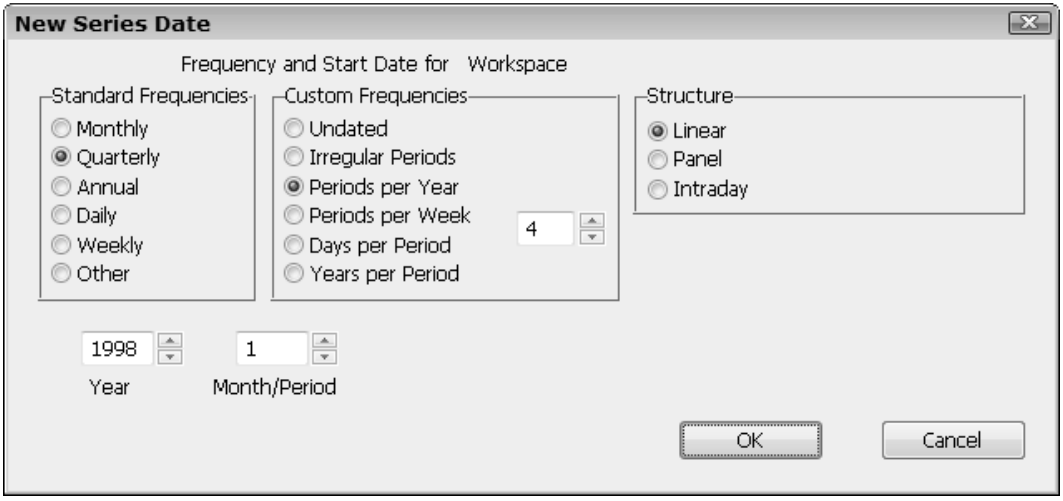
While **COMPUTE** and **DISPLAY** are important, if you're getting started with RATS, you probably have some time series data that you want to use. There are three main types of data collection: time series, cross section/survey, and panel/longitudinal. While RATS can handle all three, its specialty is time series data, and the specialized techniques used in working with it. In this example, we introduce some basic tools for working with time series data. The **CALENDAR** instruction discussed here is provided on the file `ExampleTwo.rpf` (everything else in this section is done using menu operations).

In time series data, the basic unit of data is a *time series*, which is a time sequence of data that (usually) has a specific starting place in time and a specific reporting frequency. What makes time series data very different from other forms is that the data you use may come from many sources—in the U.S., the Census Bureau, the Federal Reserve and the Bureau of Labor Statistics (among many others) are each the original sources of key data series. Time series can start at different times; in some cases, they can end rather abruptly (such as exchange rates involving Euro zone countries).

The Calendar Wizard

To work with time series data, we first want to tell RATS the frequency and starting date for our data set. One way to do that is to use the Calendar Wizard. Select the *Calendar* operation from the *Data/Graphics* menu. RATS will display the Calendar Wizard dialog box shown below, which allows you to set frequency and date information. The most common frequencies are in the "Standard Frequencies" group.

In this case, we want quarterly data, beginning in the first quarter of 1998, so click on the "Quarterly" button and change the "Year" field to read 1998, like this:



Now click on "OK". RATS will generate the corresponding **CALENDAR** instruction:

```
calendar(q) 1998:1
```

in the input window and execute it automatically. The Q in parentheses stands for "quarterly". This is called an *option*, and this option tells RATS that we will be working with quarterly data. When you use options for an instruction, the left parentheses must appear *immediately* after the instruction name, with no spaces in between.

The date 1998:1 tells RATS that our data begin in the first quarter of 1998. This value is called a *parameter*. The parameters generally are information that is either necessary or very commonly used, while the options are just that—something that you might or might not use.

Note that you generally do *not* need to use the Calendar Wizard if you intend to use one of the Data Wizards (page Int–18) to read data in from a file, since they will set the date scheme themselves.

Creating a Data Series

If at all possible, you do *not* want to type in data. Most of your work will be done using data read in from files or database connections, but for now, we'll have you type in some numbers to introduce you to tools for creating, viewing, and editing data.

First, open the *Data/Graphics—Create Series (Data Editor)* menu operation. RATS will display an empty Series Edit Window. As this is a new series, the first cell shows "NA", for "Not Available". We use the NA abbreviation both in RATS output and in the manuals to signify a missing value.

Here are the numbers to enter (data for 1998 in row one, for 1999 in row two, etc.):

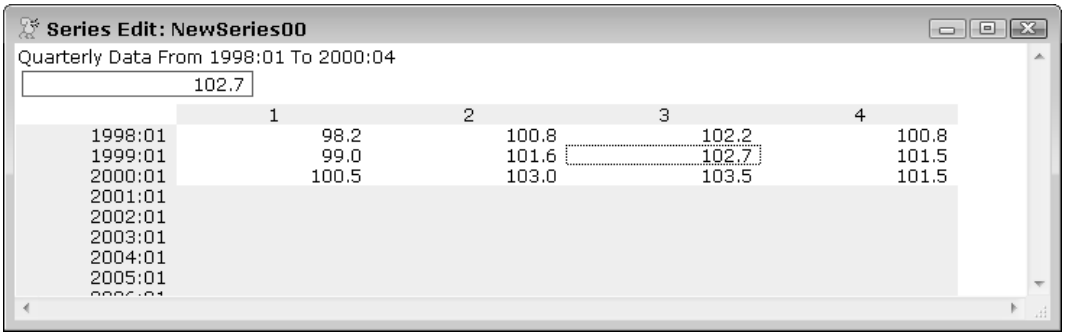
```
 98.2 100.8 102.2 100.8
 99.0 101.6 102.7 101.5
100.5 103.0 103.5 101.5
```

Start by typing the first number (98.2) into the cell at the top. Then, hit the right arrow key. This saves the value you typed in and moves to the next cell. Type in the second number (100.8), hit the right arrow key again, and so on. Note that the right arrow still works when you get to the end of a "line". Because this is a time series, the lines are just to help organize your view of the data—what we really have is a continuous sequence of values. When you get to the final value, just hit <Enter> (rather than the right arrow) to save the value.

If you only want to type in a few of the values, that's fine—everything we discuss below will still work, although the results will look different. If you are reading this as a PDF, you can also copy and paste the data lines above into the editor.

If you need to edit a value, click on (or navigate to) the cell you want to change, edit the value, and hit <Enter>.

Once you are done, the window should look like this:



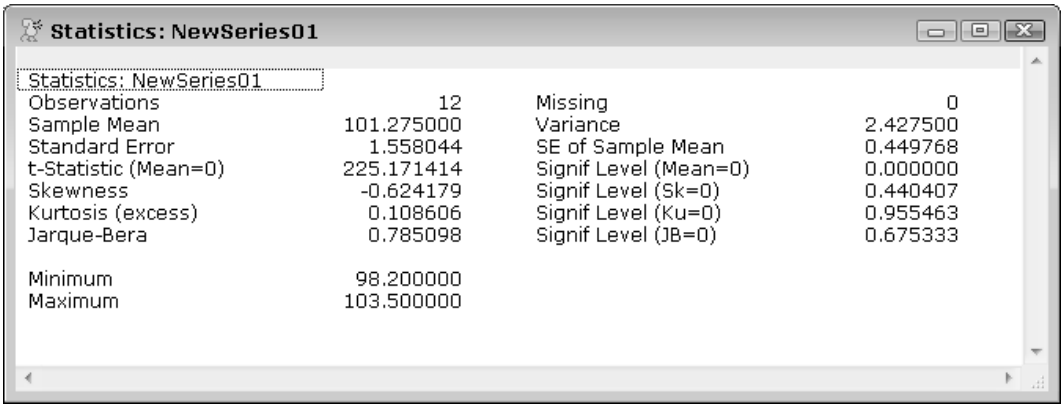
The View Menu

The operations on the *View* menu provide ways to quickly examine data series and other variables. These operations do not generate RATS instructions the way the wizards do. However, they are very handy for verifying your data and doing some "quick glance" analysis before beginning work in earnest.

The second most common error that we've found in replicating examples (and the one that is probably most significant in practice) is where a data set is simply wrong—for instance, the data are for a different set of years than the author thought. And these are for papers which have been circulated, presented in seminars, peer-reviewed and published. Do yourself a favor: make sure your data are what you think they are before doing any major work with them. The *View* menu can help with this.

Try the *View—Time Series Graph* operation (or click on toolbar item). You'll see a simple graph of the data. This is not intended to be a "production" graph; instead, it's a quick line graph of the data that can help you spot any obvious errors like missing a decimal point. If you see that you have a value that's clearly wrong, you should be able to go back to the series editor window and use the Max or Min toolbar icons (max and min), which move to the maximum and minimum values in the series, respectively, to correct the problem.

While there are several other "quick-view" operations (the ones at the bottom of the *View* menu), the other one that is very handy is *View—Statistics*. If you choose this (or the $\overline{X}$ icon), you will see a new type of window called a report that looks like this:



Report Windows

A *Report Window* is like a read-only spreadsheet. The descriptive text and values are arranged in columns, with longer strings covering ("spanning") several columns. Output from RATS will either be in the form of Report Windows or text in the Output Window, and sometimes both. You can copy information out of a Report Window and paste it into a spreadsheet or word processing program. See page Int–15 for more.

Test Results

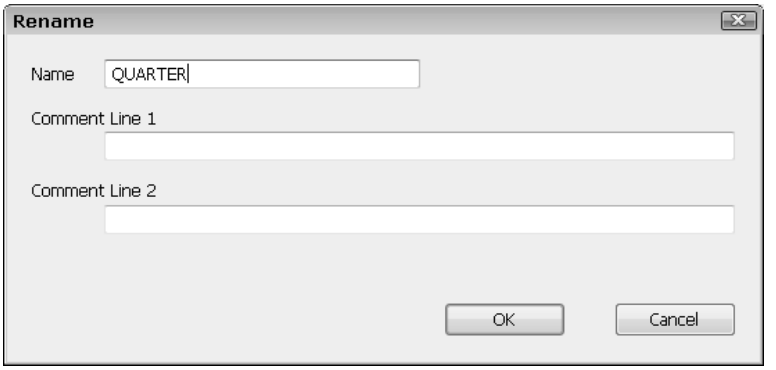
The report for *Statistics* includes four different "tests": the *t*-statistic, skewness, kurtosis, and Jarque-Bera. For each of these, the left column gives the test statistic and the right gives the *marginal significance level* (also known as the *p*-value). This is the probability that a random variable with the appropriate distribution will *exceed* the computed value (in absolute value for Normal and *t*). Wherever possible, RATS will compute the significance level for tests, which allows you to avoid using "lookup tables" for critical values. You reject the null if the marginal significance level is *smaller* than the significance level you want. For instance, a marginal significance level of .03 would lead you to reject the null hypothesis if you are testing at the .05 level (.03<.05), but not if you are testing at the .01 level. Here, the only of the four that we would "reject" under standard procedures is mean=0, which has a test statistic so far out in the tails that the *p*-value shows as just being 0. (If you paste it into a spreadsheet, you'll find that it's actually roughly 10^{-21}.)

This would be a good time to warn you about over-interpreting test results. RATS will often compute tests and summary statistics which *might* be useful. It's up to you to decide whether they actually make any sense in particular situation. The three test statistics here that "pass" are all tests for the Normal distribution (the "excess" in excess kurtosis is the difference between the sample value and what would be expected if the data were from a Normal population). Should we therefore conclude that the data are from a Normal population? Absolutely not. Those are all test statistics computed under the assumption that the observations are independent. This is our raw time series data—we certainly don't expect it to be independent, and from the

graph it's clear that it has a bit of a trend and a rather pronounced seasonal. If we were looking at the statistics on the residuals from an estimated model, then those last three tests might be interesting; here, they aren't. Some other software packages actually do many more of these "automatic" tests than RATS, which has led people misusing them to conclude that a model that made no sense at all was fine because it "passed" all the "tests".

Saving the Data

You may have noticed that we haven't yet talked about saving the data. If you do *File—Save*, you will be prompted for a series name (and comments), or if you do *File—Close*, you will first be asked if you want to save the changes; if you answer "yes", you will be asked the same question about the name and comments:



For our purposes, enter `QUARTER` as the name as shown above. Leave the comments blank.

If you do the menu operation *View—Series Window*, you will now see a Series Window listing all the series in memory. We'll talk more about this when we have more data. For now, this should just have one line for the series `QUARTER`. At this point, we have saved the data out of the Series Edit Window into a RATS series. If you need a permanent copy, you need to take one more step: exporting the data to a file, which can be done with *File—Export* or *File—Save As*.

Series Names

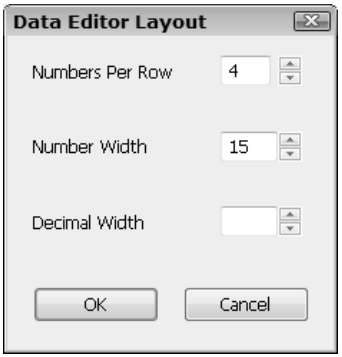
Series (and other variable) names in RATS can be from one to sixteen characters long. They must begin with a letter (or `%`, though you should generally avoid those, as they're used for names reserved by RATS), and can consist of letters, numbers, and the `_` , `$` and `%` symbols. Variable names aren't case-sensitive.

Generally, you'll want to choose names which are just long enough so you will know what each represents. In most cases, your data files will include names for the series stored on the file, and RATS will use those names when it reads in the data. Thus, the series names on the file should conform to the restrictions described above.

1.3.1 Learn More: Series Edit Windows

Change Layout

When using a Series Edit Window to edit a series, you can change the way the data appear on the screen using the menu operation *View—Change Layout* or the toolbar icon . This brings up the dialog:



The main value of changing the "Numbers Per Row" is to match the view of the data to your original source if you need to double-check numbers that you have had to type in. The most common change in the layout is to adjust the decimal digits. By default, the data editor uses the shortest representation that can display all the data. That works well with original source data with a limited number of digits. However, with data that have been through some transformations (such as logs or averages), you might get ten (or even more) decimals. *View—Change Layout* lets you adjust that. Click on "OK" to close the Change Layout dialog box and apply your changes.

Copy and Paste

You can copy data out of a Series Edit Window by selecting a (consecutive) range of values and using the keystrokes <Ctrl>+C, the menu operation *Edit—Copy*, the toolbar icon or the "Copy" contextual menu item. When you paste into a target application, the values will have the same row and column arrangement as you have in the Series Edit Window in RATS, that is, if you have 40 rows of 4 numbers, you will also get a 40×4 table in the target. If you paste as text, the data will be pasted as it appears in the window as well, with the number of digits shown. You can use *Change Layout* to adjust either the arrangement of the data or the number of digits.

You can paste into a Series Edit Window by positioning the selection to the point where you want to insert values, then doing <Ctrl>+V, the menu operation *Edit—Paste*, or the toolbar icon . The paste operation will take the values in order, even if the layout is different from what you have in the Series Edit Window. For instance, if you select and copy a column of data from a web site and paste it into the time series editor, the cells will be filled, in order, starting from the active cell, going first across and then down to the start of the next row, across that, etc. Note that you need the source data to be values from a single series in *ascending* time sequence.

Toolbar Icons

In addition to standard toolbar items like Select All, Copy, Paste, Print, the following specialized toolbar icons are available when using a Series Edit Window

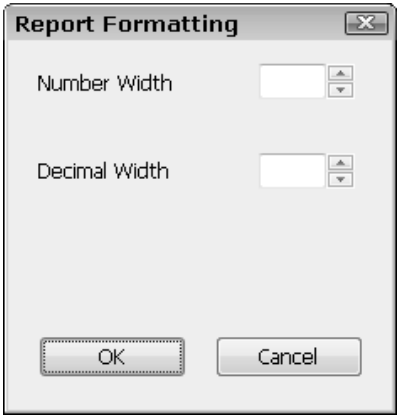
 (Insert)	Inserts a new cell at the current cursor position (cell value is set to NA).
 (Remove)	Removes the current cell and shifts the remaining data one position to the left to fill its place.
 (NA)	Sets the current cell to the missing value code (NA, or Not Available).
 (Max. value)	moves the cursor to the cell containing the largest (maximum) value in the series.
 (Min. value)	moves the cursor to the cell containing the smallest (minimum) value in the series.
 (Find)	searches for an (exact) value.
 (Statistics)	displays a Report Window with basic statistics (mean, variance, skewness, etc.) for the series.
 (Graph)	displays a time series graph for the series.
 (Histogram)	displays a histogram plot for the series.
 (Graph Xform)	displays a time series graph with several power transformations of the series.

1.3.2 Learn More: Report Windows

Change Layout

A *Report Window* is like a read-only spreadsheet. You can adjust the appearance, but can't change the values themselves. Some (in this case most) of the values shown in a report are available at a higher precision than you see in the table. On the other hand, they will often show far more decimal places that you would likely use in any document that you would be preparing.

To change the appearance of values, use the *View—Change Layout* menu operation (or the toolbar or the *Reformat* operation on the contextual menu). This will bring up the following dialog box:



If you set the "Decimal Width" box, the values will be formatted to that number of digits right of the decimal, with however many are needed to the left of it. If you set the "Number Width" box, a common format will be chosen that displays all the values within that number of characters. (If you set both, the decimal width choice determines the representation).

Copying To Other Applications

If you select information out of this and copy it (which you can do many ways: the keystroke <Ctrl>+C, *Edit—Copy* menu, Copy toolbar icon , and *Copy* contextual menu), and then paste into a word processor, you will get the information that is shown on the screen, with the number of digits shown. If your target application is a spreadsheet, you may need to choose *Paste Special* in order to control what comes through. (RATS copies report information in many different formats). The DIF, CSV and XML spreadsheet formats will paste numbers at full (up to fifteen digit) precision. On the other hand, if you paste as (Unformatted) Text, you will get only the number of digits shown on the screen in RATS, so if you adjust this using Change Layout, you will get what you see. Note that RATS only copies "content", not column widths and cell formats, so you can reformat the cells in your spreadsheet to show them the way that you want.

Copying to TeX

If you want to paste into a TeX document (as a table), you need to use the *Edit—Copy as TeX* (or  toolbar or *Copy->TeX* contextual menu operation). When you do that, RATS puts into the clipboard a TeX tabular environment for displaying the table, with the numbers as shown on the screen, so do the *Change Layout* first to get the representation you want. See the description of `FORMAT=TEX` for more.

Exporting to a File

Select the cells that you want to save and use *File—Export* (or *File—Save As*, or the  toolbar icon or the "Export" contextual menu). You have the choice of quite a few formats, most of which are also among the formats that RATS "copies" when you do a Copy operation. The format here that is most likely to be useful is TEX, since you can use an `\include` directive in your TeX document. By using that, rather than copy and paste, you can quickly replace the table if you need to re-generated the report.

1.4 Example Three: Transformations and Regressions

In this section, we will work with a sample data set from Pindyck and Rubinfeld(1998) to demonstrate a variety of tasks, including data transformations and multiple regression models.

All of the commands that we will describe here are also provided for you in a RATS program file called `ExampleThree.rpf`.

We recommend that you begin by entering the commands yourself as we discuss them. However, if you encounter difficulties, you can refer to (or execute the instructions on) `ExampleThree.rpf` to see exactly how things should look.

The sample data set is provided on an XLS (Excel) spreadsheet called `ExampleThree.xls`. The following data series are provided on the file:

Rate	Three-month treasury bill rate.
IP	Federal Reserve index of industrial production, seasonally adjusted, index 1987=100.
M1	Money Stock M1, billions of US dollars, seasonally adjusted.
M2	Money Stock M2, billions of US dollars, seasonally adjusted.
PPI	Producer Price Index, all commodities, index 1982=100, not seasonally adjusted.

Our primary goal will be to fit the following regression models to these data:

(2) $$Rate_t = \alpha + \beta_1 IP_t + \beta_2 \left(M1_t - M1_{t-3}\right) + \beta_3 PSUM_t + u_t$$

where $$PSUM_t = \frac{\Delta PPI_t}{PPI_t} + \frac{\Delta PPI_{t-1}}{PPI_{t-1}} + \frac{\Delta PPI_{t-2}}{PPI_{t-2}}$$

(3) $$Rate_t = \alpha + \beta_1 IP_t + \beta_2 GRM2_t + \beta_3 GRPPI_{t-1} + u_t$$

where $$GRM2_t = \frac{\left(M2_t - M2_{t-1}\right)}{M2_{t-1}},\ GRPPI_t = 100\frac{\left(PPI_t - PPI_{t-1}\right)}{PPI_{t-1}}$$

Equation (2) is actually from Example 4.2 in the 3rd (1991) edition of Pindyck and Rubinfeld, while equation (3) is from Example 4.2 as it appears in the 4th (1998) edition. Our data were taken from Haver Analytics' USECON database. The Haver series names for these are FTB3, IP, FM1, FM2, and PA, respectively. We've converted them to the names shown above for our example. The series begin in January, 1959, and run through September, 1999. Some of the values are slightly different than the older data used in Pindyck and Rubinfeld.

Here's a portion of the data on the file:

```
Date        IP       M1        M2         PPI        RATE
1959:01     36.000   138.900   286.700    31.700     2.84
1959:02     36.700   139.400   287.700    31.700     2.71
1959:03     37.200   139.700   289.200    31.700     2.85
1959:04     38.000   139.700   290.100    31.800     2.96
1959:05     38.600   140.700   292.200    31.800     2.85
1959:06     38.600   141.200   294.100    31.700     3.25
1959:07     37.700   141.700   295.200    31.700     3.24
1959:08     36.400   141.900   296.400    31.600     3.36
1959:09     36.400   141.000   296.700    31.700     4.00
1959:10     36.100   140.500   296.500    31.600     4.12
1959:11     36.300   140.400   297.100    31.500     4.21
1959:12     38.600   140.000   297.800    31.500     4.57
```

1.4.1 Getting the Data In

As in our previous example, we want to start by defining a date scheme and then reading data into memory. In this case, though, we'll be reading the data from a file, which is much more common. We *could* just type in the necessary instructions, but another option is to use the Data Wizard to do this.

The Data Wizard approach is usually preferable for most file formats. However, for certain file formats, or in cases where you need to pull in data from multiple sources, you may need to type the instructions directly.

A Clean Slate—The Clear Memory Operation

If you still have RATS running after completing the previous example, you will probably want to clear the memory of the settings and data entered earlier. You can do that by opening the *File* menu and selecting *Clear Memory*, or clicking on the "Clear Memory" toolbar icon:  . You can also close the Input Window, then open a new one using the menu operation *File–New-Editor/Text Window*.

Any of these will clear any data series and other variables, as well as any settings defined by instructions like **CALENDAR**, from memory. This allows you to start fresh, as if you had never executed any instructions. Note that *Clear Memory* does *not* delete any text or close any windows.

The Data Wizard

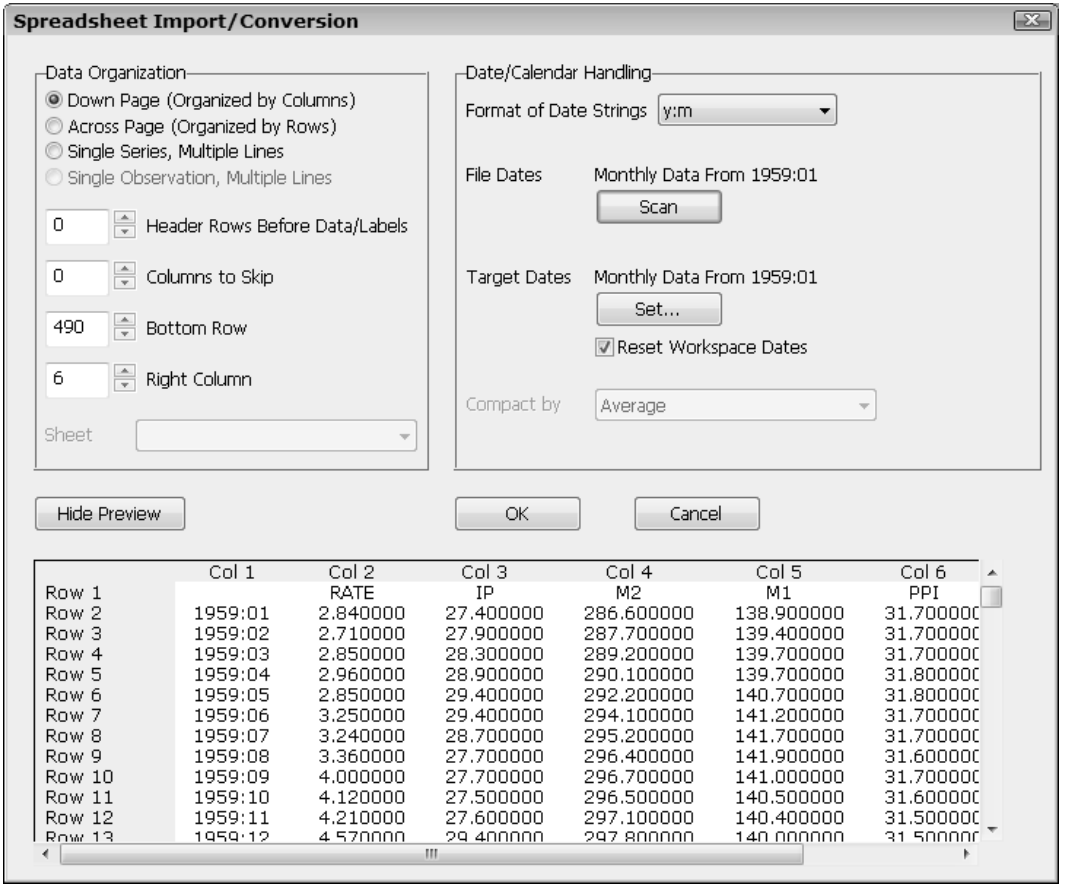
RATS actually offers *two* Data Wizards—one for use with RATS format data files, and one for all other file types. We'll discuss the wizard for RATS format data files in Section 1.7. For now, open the *Data/Graphics* menu and select the operation *Data (Other Formats)*.

We will be working with an Excel "xls" format file called `ExampleThree.xls`, so select "Excel 3.0-2003 Files (*.XLS)" from the drop-down list of file types. The dialog box should now show files with that extension in the current directory.

The default start-up directory for RATS should be the directory containing the main example programs, data files, and procedures that ship with RATS, including `ExampleThree.xls`. So, you should see as one of the files listed in the dialog box. If not, you can use the dialog box to navigate to the appropriate directory.

Select `ExampleThree.xls` and click the "Open" button. RATS will display the Data Wizard dialog box. Click on the "Scan" button, which scans the date information on the file to determine the appropriate **CALENDAR** setting.

You should see something like this:



Data Organization

The "Data Organization" section tells RATS how the data are arranged on the file. Here, the data series run down the page in columns, so we want to use the "Down Page" setting (which is, by far, the most common and should be selected by default).

You only need to use the "Header Rows" field if you need to skip lines (*other* than a row of series names) at the top of file, such as lines of text describing the contents of

the file. Our data file begins with the series labels (which we need) on row 1, so we leave "Header Rows" set to zero.

Similarly, you can use "Columns to Skip" if you want RATS to ignore columns at the left of the file. The "Bottom Row" and "Right Column" fields default to the last row and column numbers found in the file. You can reduce these values to skip rows at the bottom or columns at the right. For this example, leave these set to the default values.

If you are reading a spreadsheet file containing data on multiple sheets, the "Sheet" field will let you pick which sheet you want to read. If you change sheets, the preview table will reload and the "Bottom Row" and "Right Column" fields will reset.

Date/Calendar Handling

The "Format of Date Strings" field shows what RATS thinks is the format of the date information on the file (if any). If it thinks there is more than one possible interpretation, you can use this field to select from the choices it offers. Here, RATS has correctly guessed that the dates are in the "year:month" form (abbreviated "y:m").

The "File Dates" field describes the frequency and starting date of the data as it appears on the file. When you click on "Scan", RATS examines the file to determine the frequency and starting date of the data set. In this case, it should report that the data is monthly, starting in January of 1959.

The "Target Dates" field shows the starting date and frequency that will be used in your RATS session. By default, this will match the "File Dates" setting. You can set or change it by clicking on the "Set" button. This is useful if you only want to read in a subset of the data, or if you want to work with the data at a different frequency.

If you set a different target frequency, RATS will automatically compact or expand the data to that frequency. The "Compact by" box allows you to select the compaction method that will be used when going to a lower frequency. That will only be active if the "File Dates" and "Target Dates" have a different frequency.

Read the Data

For now, accept the default settings and click on "OK". RATS will generate and execute the appropriate **CALENDAR**, **OPEN DATA**, and **DATA** commands. They should look something like this:

```
OPEN DATA "C:\Users\Public\Documents\WinRATS\ExampleThree.xls"
CALENDAR(M) 1959:1
DATA(FORMAT=XLS,ORG=COLUMNS) 1959:01 1999:09 RATE IP M2 M1 PPI
```

Let's take a closer look at each of these.

The OPEN Instruction

```
OPEN DATA "C:\Users\Public\Documents\WinRATS\ExampleThree.xls"
```

The **OPEN DATA** instruction tells RATS the name and location of the data file you want to read. This instruction includes the full path to the file (which may be slightly different on your system). If you are typing in the instruction directly, you can leave out the path if the file is located in the default directory (see page Int–34).

The CALENDAR Instruction

```
CALENDAR(M) 1959:1
```

The **CALENDAR** instruction is similar to the one in our previous example, but here we have the option M for monthly data, rather than Q for quarterly, and the data start in January of 1959.

Some of the Pindyck and Rubinfeld examples apply only to data from November, 1959. We will use "date parameters" to skip earlier observations when necessary in this example, but another alternative would be to set our **CALENDAR** to start in November. We could do that in the Data Wizard by clicking on the "Set" button under "Target Dates" and replacing the "1" in the "Month/Period" cell with "11" (for the 11th month), the resulting **CALENDAR** would be:

```
calendar(m) 1959:11
```

With this setting, RATS would skip the data for January through October, and start reading in data beginning with the November 1959 observation.

If you have cross-sectional data, with no time series periodicity, omit the **CALENDAR** instruction.

Working with Dates

To refer to a date in RATS, use the following formats:

`year:period`	for annual, monthly, and quarterly data (or any other frequency specified in terms of periods per year). In this example, we have set a monthly **CALENDAR**, so "1996:2" translates to the 2nd month (February) of 1996.
`year:month:day`	for weekly, daily, etc.

With annual data, `period` is always one, so any reference to a date in annual data *must* end with `:1`. The `:1` after the year is *very* important, because without it, RATS will assume you are specifying an entry number, not a date.

The DATA Instruction

```
DATA(FORMAT=XLS,ORG=COLUMNS) 1959:01 1999:09 RATE IP M2 M1 PPI
```

This reads the data series RATE, M1, M2, IP, and PPI from the file.

Here, we have two options, separated by commas: FORMAT and ORGANIZATION. You can abbreviate option names to three or more characters, which is what we did with the ORG option. Recall that the list of options must appear immediately after the instruction name, with no space between the instruction and the left parenthesis.

For now, we'll go over these two options quickly. Because the data step is so important, we included a full chapter (Chapter 2) in this book to it; if you need more detail, you can check that out.

The FORMAT Option

FORMAT gives the format of the file you are reading. As noted above, ExampleThree.xls is an Excel spreadsheet file, and the option for that is FORMAT=XLS. RATS supports about 20 formats.

Note: RATS does not try to determine the format of the file based on the file name or extension—the FORMAT option must be set to match the format of the file being read.

The ORGANIZATION Option

The ORG option describes how the data are arranged on the file. The "Down Page" setting on the wizard corresponds to the setting ORG=COLUMNS while the "Across Page" setting corresponds to ORG=ROWS. You can also abbreviate the choices for options like this to three or more characters. For example: ORG=COL.

The Entry Range

The first two parameters (**1959:01** and **1999:09**) specify the starting and ending dates to be read in. You will generally omit explicit date ranges and just let RATS figure out the appropriate range for a given operation. Here, though, the wizard includes the dates so that you know (and have a record of) the exact range being read.

The Series Names

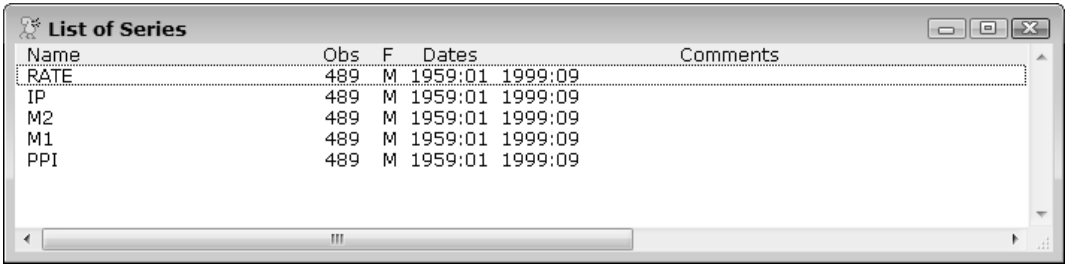
For files that include series names (which RATS requires for most formats), listing the series names on the **DATA** instruction is optional—if you omit the list, RATS reads in all the series on the file. If you do provide a list, RATS reads only those series.

For text files without any series names (which can be handled with FORMAT=FREE), you *must* supply a list of series names if typing in the **DATA** instruction yourself (otherwise RATS would have no way to identify the data). If you use the Data Wizard, RATS will prompt you for names. The series list is also required when reading some database formats, to avoid accidentally reading a huge number of series.

1.4.2 Check the Data

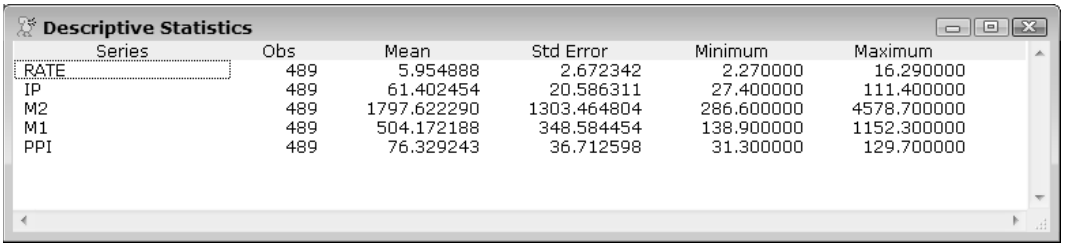
Display the Series Window

First, we need to verify that the data have been read in properly. To begin, select the *View—Series Window* operation. You'll see a window displaying a list of all the series:



This provides a quick check on the number of observations in each series, as well as the frequency and data range. Now, select (highlight) all of the series in the window and then select *View—Statistics*, or click on the "Basics Statistics" toolbar icon: 

You should see the following table of summary statistics for each series. The most important items to check are the number of observations (Obs) and the minimum and maximum values. Be sure they are reasonable given what you know about the data. A particular "red flag" is a minimum of 0 for a series that shouldn't have zero values (GDP for instance), which is a result of a data set using 0 to mean "missing".



You can explore the other *View* menu (and toolbar button) operations introduced on page Int–10 with combinations of one or more series—just highlight the series you want to include before selecting an operation.

Another way to generate the table of statistics is to use the instruction **TABLE**. Type in the following and hit <Enter>:

```
table
```

The results should appear in the output window, and match those shown in the Statistics window above.

STATISTICS and the Univariate Statistics Wizard

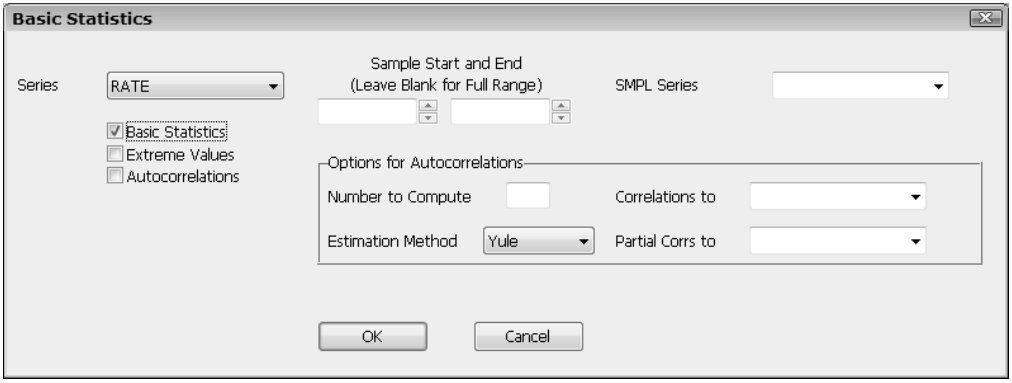
For a more detailed set of sample statistics, you can use **STATISTICS** instruction with a single series:

```
statistics rate
```

Here's resulting output:

```
Statistics on Series RATE
Monthly Data From 1959:01 To 1996:02
Observations                  446
Sample Mean              6.058587      Variance              7.701206
Standard Error           2.775105      of Sample Mean        0.131405
t-Statistic (Mean=0)    46.106212      Signif Level          0.000000
Skewness                 1.186328      Signif Level (Sk=0)   0.000000
Kurtosis (excess)        1.587381      Signif Level (Ku=0)   0.000000
Jarque-Bera            151.440700      Signif Level (JB=0)   0.000000
```

The corresponding wizard is the *Statistics—Univariate Statistics* operation. If you select that operation, RATS will display the following dialog box:



The first step is to select the series you want to use from the "Series" drop-down list—here, we've selected the series RATE.

Next, click on one or more of the "Basic Statistics", "Extreme Values", or "Autocorrelations" check boxes. Here, we've selected "Basic Statistics", which generates a **STATISTICS** command as shown above.

"Extreme Values" generates an **EXTREMUM** instruction, which reports the maximum and minimum values of the series. "Autocorrelations" generates a **CORRELATE** instruction, which computes autocorrelations (and partial autocorrelations if you provide a series for the "Partial Corrs" field). You can check any combination of these three boxes. The other fields allow you to select the range used for the computations and to select various options for the autocorrelation computations.

1.4.3 Data Transformations and Creating New Series

The SET Instruction

In most of your work with RATS, you will need to do at least a few data transformations, and you will often need to create new series from scratch. You can do that using the **SET** instruction, or one of several wizards. For our example, we need to define quite a few new series. We'll start by generating a couple of differenced series using **SET**. Execute the following instructions:

```
set ppidiff = ppi - ppi{1}
set m1diff = m1 - m1{3}
```

Be sure to put at least one blank space before the = sign.

Let's examine the first transformation. The **{1}** notation (which we refer to as "lag notation") tells RATS to use the first lag of PPI in the transformation. This creates a new series called PPIDIFF, and sets it equal to the first difference of PPI:

$$\text{PPIDIFF}_t = (\text{PPI}_t - \text{PPI}_{t-1}) \text{ for each entry } t \text{ in the default entry range.}$$

Note that PPIDIFF cannot be defined for 1959:1, because we do not have data for 1958:12, which would be the one period lag from 1959:1. RATS recognizes this, and defines entry 1959:1 of PPIDIFF to be a *missing value*.

Similarly, M1DIFF is defined as the three-lag difference of the series M1, so that $\text{M1DIFF}_t = \text{M1}_t - \text{M1}_{t-3}$. M1DIFF will be defined starting in 1959:4.

If you want to see the values of these series, click on or re-open the Series Window, highlight the series you want to view, and click on *View—Data Table* (or click on the  icon).

Next, we'll create some quarter to quarter growth rates, again using the "{L}" lag notation, and the "/ " division operator:

```
set grm2  = (m2 - m2{1})/m2{1}
set grppi = (ppi - ppi{1})/ppi{1}
```

We also need to create a three-period moving average term. First, we define PRATIO as the ratio of PPIDIFF (the first difference of PPI that we created above) to PPI. Then, we define PPISUM as the sum of the current and two lags of PRATIO. This could be done with a single **SET**, but is easier to read or modify this way:

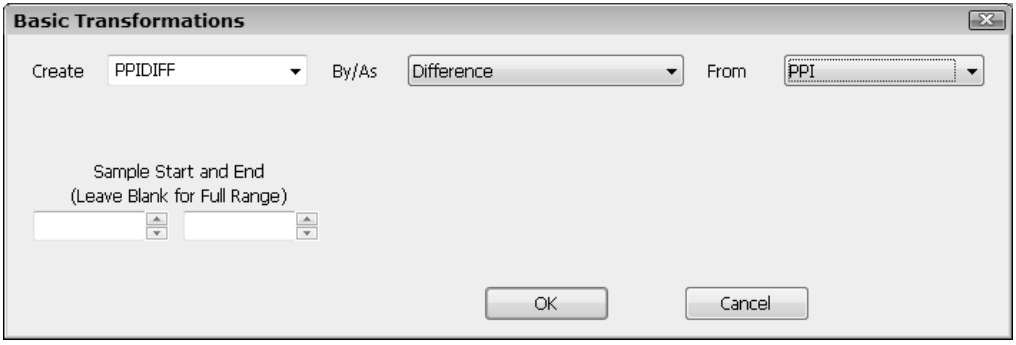
```
set pratio = ppidiff/ppi
set ppisum = pratio + pratio{1} + pratio{2}
```

Note that there are specialized instructions that can be used for some of these operations, such as **DIFFERENCE** and **FILTER**, but **SET** is the most important because it is the most flexible.

Data Transformation and Related Wizards

RATS offers several wizards for doing transformations, creating dummy variables, and other data-related operations. The *Data/Graphics—Transformations* operation is probably the most versatile.

For example, another way to create PPIDIFF as the first difference of PPI is to select *Transformations*, type in the name PPIDIFF in the "Create" field, select "Difference" from the "By/As" field, and select PPI in the "From" drop-down list.



You can use the "Create" field to type in or select the name of the series you want to create or redefine. The "By/As" field controls the type of transformation. Select "General–Input Formula" to enter your own formula for the transformation, or use one of the pre-defined transformations, including difference, log and square root.

The *Trend/Seasonals/Dummies*, *Differencing*, and *Filter/Smooth* operations offer similar series creation and transformation capabilities.

Keep Data Transformations in Your Program!!

While you could save these transformed series to a data file and then rely on those saved versions in subsequent analysis, we *strongly* recommend that you continue to read in the original source data, and retain the instructions used to generate your transformations as part of your RATS programs.

This will help ensure that you can reproduce your results later on, and will avoid any confusion about exactly how the transformed series were derived. (Consider how many ways there are to compute a "growth rate"). Also, even someone who has never used RATS should be able to tell exactly how the transformations were done simply by reading through the instructions.

1.4.4 Estimating Regressions

Our Equations

Recall that we want to estimate two regression equations:

$$(4) \quad Rate_t = \alpha + \beta_1 IP_t + \beta_2 \left(M1_t - M1_{t-3}\right) + \beta_3 PSUM_t + u_t$$

$$(5) \quad Rate_t = \alpha + \beta_1 IP_t + \beta_2 GRM2_t + \beta_3 GRPPI_{t-1} + u_t$$

We've done the necessary transformations, and are ready to estimate the models.

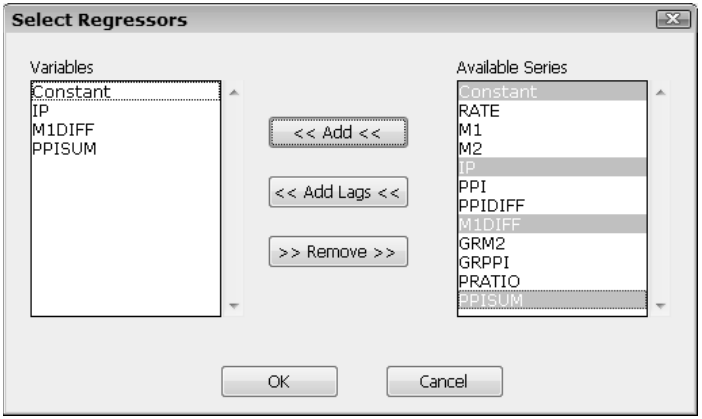
Estimating a Linear Regression

We'll start by using the Regression wizard to estimate equation (4). Select *Statistics—Linear Regressions*. In the dialog box, use the "Dependent Variable" drop-down list button to select `RATE`.

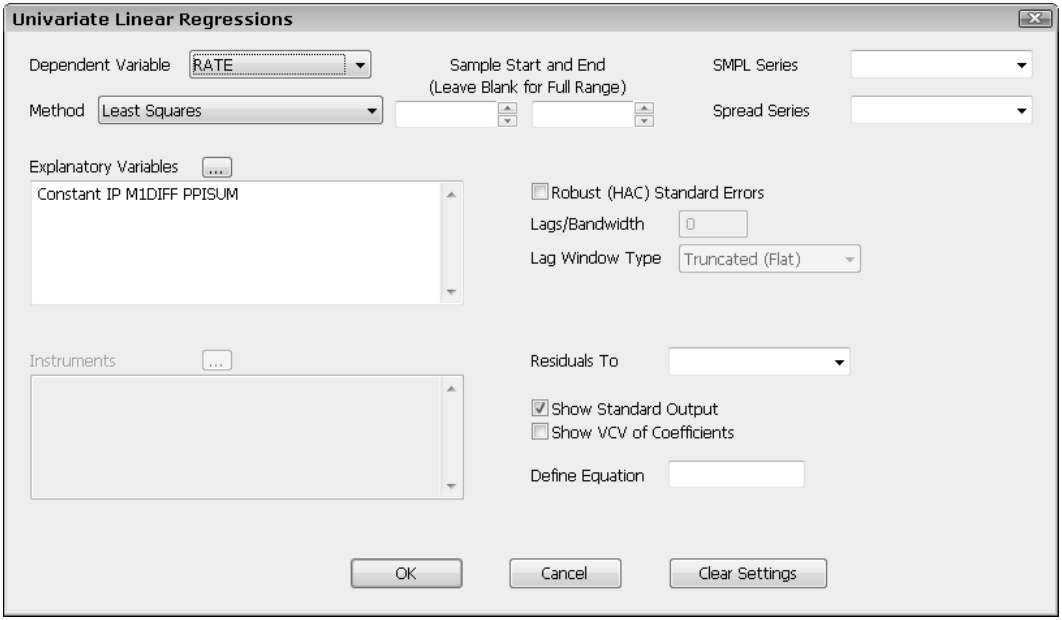
Next, we need to select the explanatory variables. You can type the regressors directly into the "Explanatory Variables" field (using blank spaces to separate variables), or you can click on the ... button, which opens the dialog box shown below.

You can add variables from the "Available Series" list to the regression by: double-clicking on a series name in the "Available" list; selecting one or more series and clicking the << Add << button; or using the << Add Lags << button to add lagged variables to the regression list

For this regression, add `CONSTANT`, `IP`, `M1DIFF`, and `PPISUM` to this list of regressors (`CONSTANT` is a special built-in series name, used to include a series of ones in a regression):



Click "OK" to close the list. The main dialog should now look like this:



Click "OK" to run the regression. The Wizard will generate the instruction below. This will actually be in upper case to help you recognize which instructions are generated using wizards—we edit them to lower case using the menu operation *Edit—To Lower Case* to keep with the style of the manual.

```
linreg rate
# constant ip m1diff ppisum
```

LINREG is the standard instruction for estimating linear regressions. Here, it regresses the dependent variable RATE on the independent variables IP, M1DIFF, and PPISUM, and includes a constant term (intercept) in the regression.

The line beginning with the # symbol is called a *supplementary card*. Supplementary cards are used with many instructions to supply additional information to an instruction—usually lists of series or equations. Supplementary cards always begin with the # character.

The results will be somewhat different from those shown in the text book, because the data are not identical due to revisions made since Pindyck and Rubinfeld extracted their data. **LINREG** estimates using ordinary least squares (OLS).

The output produced by this command is shown on the following page:

```
Linear Regression - Estimation by Least Squares
Dependent Variable RATE
Monthly Data From 1959:04 To 1996:02
Usable Observations                       443
Degrees of Freedom                        439
Centered R^2                         0.2526958
R-Bar^2                              0.2475890
Uncentered R^2                       0.8717009
Mean of Dependent Variable       6.0806546275
Std Error of Dependent Variable  2.7714419161
Standard Error of Estimate       2.4039938631
Sum of Squared Residuals         2537.0628709
Regression F(3,439)                   49.4816
Significance Level of F               0.0000000
Log Likelihood                    -1015.1499
Durbin-Watson Statistic               0.0816

    Variable                         Coeff      Std Error      T-Stat      Signif
******************************************************************************
1.  Constant                      2.11841571   0.42030566      5.04018  0.00000068
2.  IP                            0.06417324   0.00768853      8.34662  0.00000000
3.  M1DIFF                       -0.04183328   0.01485687     -2.81575  0.00508547
4.  PPISUM                       58.26459284   8.01033322      7.27368  0.00000000
```

Loading into a Report Window

For many uses, particularly when you're examining several possible models, the text output shown above is most convenient, since the output from all the models will be together in a single file. However, when you've finally settled on a specification, this *isn't* what you want, since the format for the numbers is fixed the way you see them in the editor. You *could* round them yourself and type them into a document, but we're trying to show you how to avoid that.

Instead, you can reload the same information into a Report Window. To do that, you can open the *Report Windows* submenu on the *Window* menu and select the "Linear Regression–Least Squares" report. The Report Windows list shows (up to) the last fifteen "reports" generated, with the most recent ones at the top of the list.

Once you've reloaded the regression into a Report Window, you can use the operations described on page Int–11 for reformatting numbers and copying the information for use in word processors. For example, you can select sections of the report and use *View—Change Layout* to change the number of decimal places for the selected numbers, and then use *Edit—Copy* or *File—Export* to get the results into another program or file.

In addition to reports generated by RATS instructions like **LINREG**, you can also create your own reports with the precise information that you want. That's a more advanced, but very useful, feature covered in Section 1.6.7 in the *User's Guide*.

Estimating the Second Equation

Now let's estimate equation (5). If you select the *Linear Regressions* operation again, you'll see the same dialog box as before, loaded with the settings from the last regression. To modify the equation, you can either edit the regressor list directly in the "Explanatory Variables" box, or you can click on the ... button to use the dialog box. In the dialog box, use >> Remove >> to delete M1DIFF and PPISUM from the list. Add GRM2, and use << Add Lags << to add lag 1 of GRPPI. The regressor list should look like this:

```
constant ip grm2 grppi{1}
```

Click "OK" in the main dialog box to execute the regression.

Here's the command that is generated and the output.

```
linreg rate
# constant ip grm2 grppi{1}
```

```
Linear Regression - Estimation by Least Squares
Dependent Variable RATE
Monthly Data From 1959:03 To 1996:02
Usable Observations                        444
Degrees of Freedom                         440
Centered R^2                         0.2215769
R-Bar^2                              0.2162694
Uncentered R^2                       0.8660034
Mean of Dependent Variable           6.0733783784
Std Error of Dependent Variable      2.7725545963
Standard Error of Estimate           2.4545026092
Sum of Squared Residuals             2650.8165458
Regression F(3,440)                       41.7484
Significance Level of F                    0.0000000
Log Likelihood                         -1026.6780
Durbin-Watson Statistic                    0.1730

    Variable                         Coeff      Std Error      T-Stat      Signif
********************************************************************************
1.  Constant                      1.20988109   0.52637190      2.29853  0.02199968
2.  IP                            0.06525852   0.00727000      8.97641  0.00000000
3.  GRM2                        136.19299812  34.77735583      3.91614  0.00010426
4.  GRPPI{1}                    101.94478474  17.17615119      5.93525  0.00000001
```

Range Parameters

Our first model used M1DIFF, which isn't defined for the first three periods, since it uses lag 3 of M1. This second model replaces that with GRM2, which only drops one point; the range for this regression is determined by the one-period lag of GRPPI which isn't defined until 1959:3. When you execute the **LINREG**, RATS will scan the data, determine that 1959:4 is the earliest possible starting point for the first regression and 1959:3 is the earliest possible date for the second. Its ability to handle such entry range issues automatically is a very powerful feature.

With time series models with lags, you need to be somewhat careful about doing

comparisons of estimates based upon different ranges, as we have with these two. The statistics which are sums (rather than averages) are especially affected by this: here the "Sum of Squared Residuals" and the "Log Likelihood", which shouldn't be compared when computed over different ranges.

Even the other statistics, which are based on averages, are only somewhat comparable. If we were seriously interested in choosing between these (we aren't, since both have *very* low Durbin-Watson statistics, so neither is a serious model for the interest rate), we should re-estimate the second regression over the same range as the first. While you can redo the Regression Wizard, and reset the "Sample Start" box, it's simpler to just edit the instruction and re-estimate it. We can change it to

```
linreg rate 1959:4 *
# constant ip grm2 grppi{1}
```

Almost any instruction which operates across a set of data allows you to give an explicit range, or to let RATS figure it out. Here, we need to override the standard handling, which for a regression uses the maximum range possible with the series involved. The estimation range has *start* and *end* parameters; we're fine with the *end* being the automatic value, which is what the * means. What we need to override is the *start*, which we do by giving the start date of 1959:4. If we wanted to restrict the top end of the regression range (say to the end of 1992), we would use

```
linreg rate 1959:4 1992:4
# constant ip grm2 grppi{1}
```

If we wanted to restrict the end of the range, but use as much data as possible at the start, we would use:

```
linreg rate * 1992:4
# constant ip grm2 grppi{1}
```

If you want to use the default for both range parameters, you can just leave them out *if* nothing comes after them. If there are trailing parameters, you either have to use * *, or you can also use the shorthand / to cover both. For instance, **LINREG** has a fourth parameter (for the generated residuals), so if we wanted to estimate this over the full range and save the residuals into the series U, we could use

```
linreg rate / u
# constant ip grm2 grppi{1}
```

While not common with time series data, it's also possible to skip entries out of the middle of the data set. We'll look at that in Section 1.6, which looks at cross section data.

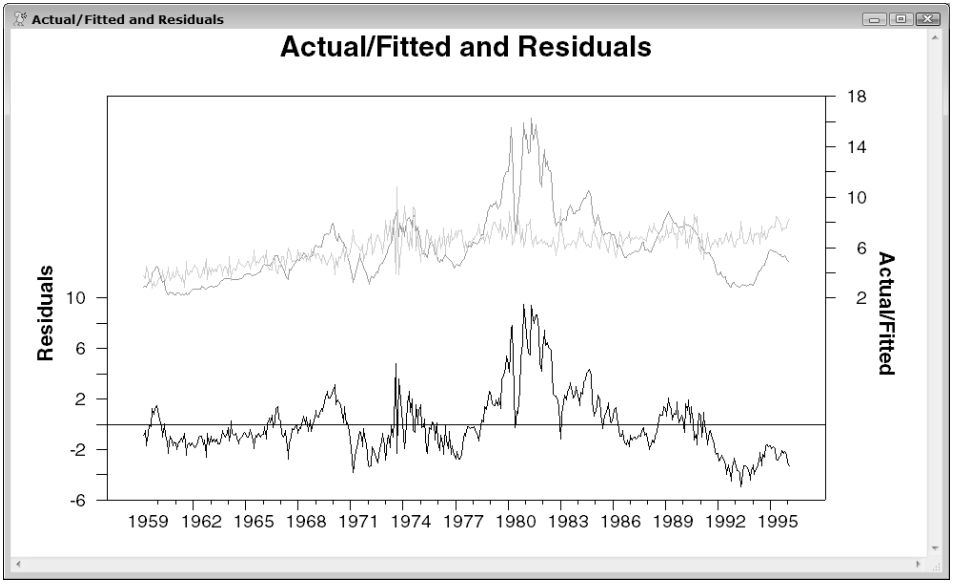
1.4.5 Learn More: Procedures

Procedures are collections of RATS commands that can be executed with a single "call" to the procedure, using a syntax nearly identical to that used for built-in instructions. While you may eventually want to write your own procedures—see Section 15.2 in the *User's Guide*—the main advantage for new users is that you have access to hundreds of existing procedures that greatly increase the scope of what you can do quickly and easily. We provide more than two hundred procedures with RATS.

As a simple example, type in and execute the following procedure call:

```
@regactfit
```

This executes a procedure called **@RegActFit** which generates a plot showing the residuals and actual and fitted values from the most recent regression:



If you want to see what the procedure code looks like, you'll find it on the file called REGACTFIT.SRC. It is a fairly simple procedure that provides a handy alternative to writing the necessary **SET** and **GRAPH** instructions yourself.

You can find many more, including procedures written by other RATS users from around the world, on our website. The website also includes a handy "Procedure Browser" for locating procedures of interest. You can find the browser in the "Resources" section of the website.

Each procedure is usually stored on its own text file. Some of the procedures included with RATS are described in the manual. For the others (including those available on our web site), see the comment lines at the top of the procedure file.

Executing a Procedure

The basic syntax for executing a procedure is:

```
@procedure name( options )      parameters
# < supplementary cards > (f needed)
```

Everything is the same as a standard RATS instruction except two things:

- The procedure name is preceded by `@`.
- You can't abbreviate the procedure name.

The following uses the procedure **@DFUNIT** (Dickey–Fuller Unit root test) to do unit root tests on the `RATE` series from the data set in this section:

```
@dfunit rate
```

Loading the Procedure

To use a procedure or function stored on a separate file, you need to have RATS execute the instructions that define the procedure. In most cases, RATS can do this automatically, by searching for a file with a `.SRC` extension whose name matches the name of the procedure. Otherwise, you can compile procedures stored on a file using the **SOURCE** instruction:

```
source   name of file with PROCEDURE or FUNCTION
```

If RATS can't load a procedure:

- Check to make sure you did a full installation, including all of the files supplied with RATS. See page Int–158 (installing RATS) for details.
- Do *File—Preferences* and check the "Procedure Directory" setting on the "Directories" tab. In order for RATS to find procedure files automatically, this should point to the directory containing your procedures.
- If you know where the procedure file is installed on your computer, include a **SOURCE** instruction (with a complete path and filename) prior to the procedure call to source in the procedure.
- You may be running an out-of-date version of RATS that did not ship with the procedure in question—you can do *Help—About RATS* to check which version you are using. See *www.estima.com*, email *sales@estima.com*, or call 847-864-8772 for information on updating. If you are using a network license at a company or university, your institution may already have the most recent version available—check with your system support staff.
- If all else fails, you should be able to download the procedure using the "Procedure Browser" on the "Resources" section of our website.

1.4.6 Learn More: Reading Data

The Default Directory and the OPEN Instruction

When you use the Data Wizard to read a file, RATS includes the full path to the file on the **OPEN** instruction. However, RATS also maintains a "default" directory setting, which is where it will look for (or create) files if you don't specify the full path.

You can use the menu operation *File—Directory* if you want to change the default directory for the current session. If you want to make that change permanent, you can reset that as part of your Preferences.

So, if you have the directory containing the RATS example files set as the default directory (which it should be when you first install the program), you could type the **OPEN** instruction shown earlier like this:

```
open data ExampleThree.xls
```

This approach (omitting the path) often works better if you will be sharing programs and data with another user who has a different directory structure on their system. Your colleagues just need to make sure that their programs and data files are stored in the same directory, and that they set that as the default directory in RATS.

Blank Spaces? Enclose in Quotes

If your path and filename includes any blank spaces, you need to enclose the entire string in single or double quotes, as in the code generated by the wizard. For example:

```
open data "C:\My RATS Files\ExampleThree.xls"
```

or

```
open data 'C:\My RATS Files\ExampleThree.xls'
```

For a path or filename that does not include any blank spaces, you don't need quote marks around the name, although there is no harm in including them.

Data from Multiple Files

You do not have to load all your data with a single instruction. You can use several **OPEN DATA** and **DATA** instructions to read from multiple data sources.

While you could use the *Data Wizard* operation for each file, you would need to make sure to use the same "Target Dates" setting each time. Otherwise, the date mappings specified by the last **CALENDAR** setting will no longer match up properly with data read in previously. So, if you need to read from multiple files or sources, you may find it easier to just type in the instructions directly (or use the Wizard for the first file, and then type the instructions for the additional files directly).

1.4.7 Learn More: Annotated Regression Output

This is the output from the first linear regression with a description of what each element shows. Note that this set of output is specific to least squares regression; other forms of estimation will be similar, but may leave out some statistics that aren't defined or have no usable interpretation.

```
(a) Linear Regression - Estimation by Least Squares
(b) Dependent Variable RATE
(c) Monthly Data From 1959:04 To 1996:02
(d) Usable Observations                          443
(e) Degrees of Freedom                           439
(f) Centered R^2                           0.2526958
(g) R-Bar^2                                0.2475890
(h) Uncentered R^2                         0.8717009
(i) Mean of Dependent Variable          6.0806546275
(j) Std Error of Dependent Variable     2.7714419161
(k) Standard Error of Estimate          2.4039938631
(l) Sum of Squared Residuals            2537.0628709
(m) Regression F(3,439)                      49.4816
(n) Significance Level of F                0.0000000
(o) Log Likelihood                         -1015.1499
(p) Durbin-Watson Statistic                    0.0816

  (q)Variable                 (r) Coeff   (s) Std Error  (t) T-Stat  (u) Signif
************************************************************************************
1.  Constant                 2.11841571   0.42030566      5.04018  0.00000068
2.  IP                       0.06417324   0.00768853      8.34662  0.00000000
3.  M1DIFF                  -0.04183328   0.01485687     -2.81575  0.00508547
4.  PPISUM                  58.26459284   8.01033322      7.27368  0.00000000
```

We'll use the following notation:

- $\mathbf{y}$ The vector of values for the dependent variable
- $\bar{\mathbf{y}}$ The sample mean of the dependent variable over the estimation range
- $\tilde{\mathbf{y}}$ The deviations from the mean of the dependent variable
- $\mathbf{e}$ The vector of residuals
- T The number of observations
- K The number of regressors

(a) The type of model and estimation technique used.

(b) The dependent variable

(c) If you are using a **CALENDAR**, RATS will list the frequency of the data and the beginning and ending of the estimation range. If you have date without a date scheme, this will be skipped.

(d) The number of usable entries in the estimation range: T

(e) The degrees of freedom: $T-K$

(f) The centered R^2 statistic: $1-\dfrac{\mathbf{e}'\mathbf{e}}{\tilde{\mathbf{y}}'\tilde{\mathbf{y}}}$

(g) The adjusted R^2 statistic ($\bar{R}^2$): $1-\left(\frac{\mathbf{e}'\mathbf{e}}{(T-K)}\Big/\frac{\tilde{\mathbf{y}}'\tilde{\mathbf{y}}}{(T-1)}\right)$

(h) The uncentered R^2 statistic: $1-\frac{\mathbf{e}'\mathbf{e}}{\mathbf{y}'\mathbf{y}}$

(i) The mean of the dependent variable: $\bar{\mathbf{y}}$

(j) The standard error of the dependent variable $\sqrt{\frac{\tilde{\mathbf{y}}'\tilde{\mathbf{y}}}{(T-1)}}$

(k) The Standard Error of Estimate: $\sqrt{\frac{\mathbf{e}'\mathbf{e}}{(T-K)}}$

(l) The Sum of Squared Residuals: **e'e**

(m) The regression F-statistic: $\frac{\tilde{\mathbf{y}}'\tilde{\mathbf{y}}-\mathbf{e}'\mathbf{e}}{(K-1)}\Big/\frac{\mathbf{e}'\mathbf{e}}{(T-K)}$

(n) The marginal significance level of the F, with $K-1$ and $T-K$ degrees of freedom.

(o) The log-likelihood: $-\frac{T}{2}\left(\log\left(\frac{\mathbf{e}'\mathbf{e}}{T}\right)+1+\log(2\pi)\right)$

(p) The Durbin-Watson statistic: $\sum_{t=2}^{T}\left(e_t-e_{t-1}\right)^2\Big/\sum_{t=1}^{T}e_t^2$

(q) The names of the explanatory variables. Lags are shown as `name{lag}`.

(r) The estimated coefficients.

(s) The standard error of the coefficient estimate.

(t) The t-statistic of the coefficient (coefficient/its standard error).

(u) The marginal significance level for a (two-tailed) test for a zero coefficient.

Goodness of Fit Measures

You will notice there are three versions of the R^2 statistic: the centered R^2, the $\bar{R}^2$ (R^2 adjusted for degrees of freedom) and the uncentered R^2. The centered and adjusted R^2 are typically the only ones of interest.

RATS also displays the mean and standard error of the dependent variable. These are simply statistics on the dependent variable, and tell you nothing about the accuracy of the regression model. They are the same values you would get by doing a **STATISTICS** instruction on the `RATE` series over the same range.

Regression *F* and Significance Level

These are only included if the regression includes a CONSTANT (or its equivalent in some set of dummy variables). This is the result of an *F*-test for the hypothesis that all coefficients (excluding CONSTANT) are zero. The numbers in parentheses after the *F* are the degrees of freedom for the numerator and denominator, respectively.

Log Likelihood

For a linear regression, this is the log likelihood of the data assuming Normal residuals. Note that this includes the "constants": the $1+\log(2\pi)$ terms only interact with T and not with the residuals and so could be dropped from any comparison of two linear regressions with the same number of observations. While some programs omit these, RATS always includes them in all calculations of likelihoods or any density functions.

Durbin-Watson Statistic

The Durbin-Watson statistic tests for first-order serial correlation in the residuals. The ideal result is 2.0, indicating the absence of first-order serial correlation. Values lower than 2.0 (and particularly below 1.0) suggest that the residuals may be serially correlated.

RATS always computes a Durbin-Watson statistic, even if you have cross-section data where "serial correlation" isn't likely to be an issue. However, you should keep in mind that the tabled values for the Durbin-Watson statistic are known to be invalid in a variety of circumstances, such as the presence of lagged dependent variables.

Coefficient Table

RATS attempts to come up with a common format for displaying the regression coefficients and standard errors. This makes the results easier to read than if the decimal points didn't align. The table lists all the variables in the regression, including the constant if you included it. The Coeff column lists the estimated coefficients for each variable. The *t*-statistic is the ratio of a coefficient to its standard error.

The significance level (see page Int–11) is for a two-tailed test for a zero coefficient. For models estimated by least squares, the *t*-statistic in the T-Stat column is treated as having a *t* distribution with $T-K$ degrees of freedom. If the model is estimated by some other technique, the *t*-statistic is treated as having a Normal distribution.

1.4.8 Learn More: The Series Window

The Series Window is opened by selecting the *View—Series Window* menu operation. This is associated with your current RATS session. If it's open, it will get updated as you add other series. If you close the Input Window, or do a Clear Memory operation, the Series Window will be emptied, since it shows only the series is working memory. While you can use menu operations to add or edit series using the Series Window, those series will also be cleared when you finish a RATS session, so you should only make changes like that if you are planning to export the data to a permanent file.

You can double-click on a series to open up a Series Edit Window (Section 1.3.1). If you select one series, it will be the initially chosen series in the drop-down box on such things as the Linear Regressions Wizard (for the dependent variable).

It has the following toolbar operations and associated menu operations:

 (Select All)	Shortcut for *Edit–Select All*. Selects all the series in the list.
 (Change Layout)	Shortcut for *View–Change Layout*. This is more useful in the possibly very large lists of series for data browsers, as it allows you to make a much shorter list by filtering out series which don't meet certain criteria.
 (Find)	Shortcut for *Edit–Find*. Searches for a value across series.
 (Import)	Shortcut for *File–Import*. Imports data from an external file. While this will work, it's usually better to use the Data Wizard, which does much the same thing, but keeps a record of the instructions.
 (Export)	Shortcut for *File–Export*. Exports the selected series in one of many formats.
 (New Series)	Shortcut for *Data/Graphics–Create Series (Data Editor)*. Opens a Series Edit Window for a new series.
 (Series Graph)	Shortcut for *View–Time Series Graph*. Displays a time series graph for the series.
 (Histogram)	Shortcut for *View–Histogram*. Displays a histogram plot for the series.
 (Box Plot)	Shortcut for *View–Box Plot*. Displays a box plot for the selected series.
 (Autocorrelations)	Shortcut for *View–Autocorrelations*. Computes and graphs the autocorrelations and partial autocorrelations of the series.

 (Basic Statistics) Shortcut for *View–Statistics*. Computes and displays descriptive statistics for the selected series, including the number of observations, mean, standard error, and maximum and minimum values of each series.

 (Cov./Corr.) Shortcut for *View–Covariance Matrix*. Computes and displays a covariance and correlation matrix for the selected series.

 (View Data) Shortcut for *View–Data Table*. Generates a (read-only) table of data for the selected series.

1.4.9 Learn More: Arithmetic Expressions

The arithmetic expressions in RATS form the basis of the general transformation instruction **SET**, as well as calculations involving scalar and array variables, which you do primarily with the instruction **COMPUTE** (see page Int–6 and *User's Guide* Chapter 1). RATS expressions are similar to those employed in many programming languages and applications. We give enough detail here to allow you to do almost any standard data transformation.

Constants

You can represent a numeric value using a variety of forms:

- with or without decimal points: `5  13.6  -.393`
- in scientific notation with a suffix of one of the forms: `En`, `E+n`, `E-n`, where `n` is a non-negative whole number: `2.3E5` (230000) `-.4E-4` (-.00004)

In addition, RATS provides the following two constants:

`%PI`	The constant π
`%NA`	The missing value code

Arithmetic Operators

RATS supports the following arithmetic operators:

+	addition
–	subtraction or negation
*	multiplication
/	division
^ or **	exponentiation
+=	Increment and assign (a += b is equivalent to a = a+b)
–=	Decrement and assign (a –= b is equivalent to a = a–b)
*=	Multiply and assign (a *= b is equivalent to a = a*b)
/=	Divide and assign (a /= b is equivalent to a = a/b)

You can use parentheses `()` to control the order of execution of the operations, and you can nest sets of parentheses if necessary. *You cannot use brackets* `[]` *or braces* `{}` *as substitutes, because these have other meanings in RATS.* In the absence of parentheses, operations are done in the following order:

1. Negation (– used to change sign)
2. Exponentiation (exception: `-a^b` does `a^b` first)
3. Multiply and divide
4. Add and subtract
5. Logical operators (see below)

If two operations have the same precedence, they are done from left to right, so `A-B+C` is equivalent to `(A-B)+C`. The one exception to this is `^`: `A^B^C` is the same as `A^(B^C)`. All of this is based upon natural order of operations in algebraic formulas. Just as a+b/c is interpreted as a+(b/c) in algebra, `A+B/C` is `A+(B/C)` in RATS.

Functions

RATS provides many useful functions (over 300). Functions accept zero or more arguments and return a value or set of values. Function arguments must be enclosed within parentheses () and there should not be a space between the function name and the "(". Some of the more important ones:

LOG(x)	natural log ($\log_e$)	**EXP(x)**	e^x
SQRT(x)	square root	**ABS(x)**	absolute value
SIN(x)	sine (of *x* in radians)	**COS(x)**	cosine (of *x* in radians)
%IF(x,y,z)	is *y* if *x* is non-zero and *z* if *x* is zero.		
%VALID(x)	is 0 if *x* is "missing" and 1 otherwise		

See Section 2 in the *Reference Manual* for a complete list of available functions.

Examples

Expression	RATS Code	Expression	RATS Code
$b^2 - 4ac$	`b^2 - 4*a*c`	$1/\left(1+y^2\right)$	`1/(1 + y^2)`
2^{-C}	`2^-c`	$a - bc^d$	`a - b*c^d`
$\log\left(1+w^2\right)$	`log(1 + w^2)`	$e^{-\lvert u\rvert}$	`exp(-abs(u))`

Logical and Relational Operators

Logical operators have many uses in RATS, one of which is creation of dummy variables. They code true/false statements, returning a one for true and zero for false. These are shown below ("A" and "B" can be any expression or number.)

Expression	Translates to the Statement
`A==B` (or `A.EQ.B`)	A is EQual to B
`A<>B` (or `A.NE.B`)	A is Not Equal to B
`A>B` (or `A.GT.B`)	A is Greater Than B
`A>=B` (or `A.GE.B`)	A is Greater than or Equal to B
`A<B` (or `A.LT.B`)	A is Less Than B
`A<=B` (or `A.LE.B`)	A is Less than or Equal to B
`.NOT.A`	A is NOT non-zero (that is, A is false)
`A.AND.B`	Both A and B are non-zero (true)
`A.OR.B`	Either A or B or both are non-zero (true)

- The logical operators are lowest in order of precedence, so `A+B==C*D` is done as `(A+B)==(C*D)`.
- The first six (the relational operators) take precedence over the last three, which are listed above in order of precedence:

  `A.OR.B.AND.C.OR.D` is done as `A.OR.(B.AND.C).OR.D`
- When testing for equality, be sure to use "==" (two equals), not "=" (just one)!!!

Examples

$$\texttt{Y>3.0}=\begin{cases}1 & \text{if } Y>3\\ 0 & \text{if } Y\le 3\end{cases} \qquad \texttt{T>=10.and.T<=20}=\begin{cases}1 & \text{if } T\ge 10 \text{ and } T\le 20\\ 0 & \text{Otherwise}\end{cases}$$

Integer vs Real Numbers

RATS distinguishes between integer and real (floating point) numbers. Integers are whole numbers which serve primarily as entry numbers, loop indices and subscripts. For instance, the date `1970:1` is processed as an integer, and subscripts such as `T-1` are integer expressions. A constant typed without a decimal point is considered to be an integer: `100.0` is real, `100`, `1`, `2` and `3` are integer.

When an operation mixes an integer with a real, the integer is converted to its equivalent as a real number before the operation is completed. For instance, if any one of `A`, `B` or `C` is real, `(A+B+C)/3` is the same as `(A+B+C)/3.0`.

- Division of two integers results in the integer quotient with no fraction, so `10/3` is 3 and `1/2` is 0. This may or may not be what you intend, so you need to be careful in such situations.
- If you need to force RATS to convert an integer to a real in a situation where it would not be done automatically, use the function `FLOAT`, for instance `FLOAT(T)`. Similarly, you can convert reals to integers with the function `FIX`. `FIX` truncates any remainder. If you want to round first and then convert to integer, do something like `FIX(%ROUND(X,-1))` which rounds to the nearest 10 and converts to an integer.

Missing Values

Operations involving a missing value produce a missing value as a result, except for equality and inequality (== and <>). Because missing values propagate automatically, you don't have to worry about making special tests to insure that an expression only operates on valid numbers.

An invalid operation of any form will also produce the missing value code as a value. Examples are `SQRT(-1.0)` and `1/0`. RATS does not distinguish between illegal operations (such as `SQRT(-1.0)`) and operations which could be assigned a "value" of infinity, such as `1./0`. Within an expression, you can represent a missing value only as `%NA`; not as `NA` or `.` or any of the alternatives that can be used on data files.

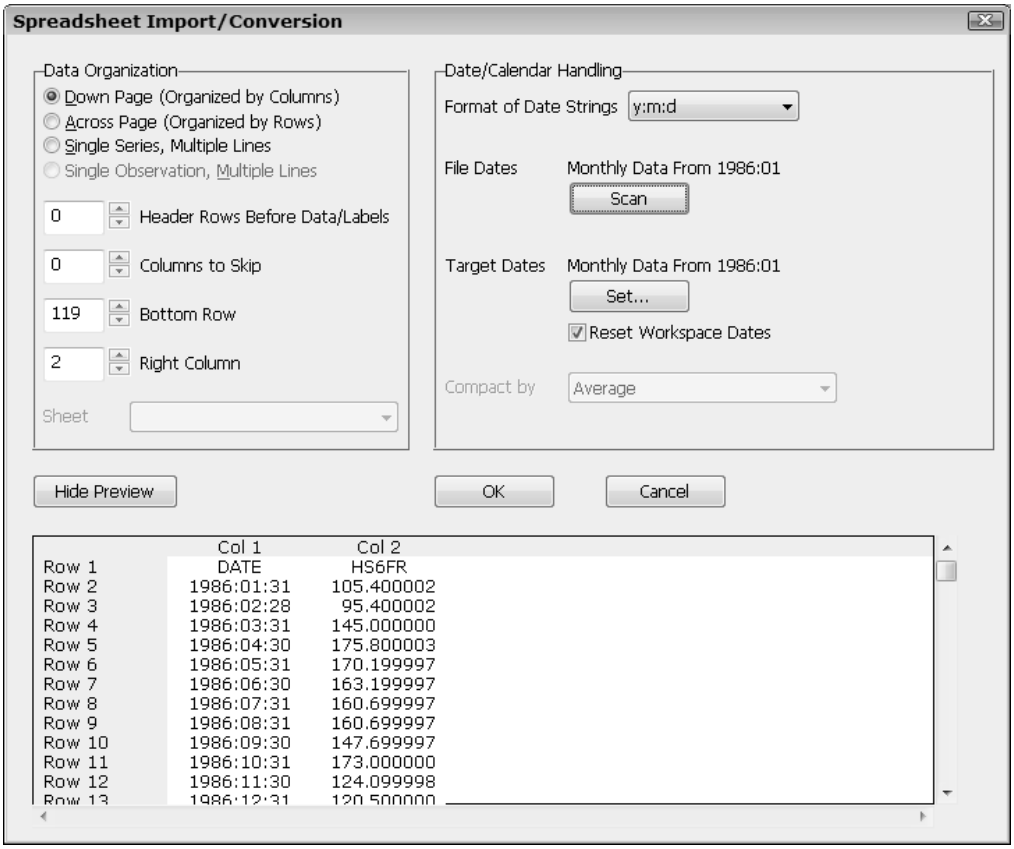
Referencing Series Elements

You can access a particular entry of a series by listing the entry or element in parentheses immediately after the variable name. For series, you would use an expression of the form "`seriesname(entry)`", such as: `FOODPROD(5)` or `DISPINC(1939:1)`. As we've already seen, in a **SET** instruction you reference the current value of a series (the value at the entry being computed) as `seriesname` alone, and its lag as `seriesname{lag}`.

1.5 Example Four: Time Series Graphs and Analysis

Now we'll look at an example to demonstrate some time series filtering and analysis techniques, and explore the graphing capabilities of RATS. It uses monthly data on housing starts in the United States, and is also from Pindyck and Rubinfeld (1998), page 480. The instructions are provided on the file `ExampleFour.rpf`. Before starting, close the Input Window and start a new one (with *File—New: Editor/Text Window*). This time, we will use separate input and output windows. Once the editor window is open, hit the toolbar.

The data file we want is called `EX152.XLS`. Use the *Data Wizard (Other Formats)* operation to open the file. Click on click on "Scan" to process the date information. The dialog box should look like this:

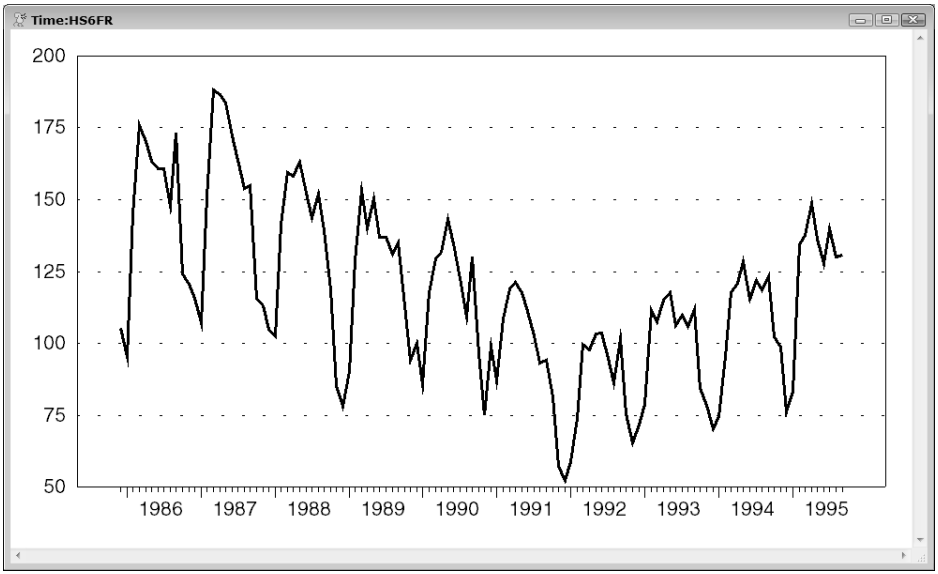


The dates are in the form "year:month:day", but the file only contains one observation per month. RATS is able to recognize this, and correctly identifies this as monthly data. Click on "OK". RATS will generate and execute the following instructions:

```
open data ex152.xls
calendar(m) 1986:1
data(format=xls,org=columns) 1986:1 1995:10 hs6fr
```

Examine the Data

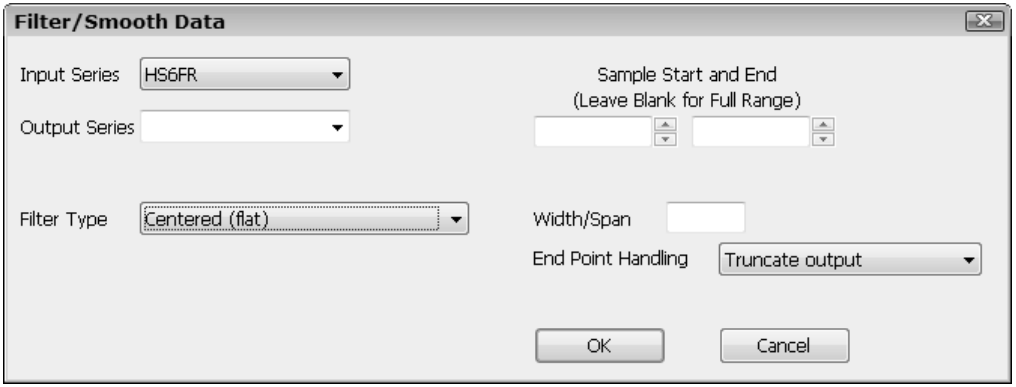
As always, we recommend that you examine the data before proceeding. For example, select *View—Series Window* from the menu, click on the `HS6FR` series, and do *View–Time Series Graph*. You should see the following:



As you would expect, the series exhibits a very strong seasonal behavior. There also appears to be a decreasing trend over the first few years, followed by an increasing trend over the remaining years.

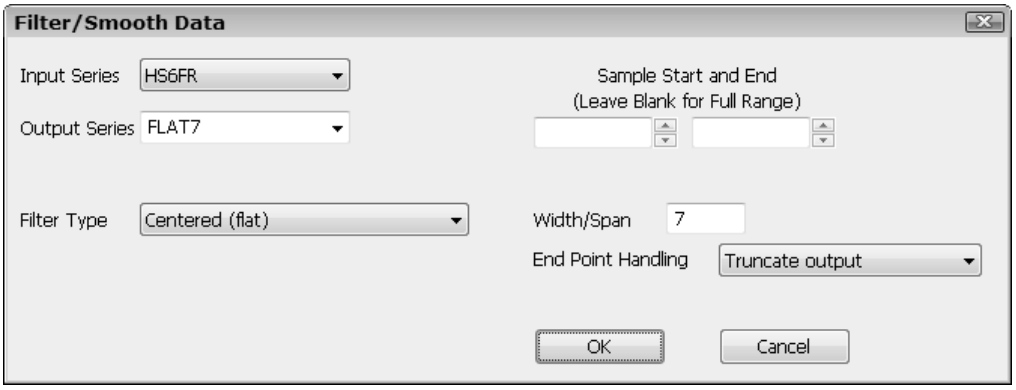
1.5.1 Filtering and Smoothing

We will examine some techniques for smoothing the data. Start by selecting *Data/Graphics—Filter/Smooth*, which will bring up this dialog box:



Because we only have one series in memory (HS6FR), RATS automatically selects it as the "Input Series". The default filter type is a centered moving average filter, which is what we want ("centered" means that for a window width of $2N+1$ periods, the filtered value at time T is computed using observations $T-N$ through $T+N$).

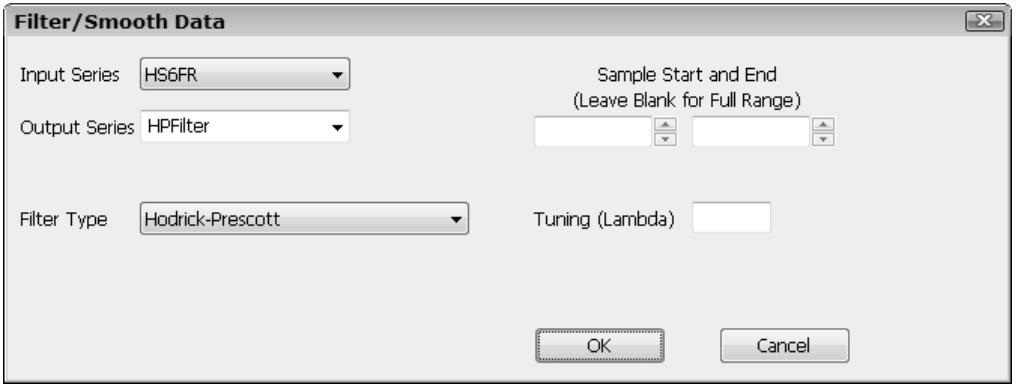
Type in FLAT7 as the name for the "Output Series", and enter 7 as the "Width":



Click on "OK". RATS will generate a **FILTER** instruction with the appropriate options:

```
filter(type=centered,width=7) hs6fr / flat7
```

This seven-period moving average should smooth the series considerably, but will probably retain some of the seasonal behavior. To reveal the underlying trend behavior of this series, we can try the Hodrick-Prescott (HP) filter. It is designed to separate the trend behavior of a series from its cyclical component. Select *Filter/Smooth* again. This time, you'll have to select HS6FR as the input series (RATS doesn't guess, now that there is more than one series in memory). Enter HPFILTER as the "Output Series" and select "Hodrick-Prescott" as the "Filter Type". You can leave the Tuning parameter field blank to use the default value. The dialog should look like this:

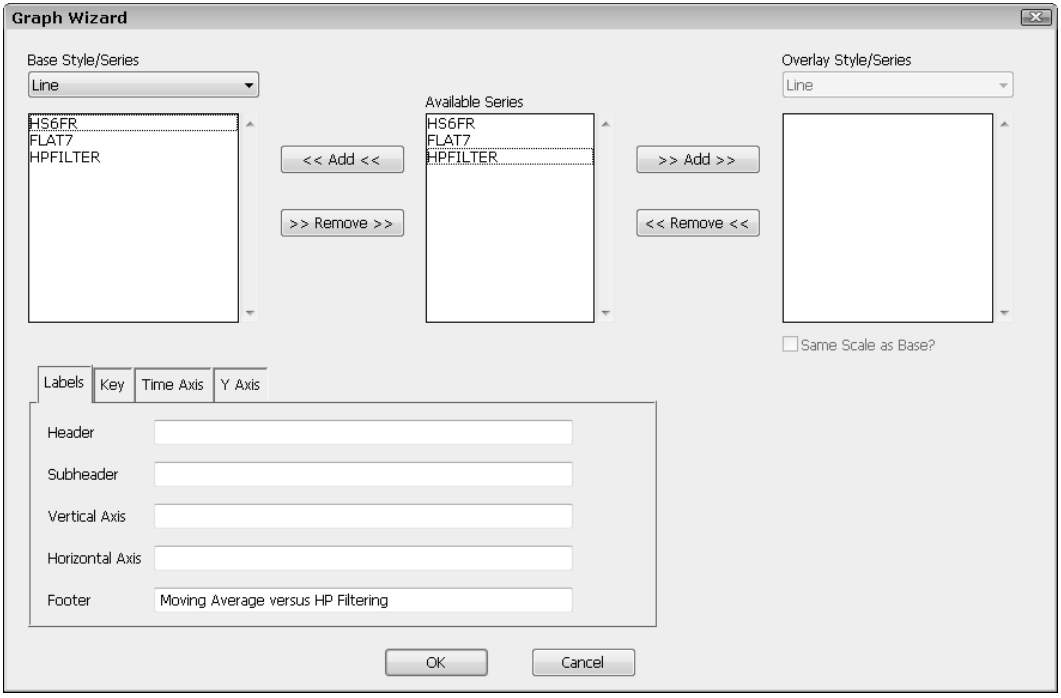


1.5.2 Graphing the Data

Let's examine the results by generating a graph that includes all three series. This time we will use the *Data/Graphics—Graph* operation on the menu. Unlike *View–Time Series Graph*, this actually generates a RATS instruction, making it easy to reproduce the graph later. It also gives us more control over the appearance of the graph.

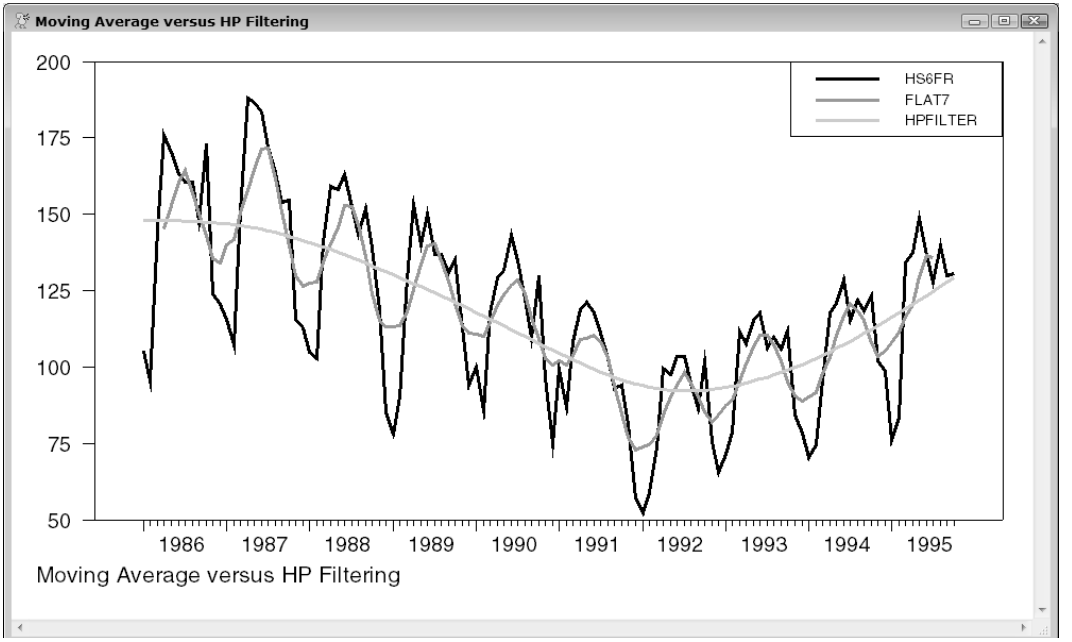
Select the *Graph* operation, which displays the dialog box shown below. The "Series" field in the middle lists all the series available in memory. We want to graph all three, so click on `HS6FR` and then `<Shift>`+click on `HPFILTER` to highlight the whole list. Click on the << Add << button to add the three series to the "Base Series" list on the left (as opposed to the "Overlay Series" list on the right, which is used to graph some series using a second vertical scale).

That's all you need to do to produce a graph, but let's add some features. First, type in "Moving Average versus HP Filtering" in the "Footer" field at the bottom. The dialog should now look like this:



Next, click on the "Key" tab, and select "Upper Right (Inside)" as the position for the key. Then click "OK" to generate the graph.

Here's the resulting Graph Window:



As expected, FLAT7 is considerably smoother than the original series, but still shows a significant amount of seasonality. The HPFILTER series, however, seems to fit the overall trend behavior of the original series quite well, with nearly all of the cyclical behavior and short-term fluctuations removed.

Here is the **GRAPH** instruction generated by the wizard. Due to space constraints, we use the "line continuation" symbol $ here, which tells RATS that a command is continued on the next line:

```
graph(style=line,footer="Moving Average versus HP Filtering",$
      key=upright) 3
# hs6fr
# flat7
# hpfilter
```

The main instruction includes several options: STYLE, which controls the way data is represented; FOOTER, which adds a text label below the graph; and KEY, which adds a key. Both STYLE and KEY select from a specific list of choices—see the description of **GRAPH** for a list of the possible choices. The number "3" at the end of the (continued) **GRAPH** line tells RATS that you are graphing three series.

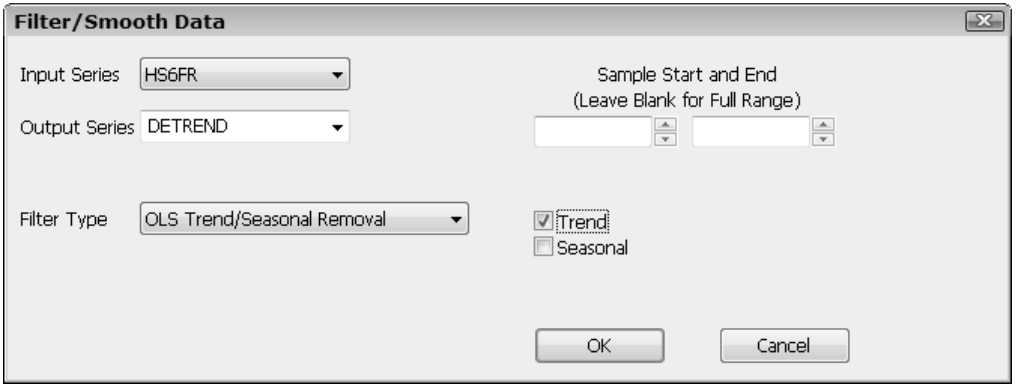
When you are done viewing the graph, close the graph window or click on the input window to bring it back to the front.

1.5.3 Detrending, Exponential Smoothing, Forecasting

Pindyck and Rubinfeld also examine the use of exponential smoothing techniques. First, they remove the trend component from the series by regressing it on a linear trend series. Then they apply exponential smoothing to get a smoothed version of the detrended series. They then add back the trend component removed in the first step to get a smoothed version of the original series.

This can actually be done in a single step in RATS. We'll demonstrate that in a moment, First, here is the step-by-step version.

The *Filter/Smooth* operation offers one way of removing a trend using a regression. Select *Filter/Smooth*, and choose HS6FR as the input series. Type in DETREND as the output series, select "OLS Trend/Seasonal Removal" as the filter type, and turn on the "Trend" checkbox:



Click "OK" to generate the DETREND series. The instruction looks like this:

```
filter(remove=trend) hs6fr / detrend
```

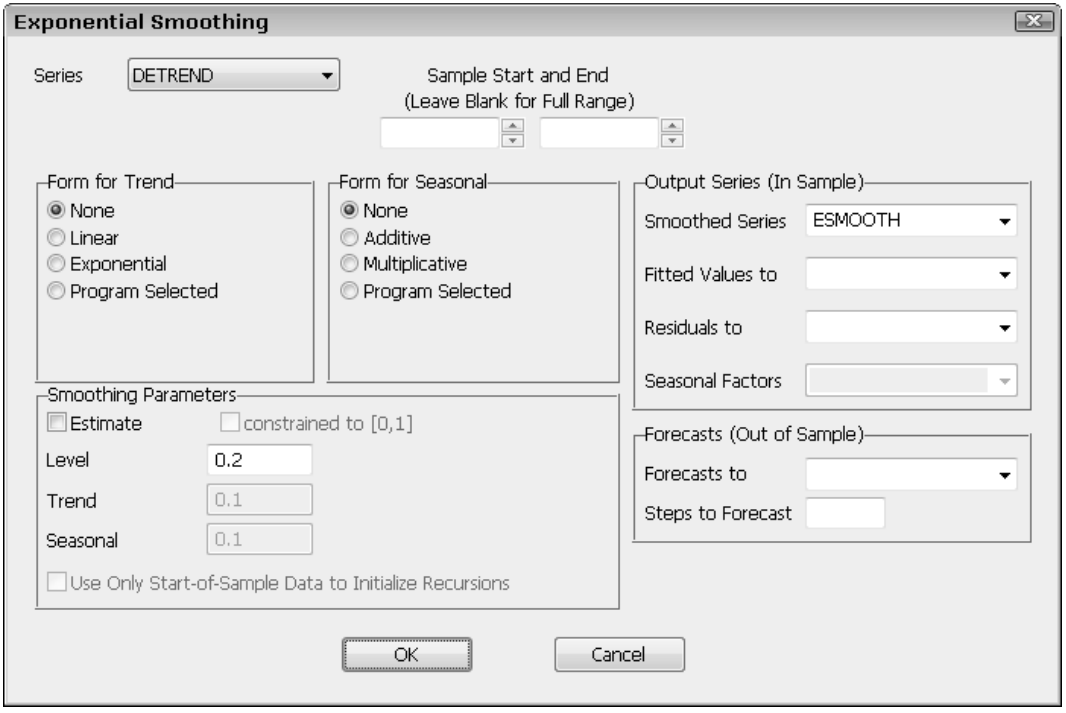
Next, we need to compute and save the trend component that was removed, by subtracting the detrended series from the original series. You could use the *Data/Graphics—Transformations* wizard to do this, but you may find it easier just to type in the **SET** instruction yourself:

```
set removed = hs6fr - detrend
```

Now, we can do our exponential smoothing on the detrended series. Because it is time-series specific, the *Exponential Smoothing* wizard is located on the *Time Series* menu. Go ahead and select *Exponential Smoothing*.

Select DETREND as the input series, "None" as the "Form for Trend", and type in the name ESMOOTH for the output series. The textbook uses a value of 0.2 for the "level" smoothing parameter α, so enter that value in the "Level" field.

The dialog box should look like this:



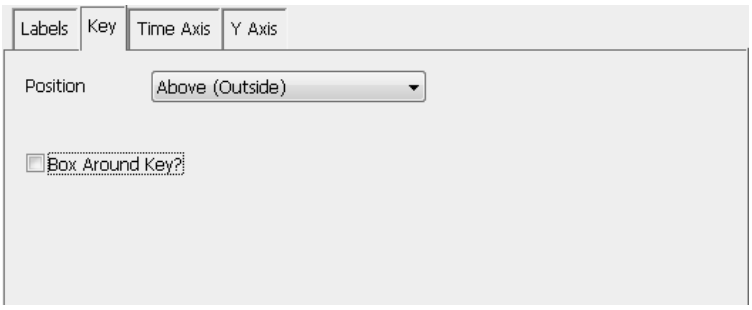
Click on "OK" to generate the **ESMOOTH** instruction:

```
esmooth(alpha=.2,smoothed=esmooth) detrend
```

Now we need to add back in the trend component removed earlier. We can replace the existing values of ESMOOTH with the sum of those values and REMOVED:

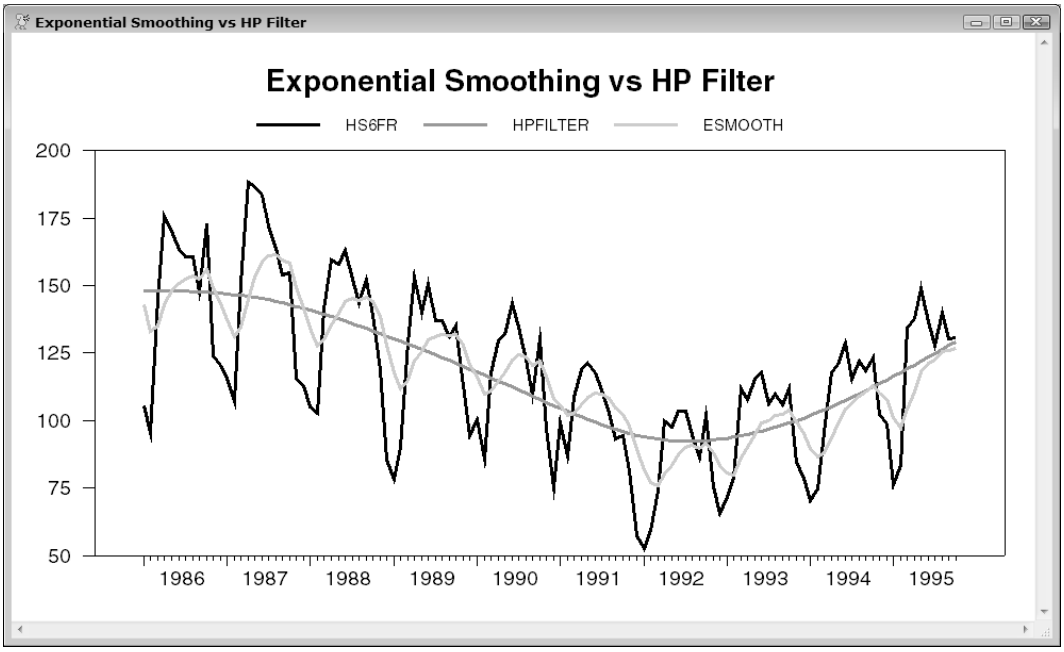
```
set esmooth = esmooth+removed
```

Now, select the *Graph* operation again, and choose HS6FR, HPFILTER, and ESMOOTH as the series to be graphed. To label the graph, enter "Exponential Smoothing vs HP Filter" as the header. On the Key tab, choose "Above (Outside)" and turn off the "Box Around Key?":



This generates the instructions and graph shown below:

```
graph(style=line,header="Exponential Smoothing vs HP Filter",$
     key=above,nokbox) 3
# hs6fr
# hpfilter
# esmooth
```

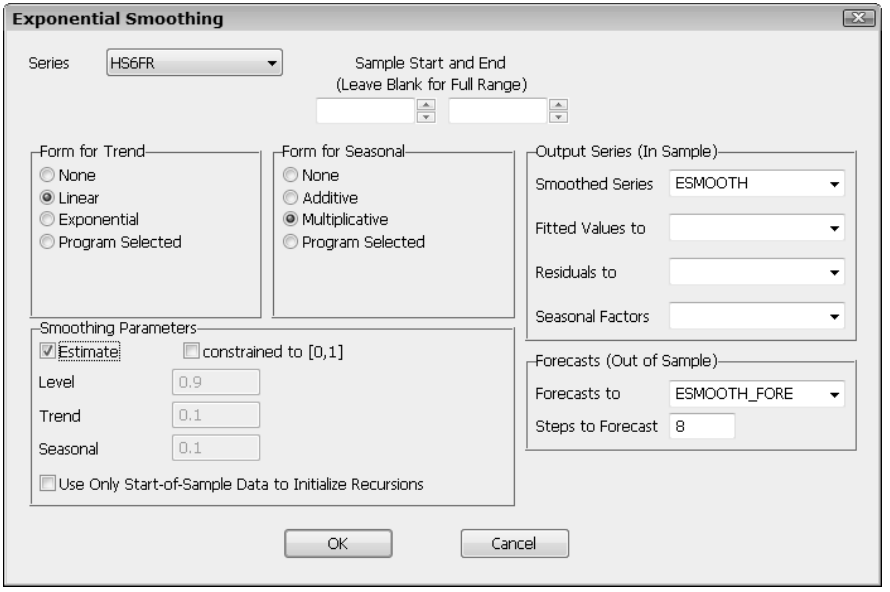


Modelling the Trend and Seasonal Behavior

As noted earlier, there is an easier way to do this, which is to include a component for the trend behavior in the exponential smoothing model. Select *Time Series—Exponential Smoothing* again. This time, select the original series HS6FR as the series to model. Then, select the Linear choice for the trend model in the "Form for Trend" box, and turn on the "Estimate" switch in the "Smoothing Parameters" box (which tells RATS to estimate the smoothing parameters used in the model).

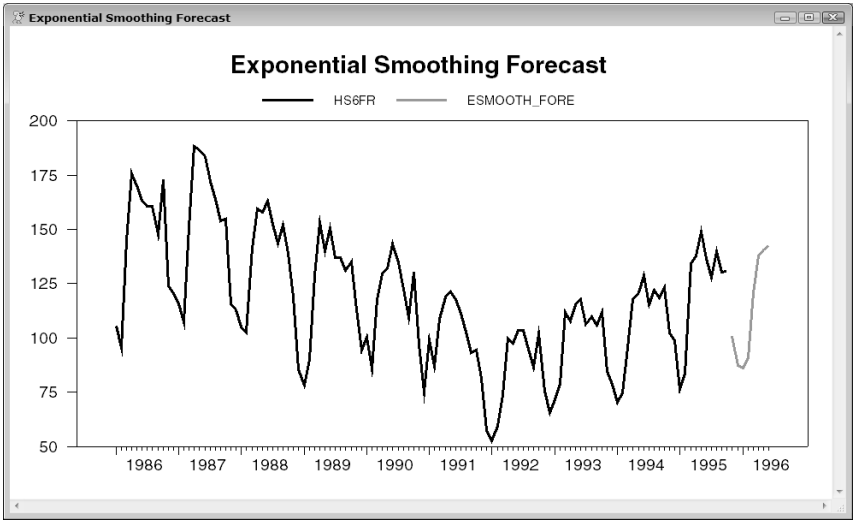
That's all you need to reproduce the previous analysis, but let's go a couple of steps further, by also including a seasonal term in the model and using the resulting model to produce forecasts.

In the "Form for Seasonal" box, select "Multiplicative". Provide a name for the smoothed series (we used ESMOOTH again here). Finally, enter a name in the "Forecasts to" box to hold the forecasts (we use ESMOOTH_FORE), and enter 8 as the number of forecast steps to compute. The dialog should look like:



When you are ready, click "OK" to do the smoothing. The resulting instruction is shown below, along with a **GRAPH** instruction that plots both the original series and the forecasts.

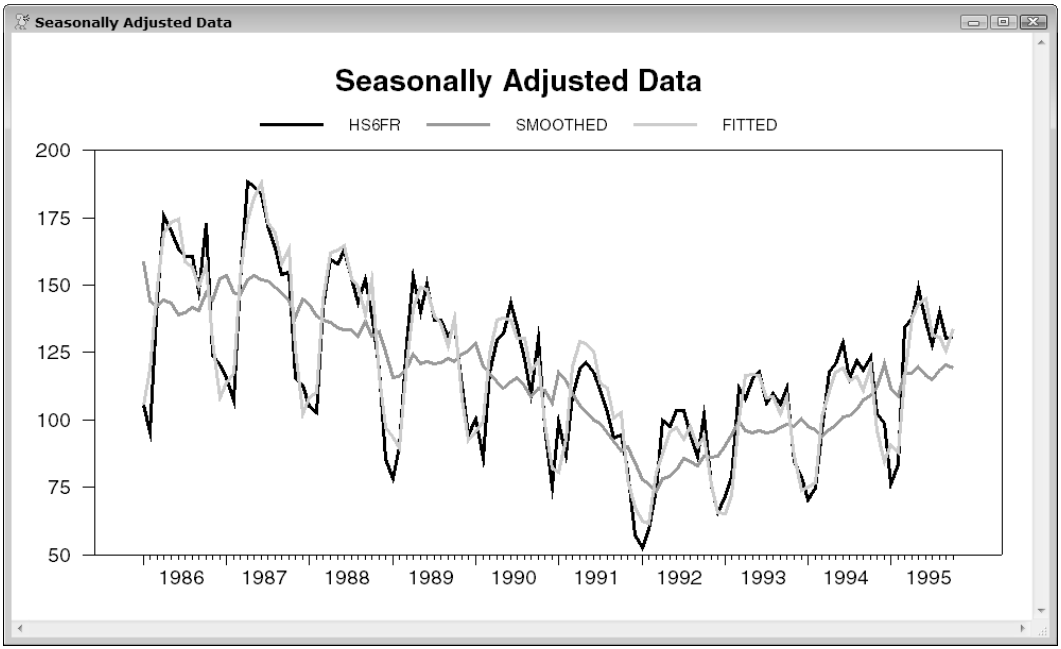
```
esmooth(trend=linear,seasonal=multiplicative,estimate,$
       smoothed=esmooth,forecast=esmooth_fore,steps=8) hs6fr
graph(style=line,header="Exponential Smoothing Forecast", $
     key=above,nokbox) 2
# hs6fr
# esmooth_fore
```



Seasonally Adjusting Data

If you just want to seasonally adjust (de-seasonalize) the data, you would include a seasonal component in the **ESMOOTH** model, but omit the trend term. The **ESMOOTH** below does just that, and saves both the smoothed (seasonally adjusted) values and the fitted values. This also uses the ESTIMATE option to estimate the values of the model coefficients.

```
esmooth(seasonal=multiplicative,estimate,$
        smoothed=smoothed,fitted=fitted) hs6fr
graph(header="Seasonally Adjusted Data",key=above,nokbox) 3
# hs6fr
# smoothed
# fitted
```



As you can see from the graph, the seasonal component has been removed from SMOOTHED, leaving the underlying trend and short-term fluctuations.

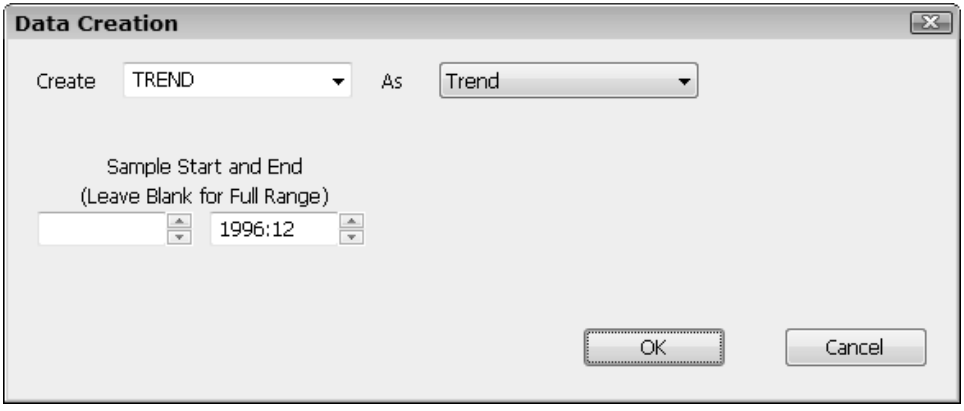
Exponential smoothing offers a relatively simple form of seasonal adjustment. You can also treat seasonality using state-space models (*User's Guide*, Chapter 10), or, if you have a Professional level of RATS, you can use the more sophisticated Census X11/X12 adjustment procedure.

1.5.4 Regression-based Forecasting

The graph shows that a general upward trend from 1992 on in the seasonally adjusted series. Suppose we want to generate an out-of-sample forecast of that trend through 1996. One way to do that is to do a simple regression on a constant and a trend series.

First, we need to define the trend series. Because we'll be forecasting out-of-sample, we need to define that trend series beyond the end of the actual data to include the forecasting range (the special built-in CONSTANT series is automatically available at all time periods).

Select *Data/Graphics—Trend/Seasonals/Dummies*. Enter TREND as the series name, and 1996:12 as the ending date:



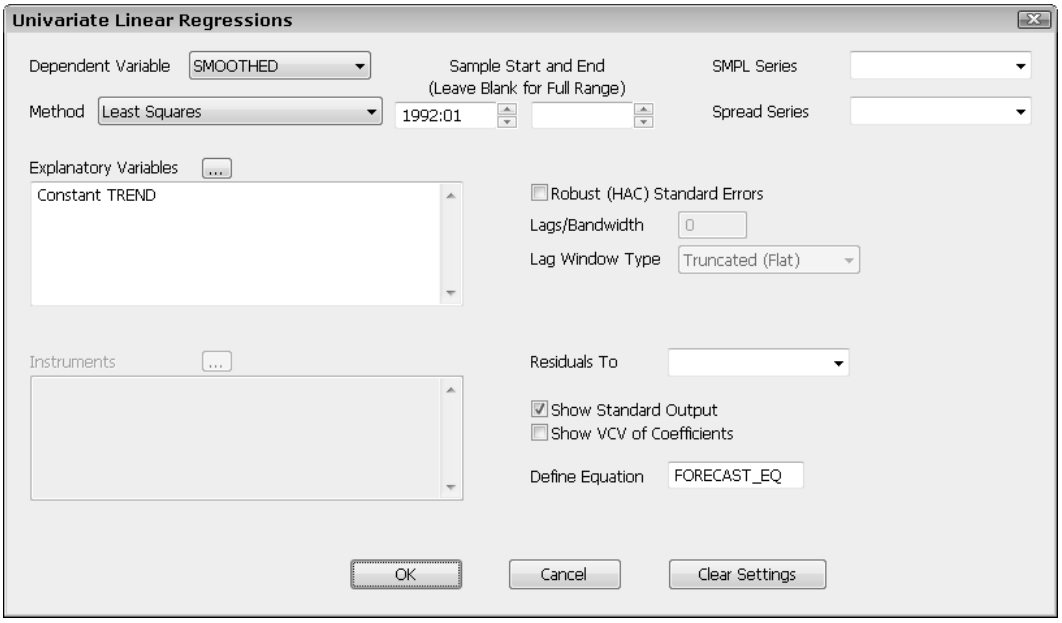
This generates the instruction:

```
set trend * 1996:12 = t
```

which uses the (normally omitted) *start* and *end* parameters on **SET**. The * symbol used for the *start* parameter tells RATS to use the earliest date in the default range (entry one in this case). Specifying 1996:12 as the end date defines the trend through the end of our forecast range.

The T on the right side of the equal sign is a reserved variable in RATS. It returns the current entry being set or evaluated. So, this creates a series called TREND that contains the number 1 in entry 1, the number 2 in entry 2, and so on.

Next, select *Statistics—Linear Regressions*, and enter SMOOTHED (the seasonally adjusted series) as the dependent variable, 1992:1 as the starting date, CONSTANT and TREND as the explanatory variables, and FORECAST_EQ as the equation to define. The dialog box should look like this:

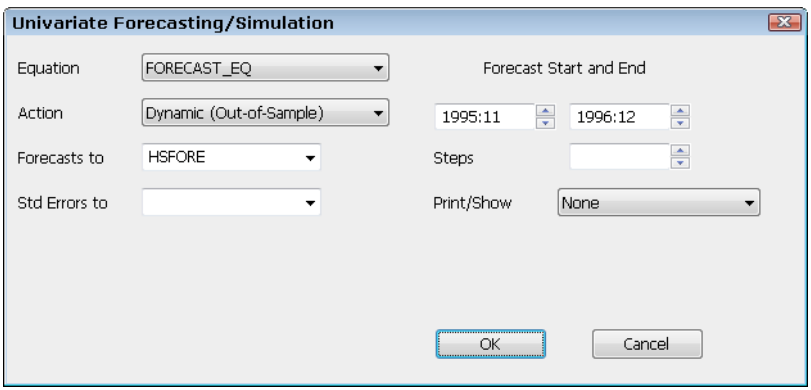


This generates the following **LINREG**:

```
linreg(define=forecast_eq) smoothed 1992:1 *
# constant trend
```

The DEFINE option creates an equation called FORECAST_EQ containing the structure and estimated coefficients from the regression. We'll use that to produce forecasts.

Now select *Time Series—Single-Equation Forecasts*. The "Equation" field tells RATS which equation you want to forecast. The default choice ("Last Regression") would actually work, because the last regression performed is the one we want to forecast, but go ahead and select FORECAST_EQ. Then, enter 1995:11 and 1996:12 as the starting and ending date for the forecasts. Finally, enter HSFORE in the "Forecasts To" field to save the computed forecasts into a series with this name:

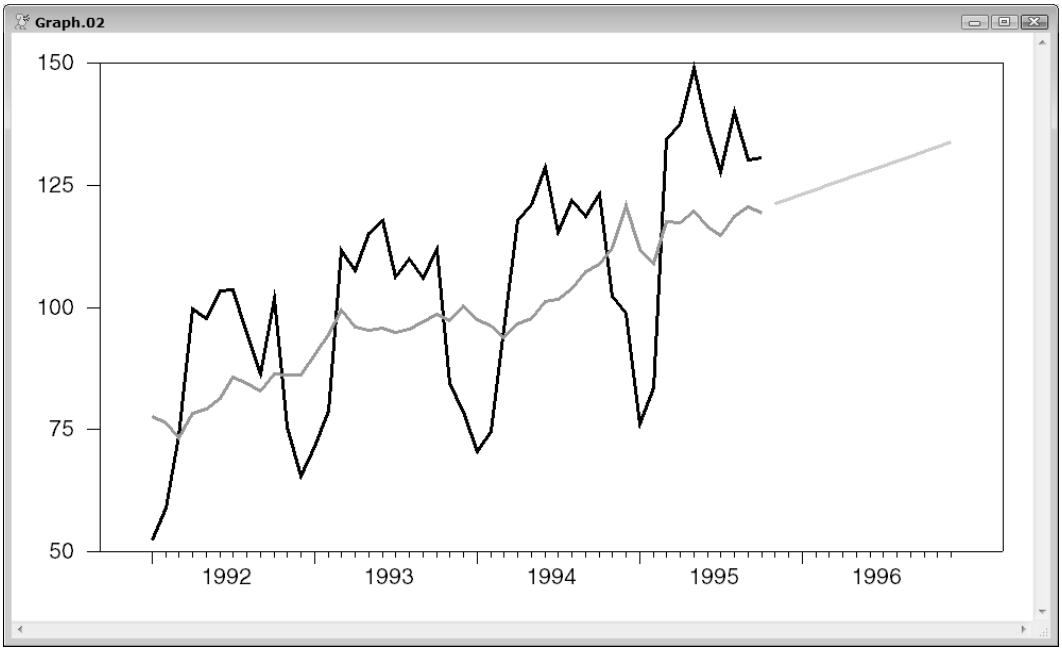


Click "OK" to compute the forecast. This generates the following **UFORECAST** instruction (the U stands for univariate, or single-equation):

```
uforecast(from=1995:11,to=1996:12,equation=forecast_eq) hsfore
```

To see the results, execute the following—note the use of the date ranges to limit the graph to our estimation and forecasting range (we don't need a range on HSFORE because it is only defined over the range we want to see):

```
graph 3
# hs6fr 1992:1 *
# smoothed 1992:1 *
# hsfore
```



1.5.5 Learn More: Graph Windows

Graph Windows are, not surprisingly, windows used to display graphs. Each graph generated gets its own window. You may note that Graph Windows have relatively few "toolbar" icons. With RATS, you create the appearance that you want using the graphics commands, rather that adding them (manually) afterwards. This gives you graphs that are perfectly reproducible. Chapter 3 is devoted to the tools for making high-quality graphs. RATS allows all types of special effects, such as shading regions, adding labels and text, and configuring colors, line and fill patterns.

Programs can sometimes generate a large number of graphs. To reduce clutter, you might want to minimize graphs that you don't need to look at right now (but want to keep available)—you can always reload them from the *Window* menu. There's also a *Close All Graphs* on the *Windows* menu which will close (permanently) all the current graph windows.

Resizing Graphs

You can resize a graph window by clicking and dragging a corner of the graph. RATS will automatically adjust the elements of the graph (font size, etc.) to fit it within the new window. If you squeeze the graph down to be very narrow in one dimension or the other, the fonts may become hard to read. However, as long as you keep the proportions within "normal" ranges, it should remain readable.

When you print a Graph Window or export it for use with a word processor, RATS normally uses proportions close to a "golden ratio", with the width being about 1.5 times the height. This will not necessarily match the proportions that you are seeing on the screen. If you've changed the shape of the graph and want to use those proportions, click on the "Fix" button (). To cancel that, click on the ().

Color or Black and White

By default, graphs are displayed in color, with different colors used for each series. If you print a graph to a black and white printer, RATS will automatically convert to using different patterns to distinguish series. You can preview what the graph will look like in black and white mode by clicking on the  toolbar button. The precise set of patterns (or colors, for that matter), can be configured, if the standard ones don't work well or your publication needs a standard "style". See page Int–149.

Saving and Copying Graphs

You can use *File—Save As...* or *File—Export...* to save a graph to disk. If you want to be able to re-open the graph in RATS, save the graph in our RGF (RATS Graph Format). If you want to import the graph into another document, use one of the other formats, with PostScript usually being the best option. You can also include instructions in your program to save graphs automatically (page Int–153).

You can also use *Edit–Copy*, `<Ctrl>+C`, or Copy on the contextual menu to copy a graph to the clipboard for pasting into another application. The formats available depend on the platform you are using (page Int–129).

1.5.6 Learn More: Long and Short Program Lines

Continuation Lines, $ Symbol

RATS will accept instructions of any physical length. Sometimes, though, a line becomes so long that it is difficult for *you* to handle. To split a line into more manageable segments, put a **$** character *at the end of the unfinished line*. You just have to be careful where you break a line. Don't break up "tokens", things like numbers, variable names and quoted strings. The best locations are before or after commas or arithmetic operators—with *after* being the easiest to read, since it's clear that the expression isn't done. An example is

```
set gnp = consumption + investment + govtexpend + $
   export - import
```

There is no limit on the number of continuation lines allowed for a single instruction.

Some statistical and programming languages require a symbol, often a semi-colon (;), at the end of each command to signal to the program that a command is complete. That is not the case with RATS—it automatically assumes a command is complete when it hits the end of the physical line of text. Thus the only time you *have* to include a symbol (the $) at the end of a physical line is if the command *isn't* done, and continues to the next line of text.

Multiple Statements on a Line, the ; Symbol

You actually *can* use the semi-colon (;) in RATS to signal the end of a command, but this is only necessary if you want to include multiple commands on a single line. This is usually only a good idea for a set of short instructions which you see almost as one, or for attaching short comments to instructions. Examples:

```
set gdp = log(gdp) ; set m1 = log(m1)
smpl 1960:1 1985:3        ;* Set Sample for Remaining Instructions
```

However, if you copy a line out of a math package which uses ; to end lines, you don't have to delete that, as there is no harm in a ; if there is nothing after it. For instance, the translation of

```
nsave = 10000;
nburn = 0;
```

to RATS can be done by just inserting the instruction name **COMPUTE** (or **COMP** or **COM**, only the first three characters matter) to the start of each line:

```
comp nsave = 10000;
comp nburn = 0;
```

1.5.7 Learn More: RATS Program Files (RPF)

At this point, you should have a complete Editor Window with the instructions for the analysis that we've done with this housing data series. The advantage of using the separate I/O windows is that we have the input commands by themselves. We will now save this as a RATS Program File which we can run later.

We use the extension `.RPF` for program files. These are simple text files, so you can open them with any word processing program, but the `.RPF` extension allows us to associate the files with the RATS software.

Saving a Program

Select *File—Save* (or *File—Save As...* or the  toolbar) and assign the file a name. For example, you might save this as `MyProgram.rpf`. You can also save the file when you close the window—you'll be first asked if you want to save the changes, and if you answer "Yes", you'll be prompted for the file name.

Executing a Saved Program

Now, close all the open windows (using *Windows—Close All*). Because you are closing the input window (with your instructions), this will clear the RATS memory as well. Do *File—Open...* and re-open the file you just saved. Again, do the  to get a separate output window. To re-execute the entire program, do *Edit—Select All* (or click on the  button) and hit `<Enter>` (or click on the run icon: ).

RATS will execute all of the instructions in the file in order, generating text output in the output window, along with the various graphs and report windows. That's how easy it is to reproduce a set of results! You can do the same thing with any of our example programs.

.PRG Extension

Historically, we used the extension `.PRG` for program files, but have phased that out because it was not unique to RATS, so any of a number of programs might actually end up being associated with it, and because it caused problems with some e-mail filters since one possible use of `.PRG` is for executable batch files. However, since there are a large number of existing RATS program files which were given `.PRG` extensions, we still include that as a possible extension in the file dialog.

1.5.8 Learn More: The PRINT Instruction

Another useful instruction for examining your data is **PRINT**. This displays series to the output window or (with a WINDOW option) to a spreadsheet-style window. If you want RATS to print all the data for all the series in memory, just type:

```
print
```

To view data over a limited range of entries, you can add *start* and *end* parameters with the desired dates (or entry numbers):

```
print 1990:1 1996:6
```

You can also print only certain series by listing them after the dates:

```
print 1990:1 1996:6 hs6fr esmooth esmooth_fore
```

If you want all the data but only for certain series, you can replace the *start* and *end* parameters with a forward slash (/). This tells RATS to use the default range for the series:

```
print / hs6fr esmooth esmooth_fore
```

To display the information to a spreadsheet style window, add a WINDOW option and a title for the window:

```
print(window="ESMOOTH Forecasts") / hs6fr esmooth_fore
```

You can copy and paste the contents of the resulting window into another program, or export the window contents to a spreadsheet or other file type using *File–Export...*.

PRINT is also very useful when you are troubleshooting a program. When you get unexpected results, such as an error message complaining about missing values, try printing out the series involved. You may notice an error that had slipped through earlier checks.

You can use the PICTURE option to supply picture codes (page Int–5) to control the formatting of the output. In this case, we use an * on the left of the decimal, which tells RATS to use as many digits as necessary, while the .# tells RATS to round the output to one decimal. For a full discussion, see Picture Codes in the help.

```
print(picture="*.#") / hs6fr

  ENTRY          HS6FR
 1986:01         105.4
 1986:02          95.4
 1986:03         145.0
 1986:04         175.8
```

1.5.9 Learn More: Data Transformations

SET (introduced on page Int–25) is the general data transformation instruction. RATS also has several special-purpose instructions for doing transformations, such as **SEASONAL** for creating seasonal dummies and **DIFFERENCE** and **FILTER** (page Int–44) for difference and quasi-difference transformations. However, most of the time you will be using **SET**.

Below, we describe **SET** in more detail, and present a kit of standard data transformations. Most of these examples are *not* based upon the data sets we have been using, so you won't necessarily be able to execute them as part of the tutorial. It should be easy to adapt them to your own needs, however.

We also describe the various *Wizards* available for doing many of these types of operations.

The Instruction SET

The general form of **SET** is

set(options) *series start end = function*(T)

In the *function* part of the **SET** instruction, you can use constants, scalar variables, other series, matrices, calls to built-in or user-defined functions, and any of the arithmetic and logical operators available in RATS. In its most basic form, **SET** defines one series as a simple transformation of another series. For example:

```
set loggdp = log(gdp)
```

sets each entry of the series LOGGDP to the log of the corresponding entry of GDP, using the built-in function LOG() (see page Int–41 for more on functions).

The *start* and *end* parameters are optional, and you can usually skip them when you are doing your initial transformations after **DATA**.

To set a series as a function of other variables, you use standard arithmetic notation:

```
set totalcon = durables + nondur + services
set scaled = resids/sigmasq
```

Trend Series

You use the variable T to create trend series and dummy variables based upon "time". T is equal to the number of the entry being set, where the entry you specified on **CALENDAR** is given number 1. For instance, the following creates (in order) linear, quadratic and exponential (5% growth) trend series:

```
set trend = t
set trendsq = t^2
set exptrend = exp(.05*t)
```

Seasonal Dummies

RATS provides a special instruction **SEASONAL** for creating seasonal dummies. You use it in one of two forms:

```
seasonal  seasons
seasonal(period=1948:2)  february 1948:1  2009:12
```

The first creates SEASONS as a dummy for the *last* period of the year (4th quarter or December). By using SEASONS and its leads in a regression, you can get a full set of dummies without having to define a separate one for each period. With monthly data, the "lag field" SEASONS{-10 TO 0} covers dummies for February (lag "–10", which is a 10-period lead) to December (lag 0).

The second creates FEBRUARY as a February dummy defined from 1948 to 2009. The PERIOD option specifies the first period to receive a value of 1 (February, 1948).

Other Dummies

Dummy variables are easy to construct using logical and relational operators (see page Int–41), since these operators return the values zero or one.

```
set dummy = t>=1972:3.and.t<=1974:3
set large = pop>5000
set female = .not.male
```

The first example creates a dummy series called DUMMY using a logical expression. It stores the value 1 in entries where the logical expression is true, and a 0 in the other entries. In this case, the expression is true between 1972:3 and 1974:3 inclusive, so DUMMY is 1 for those entries and 0 elsewhere. The second sets LARGE to 1 when the corresponding entries of POP are greater than 5000, and 0 elsewhere. In the third example, entries of FEMALE are 1 when MALE is 0, and are 0 elsewhere.

Trend/Seasonals/Dummies Wizard

You can create many of the trend, seasonal, and dummy variables described above using *Data/Graphics—Trend/Seasonals/Dummies*.

Lag and Lead Transformations

Transformations involving lags or leads of a series can be written using the T subscript, lag notation, or a combination of both.

```
set pavge = ( price + price{1} ) / 2.0
set pavge = ( price(t) + price(t-1) ) / 2.0
set inflation = 400.*log( deflator/deflator{1} )
```

The first two are identical. They create PAVGE as the average of the current and first lag values of PRICE. The first uses lag notation, the second uses T explicitly. Which style you use is a matter of taste. The third example computes annualized growth rates of DEFLATOR (in percent) using the log difference approximation.

All of these transformations involve a lagged value, so the first entry will be set to missing: if PRICE is defined over 1922:1 to 1941:1, PAVGE will be defined only over 1923:1 to 1941:1.

Differencing

Simple differencing is, of course, easy to handle with **SET**:

```
set diffgdp = gdp - gdp{1}
```

RATS also offers the instruction **DIFFERENCE** for regular, seasonal, or fractional differencing (see page Int–31 on the use of the /):

```
difference gdp / diffgdp
difference(sdiffs=1,diffs=1) gdp / ddsgdp
```

As noted earlier, you can also use *Data/Graphics—Transformations* or *Data/Graphics—Differencing* to create differenced series.

Growth Rates

There are several ways to compute growth rates or approximations to them. The first two **SET** instructions below compute (for quarterly data) annualized growth rates, the third and fourth compute period over period rates and the last computes the year over year growth.

```
set growx = 4.0*log(x/x{1})
set growx = ((x/x{1}) ^ 4 - 1.0)
set growx = log(x/x{1})
set growx = x/x{1} - 1.0
set growx = x/x{4} - 1.0
```

Data/Graphics—Transformations can also create simple growth rate series.

Benchmarking and Normalizing

These generally require one of the two options for **SET**; either FIRST or SCRATCH:

```
set(first=100.) capital 1920:1 1941:1 = .90*capital{1}+invest
```

computes CAPITAL from an investment series beginning with a value of 100.0 for the first period (1920).

```
set(scratch)  gdp  1950:1 1998:4 = 1000.0*gdp/gdp(1975:3)
```

renormalizes GDP to take the value 1000 in 1975:3. You need the SCRATCH option because it is replacing the values of the GDP series that are needed in the calculation, and is using two entries of it (current entry and 1975:3) . See **SET** for more on these options.

The %IF Function—Conditional Expressions

The logical function %IF(x,y,z) returns the value y if x is non-zero, and returns z if x is zero. This is a very useful function. For example:

```
set w = %if(test<0, %na, w)
set testseries = %if(t<=1990:12, seriesone, seriestwo)
```

The first makes all entries of W for which the series TEST is negative equal to missing (%NA is how you represent the missing value in expressions). The other entries are not affected. The second stores the values of SERIESONE in TESTSERIES for all entries through 1990:12. Entries from 1991:1 on are taken from SERIESTWO.

Note that the %IF function only evaluates a single–valued expression each time it is called. It is the **SET** instruction itself that "loops" over the entries in the sample range, calling %IF once for each entry in that range.

Missing Values

SET propagates missing values through the formula. With only two exceptions (the %VALID(x) function and the %IF(x,y,z) function), any operation which involves a missing value returns a missing value.

SET also sets to missing any observation which involves an illegal operation, such as divide by zero and square root of a negative number.

```
set stockprice = %if(%valid(stockprice),stockprice,-999.0)
```

This replaces all the missing values of STOCKPRICE with the value –999.00. You might use this in preparation for exporting this series to a file format (such as ASCII) which doesn't support special codes for missing values.

1.5.10 Learn More: Forecasting

What's Available?

RATS can forecast using a wide range of modelling techniques:

- Simple regression models (Section 1.5.4 and *User's Guide* Section 5.2)
- Simultaneous equations (*User's Guide*, Chapter 8)
- Vector autoregressions (VAR's) (*User's Guide*, Chapter 7)
- Exponential smoothing (Section 1.5.3 and *User's Guide* Sections 6.2 and 6.3)
- Box-Jenkins (ARIMA) models (*User's Guide*, Sections 6.2 and 6.4)
- Spectral methods (*User's Guide*, Sections 6.2 and 6.5)

Chapter 5 of the *User's Guide* provides a general overview of forecasting, while specific kinds of forecasting models are addressed in the sections noted above.

Wizards and Instructions

You can use the **PRJ** instruction (page Int–71) to compute out-of-sample fitted values for the most recently executed regression, as long as the necessary right-hand-side variables are defined for the required time periods.

We introduced the **UFORECAST** instruction and the *Single-Equation Forecast* Wizard on page Int–54. The other, more general, forecasting instruction is **FORECAST**. It can forecast one equation or many, and can also be used to forecast non-linear models. You can use *Time Series—VAR (Forecast/Analyze)* on the menu to generate a **FORECAST** instruction once you've defined a proper model.

By default, both **UFORECAST** and **FORECAST** do *dynamic* forecasting. That is, when computing multi-step forecasts, they use the forecasted values from initial steps as the lagged dependent variable values in computing the later forecast steps. Both can also do *static* forecasting if you use the STATIC option. Static forecasting means that actual values are used for any lagged dependent variable terms in the model.

These instructions are discussed in detail in Chapter 5 and 6 of the *User's Guide* and in the *Reference Manual*.

Equations and Formulas

Section 1.5.4 introduced the idea of defining equations and using **UFORECAST** to compute forecasts using equations. **FORECAST** can also produce forecasts for a formula, or a set of *formulas*. Note the distinction between equations and formulas:

Equations are descriptions of *linear* relationships. You can define them directly using the instruction **EQUATION**, but usually you create them using the DEFINE options of estimation instructions, as shown in Section 1.5.4. With **UFORECAST**, you use an EQUATION option to supply the equation you want to forecast. With **FORECAST**, you either list the equations you want to forecast on supplementary cards (one card per equation), or you use the MODEL option to specify a MODEL (group of equations and/or formulas) that you want to forecast. MODELS are usually constructed using the **GROUP** instruction.

Formulas (also called FRMLs) are descriptions of (possibly) *non-linear* relationships. These store a "**SET**" style function together with the dependent variable, and are normally defined using the **FRML** command. Before you can forecast a formula or set of formulas, you *must* group them into a model using the **GROUP** instruction. You then use the MODEL option on **FORECAST** to forecast the model. See page Int–84 and Chapter 8 of the *User's Guide* for more on formulas.

Simulations

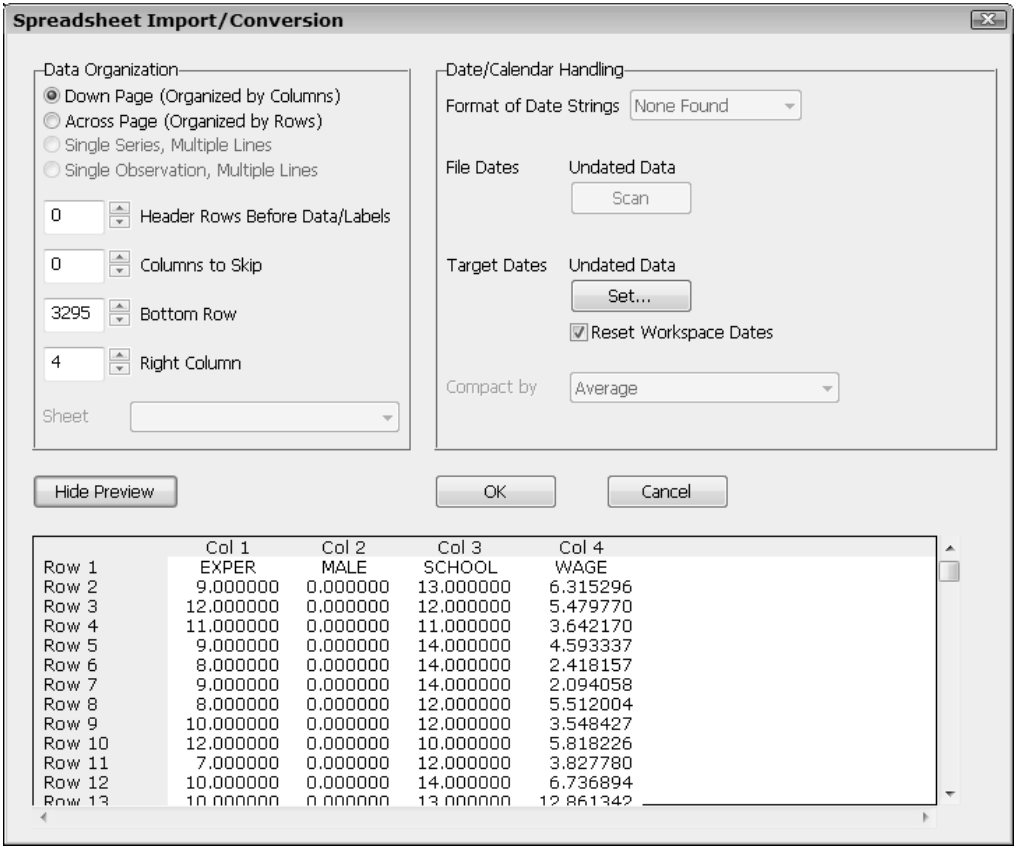
You can compute random simulations of a model using **UFORECAST**, or with the **SIMULATE** command, whose syntax is virtually identical to that of **FORECAST**. You can also use other tools to supply your own shocks to a system—see Chapter 16 of the *User's Guide*.

1.6 Example Five: Cross-Sectional Data

For our next example, we will look at a cross sectional data set. This data is taken from Chapter 2 of Verbeek (2008) and is provided on a text file called WAGES1.DAT. The file contains survey data for 3294 young workers, and includes four series: WAGE (hourly wage in dollars), MALE (a 1/0 dummy variable, equal to 1 for male respondents and 0 for female respondents), EXPER (years of experience at job), and SCHOOL (years of schooling). This example is provided on the file ExampleFive.rpf.

1.6.1 Reading the Data

To get started, close the windows from the previous session and do *File—New-Editor: Text Window* to create a new input window. When you are ready, select the *Data Wizard (Other Formats)* operation and open the WAGES1.DAT. (you may need to select "Text Files (*.*)" from the list of formats in order to see the file in the dialog box). Click on "Show Preview" to examine the data on the file:



	Col 1	Col 2	Col 3	Col 4
Row 1	EXPER	MALE	SCHOOL	WAGE
Row 2	9.000000	0.000000	13.000000	6.315296
Row 3	12.000000	0.000000	12.000000	5.479770
Row 4	11.000000	0.000000	11.000000	3.642170
Row 5	9.000000	0.000000	14.000000	4.593337
Row 6	8.000000	0.000000	14.000000	2.418157
Row 7	9.000000	0.000000	14.000000	2.094058
Row 8	8.000000	0.000000	12.000000	5.512004
Row 9	10.000000	0.000000	12.000000	3.548427
Row 10	12.000000	0.000000	10.000000	5.818226
Row 11	7.000000	0.000000	12.000000	3.827780
Row 12	10.000000	0.000000	14.000000	6.736894
Row 13	10.000000	0.000000	13.000000	12.861342

The data run down the page in columns, so the default settings should be correct. This file includes series names, so RATS won't need to ask you to provide names for each series.

1.6.2 The SMPL Option: Selecting Sub-Samples

Our first task is to examine the average wages for both males and females. As noted earlier, the series MALE is equal to 1 for males, and 0 for females. Recall from Section 1.4.2 that we can use the **STATISTICS** instruction to compute averages. But how to compute an average of WAGE using only certain observations (for males only, and for females only)?

The answer is to use the SMPL option (short for SaMPLe), which allows you to specify a sub-sample using a logical and/or numeric expression. The syntax is simply

smpl=*expression(t)*

Observations for which *expression(t)* returns a logical "true" or a non-zero numeric value are included in the operation. Observations where *expression(t)* returns a logical "false" or the number 0 are omitted from the operation.

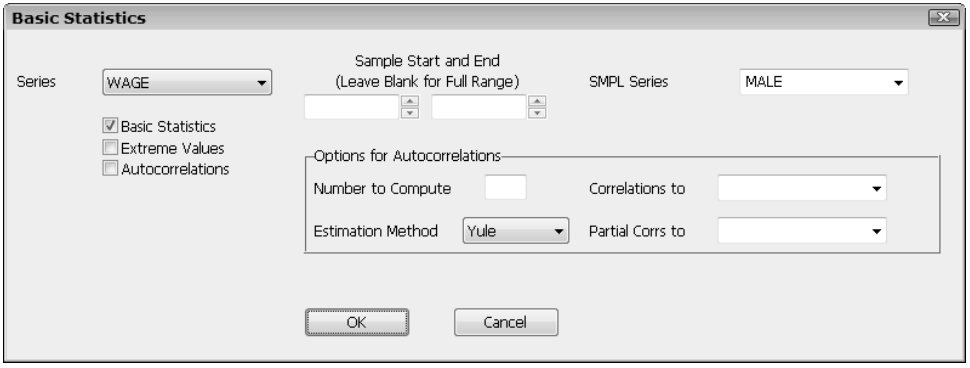
For example:

```
stats(smpl=male) wage
```

computes summary statistics for WAGE, using only observations where MALE is 1

```
Statistics on Series WAGE
Observations                 1725      Skipped/Missing            1569
Sample Mean              6.313021      Variance                12.242031
Standard Error           3.498861      SE of Sample Mean        0.084243
t-Statistic (Mean=0)    74.938512      Signif Level (Mean=0)    0.000000
Skewness                 1.921402      Signif Level (Sk=0)      0.000000
Kurtosis (excess)        8.845542      Signif Level (Ku=0)      0.000000
Jarque-Bera           6685.148147      Signif Level (JB=0)      0.000000
```

Using the *Univariate Statistics* operation, the dialog box would look like this:



Most instructions (and associated wizards) that deal with series have the SMPL option.

To get the averages for female workers, you just apply the logical operator .NOT. to the MALE series:

```
stats(smpl=.not.male) wage
```

This computes results using only observations where MALE is *not* equal to 1. This could also be written as:

```
stats(smpl=male<>1) wage
```

You could also create a series (using **SET** or the *Transformations* wizard) containing the reversed dummy variable and use that for the option. While probably not important here, defining the separate series can make it easier to see what's happening. If you create a FEMALE series using:

```
set female = .not.male
```

you can then use

```
stats(smpl=female) wage
```

If you are familiar with certain other statistics program, you might have expected that you would need an "IF" statement to accomplish this. While there is an **IF** instruction in RATS, it is used to evaluate a single condition, not as an implied loop over a range of observations.

1.6.3 Testing Regression Restrictions

Another way to examine the respective wages for males and females is to regress the WAGE series against a constant and the MALE dummy variable:

```
linreg wage
# constant male
```

The coefficient on the CONSTANT will be the average female wage, while the coefficient on the MALE variable will be the difference between male wages and female wages (that is, it reflects the impact of being male on wage). The results show an average female wage of $5.15, with males making an additional $1.17 per hour, for an average of $6.32, with a standard error of $0.11 on the differential.

To determine whether gender truly affects wages earned, Verbeek considers whether other factors, such as years of schooling and years of experience on the job, may explain some or all of the discrepancy. One way to do that is to compare the "restricted regression" used above (restricted because the schooling and experience variables are omitted) with the "unrestricted" regression which does include the additional variables to determine whether the additional components are significant.

RATS offers an instruction specifically for testing these types of exclusion restrictions. Let's look at the regression with some added factors:

```
linreg wage
# constant male school exper
```

You can simply write

```
exclude
# school exper
```

This produces the following:

```
Null Hypothesis : The Following Coefficients Are Zero
SCHOOL
EXPER
F(2,3290)=    191.24105 with Significance Level 0.00000000
```

The null hypothesis that the added coefficients on schooling and experience are zero (and thus can be excluded from the model) is tested with an F statistic with 2 and 3290 degrees of freedom. The result is significant beyond any doubt, so these variable *do* seem to help explain wages. However, the regression coefficient on MALE actually goes *up* in the expanded regression, so the use of the additional factors can't explain the difference between males and females.

The Regression Test Wizard

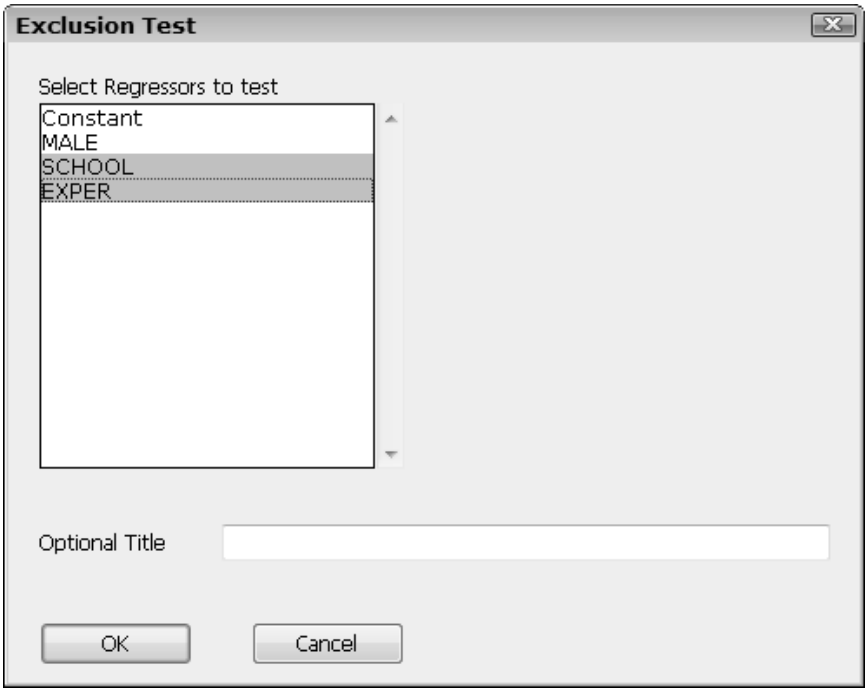
You can also generate this test using the *Statistics—Regression Tests* operation on the menu, which displays this dialog box:

Here, we want to select the "Exclusion Restrictions" option, which tests that some set of coefficients all have zero values. That's the simplest of the three options.

The "Other Constant Restrictions" is used when you have a test on one or more coefficients with a null hypothesis that they take specific values, not all of which are zero. (A common case, for instance, is that the intercept is zero and a slope coefficient is one).

"General Linear Restrictions" is used when you have restrictions that involve a linear combination of several coefficients (examples are that coefficients summing to one, or coefficients being equal, so their difference is zero).

When you click OK with the "Exclusion Restrictions" on, you get a second dialog box, listing the regressors from that last regression. Select (highlight) the SCHOOL and EXPER series:



Then click "OK" to run the test. This generates a **TEST** instruction with a ZEROS option. With **TEST**, you use numbers rather than variable names to indicate which terms are being tested. In this case, we are testing that the third and fourth regressors in the **LINREG** are (jointly) zero, so the instruction generated by the wizard is:

```
test(zeros)
# 3 4
```

More on Hypothesis Testing

Chapter 3 of the *User's Guide* provides full details on hypothesis testing, including detailed information on the instructions described above, as well as the specialized instructions **SUMMARIZE** and **RATIO**. You will also find out how to implement serial correlation and heteroscedasticity tests, Chow tests, Hausman tests, unit root tests, and many more.

1.6.4 X-Y Scatter Plots: The SCATTER Instruction

We've looked at creating time series (data vs. time) graphs with **GRAPH**. You can also graph one series against another to produce a "scatter" or "X–Y" plot.

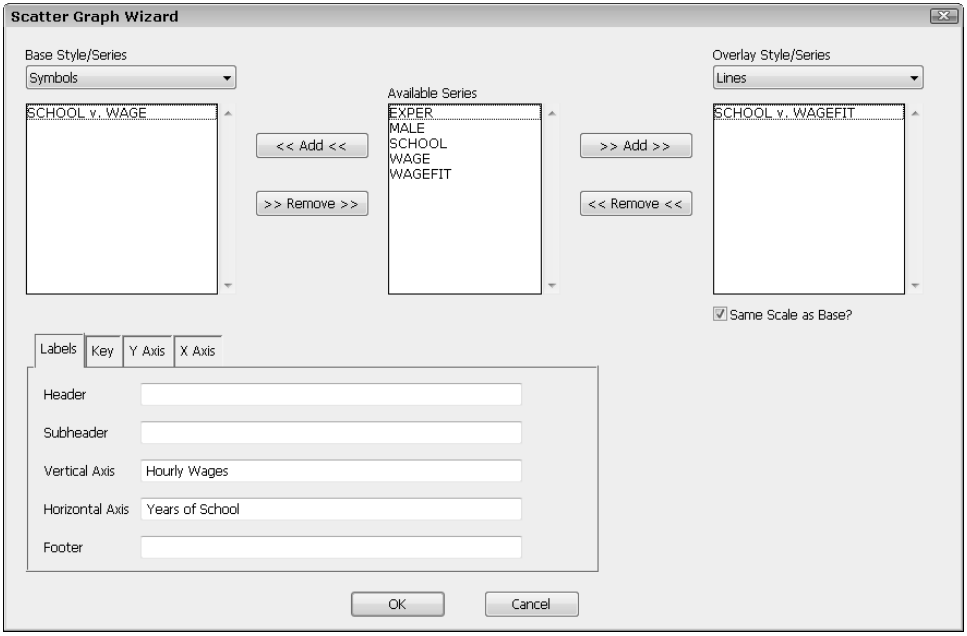
To demonstrate, let's look at the relationship between wages earned and years of schooling. First, we will regress WAGE against a constant and SCHOOL, and compute the fitted values from this regression. You can do that using the *Linear Regressions* wizard, adding a series name in the "Fitted Values to" field to save the fitted values, or you can type the instructions directly, using the instruction **PRJ** (short for project), which computes the fitted values from the most recent regression:

```
linreg wage
# constant school
prj wagefit
```

To draw the graph, select *Data/Graphics—Scatter (X–Y) Graph* to bring up the Scatter Graph wizard. Select (highlight) both WAGE and SCHOOL from the list of series and click on << Add << to add the pair to the list of series to be graphed. When prompted, choose the combination with SCHOOL as the X-axis variable and click "OK". Then, select "Symbols" as the "Style".

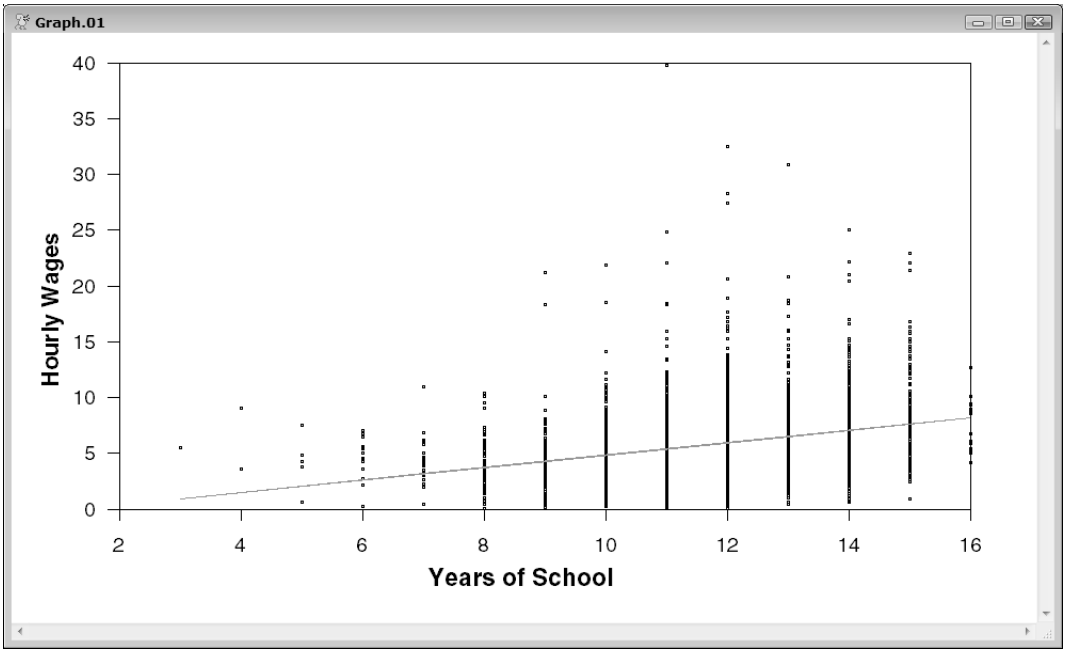
Now, select WAGEFIT and SCHOOL and click on the >> Add >> button to add this pair to "Overlay Series" list. SCHOOL should again be the X-axis variable. Select "Line" as the Style, and turn on the "Same Scale as Base?" switch (we want both pairs of series to be graphed using the same vertical scale).

In the dialog box below, we've also added labels for the horizontal and vertical axis:



Clicking "OK" generates the following **SCATTER** instruction, and the graph shown below:

```
scatter(style=symbols,overlay=lines,ovsame,$
  vlabel="Hourly Wages",hlabel="Years of School") 2
# school wage
# school wagefit
```



As you can see, the OVERLAY option allows us to combine (overlay) two different styles (possibly with two different scales) on a single graph.

1.6.5 Learn More: Comment Lines and Comment Blocks

If you open up ExampleFive.RPF, you will see, in addition to the instructions that we just generated, lines like the following:

```
* Adds school and exper to the regression and test the joint
* significance of the two additional variables.
```

These are comments that we added to explain what's going on to someone reading the example file, but not following it in this section. It's generally a good idea to add comments to any program file that you save.

Comment Lines

A line with * as the first non-blank character is treated as a comment and is ignored. We use comments rather extensively in the examples of this manual to help you understand the programs. However, note that we also often use marginal comments which are set off from the rest of the line by being in italic type rather than bold face. This is *not* legal in an actual RATS program.

Comment Blocks

You can also mark an entire block of lines as a comment by using the symbol /* to mark the beginning of the block, and */ to mark the end of the block. (Note: the /* must be the first (non-blank) characters on the lines). While this provides a convenient way to enter several lines of comments, it is probably most useful commenting out an existing block of instructions. For example:

```
* This is a comment line

/*
 This is a comment block
  linreg y
  # constant x1
*/

linreg y
# constant x1 x2
```

This skips the first regression, but does the second.

Tips on Using Comments

Your first goal is always to get the calculations correct. However, if you have a program (or part of one) that you expect that you might need again or that others might want to read or use, it's a good idea to add comments describing what each section of code does.

Also, if you have a part of your program which you're not sure is correct, commenting it can often help you spot errors. (If you can't explain why you're doing something, that might be a good sign that you're doing it wrong).

Now it's possible to overdo the comments. You may have seen some programs with a very elaborate comment block like:

```
**************************************************
* This is a program that analyzes the population    *
* of kangaroos in New South Wales.                  *
**************************************************
```

RATS provides several ways to help manage larger blocks of comments. The first are the comment block symbols. So

```
/**************************************************
This is a program that analyzes the population
of kangaroos in New South Wales.
**************************************************/
```

gives a similar appearance to the block above, but you can more easily edit the comments as you don't have to worry about having the * symbols at the start of each line.

The *Edit* menu provides operations for commenting or un-commenting sections of your code. If something not quite as pretty will do, you can use the *Edit—Format Comments* operation. This takes a selected block of text and creates a set of one or more comment lines of roughly equal length (except for the final line). If some of the lines are already comments, the *'s and leading blanks are stripped out before the lines get formatted. As an example, we pasted in the first two sentences from above as a single line and did *Format Comments*. The result is:

```
* If something not quite as pretty will do, you can
* use the Format Comments operation on the Edit
* menu. This takes a selected block of text and
* creates a set of one or more comment lines of
* roughly equal length (except for the final line).
```

The desired line length can be set on the Editor tab in the *Preferences* dialog. The comments above used a line length of 50, to fit the page, but this is usually set to something longer than that (70-80 tend to work best). You may want wider comments in your actual program files. Note that the comment length isn't a hard limit. In order to roughly balance the lines, it might go a bit above that.

There are two other "comment" operations on the *Edit* menu. *Comment-Out Lines* takes the selected text and inserts * at the beginning of each; converting each into a comment. The reverse is *Uncomment Lines*, which strips the lead * off any line which has one. These allow you to take a block of lines out of the execution and put them back in.

1.6.6 Learn More: Linear Regression Instructions

You'll notice that we've used only a few features off the Linear Regressions Wizard (page Int–28). We'll talk a bit here about what else is available.

Robust (HAC) Standard Errors

This is a checkbox which you can use to change the way that the regression standard errors are computed. It won't change the coefficients; just the standard errors, the covariance matrix, and any regression tests that you do afterwards. If you apply this to our last linear regression in this example, you will get the following instruction:

```
linreg(robust) wage
# constant school
```

so you could also simply have inserted the ROBUST option into the existing instruction. If you compare the output, you'll see a few differences. First, it now says:

```
Linear Regression - Estimation by Least Squares
With Heteroscedasticity-Consistent (Eicker-White) Standard Errors
```

so while it still estimates by least squares, the standard errors are now computed using the Eicker-White methods, which are "robust" to certain types of heteroscedasticity (hence the name of the option). If you aren't familiar with this, see Section 2.2 in the *User's Guide*. You also might notice that the regression *F* isn't shown: that's a test which is computed under the assumption of homoscedastic errors, and you've said (by using the ROBUST option) that that isn't true. The other summary statistics are the same, as are the coefficients, but the standard errors are (slightly) different, so the T-Stat and Signif columns change as well.

When you click on the "Robust (HAC) Standard Errors" check box, you will also notice that two additional fields below it become active: the "Lags/Bandwidth" and "Lag Window Type". These aren't of any special interest in this example, since this is cross section data, but these allow you to compute standard errors which are also robust to autocorrelation (the HAC is short for **H**eteroscedastic **A**uto **C**orrelated). If we applied that here (not that we should), the new instruction would be something like:

```
linreg(robust,lags=3,lwindow=newey) wage
# constant school
```

Residuals

Further down in the wizard, you will notice a drop down box labeled "Residuals To". Whenever, you run a regression, RATS creates a series named %RESIDS and fills it with the residuals. If you want to graph the residuals, or compute its statistics, you can just use %RESIDS like any other series. However, if you run a new regression, %RESIDS will now get the residuals from *that*. If you want to keep a copy of the residuals from one *specific* regression, you can give a (valid) series name in this box, which will generate the new instruction (see page Int–31 for a description of the /):

```
linreg wage / u
# constant school
```

Show Standard Output

This is a checkbox which you will almost always want to be on. If you click it off, you will generate an instruction like this, adding the NOPRINT option:

```
linreg(noprint) wage
# constant male
```

Almost all instructions which ordinarily would produce some output will have the option of NOPRINT. Why would you want to do that? Perhaps all you really want from the regression are the residuals. Or the defined equation for forecasting. When you run a regression, almost anything computed by the **LINREG** is available for further calculations. For instance, the coefficients are in a vector called %BETA; the sum of squared residuals are in %RSS. Perhaps you only need those. If you plan ahead and use NOPRINT, you don't have to read past the unwanted regression output.

Show VCV of Coefficients

This is a checkbox that you will usually want to be *off*. If it is on, the estimation instruction will include the VCV option (using the larger regression from the example):

```
linreg(vcv) wage
# constant male school exper
```

This places the following below the standard regression output (and puts a similar report into the *Window—Report Windows* list):

```
Covariance\Correlation Matrix of Coefficients
             Constant       MALE         SCHOOL       EXPER
Constant   0.216203148  -0.15560300  -0.90642459  -0.54623329
MALE      -0.007790538   0.011594097   0.09663502  -0.10932520
SCHOOL    -0.013822338   0.000341249  0.001075567   0.18093999
EXPER     -0.006035400 -0.000279728   0.000141010  0.000564669
```

This is a square table with covariances on and below the diagonal and correlations above the diagonal. For instance, the estimated variance of the coefficient on MALE (the number at the intersection of the MALE row and MALE column) is roughly .0116. The estimated correlation between the coefficients on MALE and EXPER is where the MALE row and the EXPER column meet (*above* the diagonal), roughly -0.109. The covariance of the same two coefficients is at the EXPER row and MALE column (-.00028). Note that the header says Covariance\Correlation. We use the \ to help you remember the structure of the matrix: covariance below and correlation above.

We recommend that you not use this on a regular basis simply because there is little useful information in it. The covariances *are* used in doing joint hypothesis tests as in Section 1.6.3, but you will be using instructions like **TEST** and **EXCLUDE** that do the computations for you, rather than using numbers out of this table.

The Method Popup Box

Everything we have done so far has used the (default) choice of "Least Squares" in the "Method" box at the top left. There are several other choices available, some of which generate **LINREG** with a different set of options; others generate different instructions with a similar overall look to **LINREG**.

The next two choices are "Two Stage Least Squares" and "GMM (IV with Optimal Weights)". If you choose one of those, you'll notice that the "Instructions" box and button become enabled. These are (closely related) instrumental variables estimation methods which require both a list of regressors and a list of instruments. (You can read more about instrumental variables in Section 2.5 of the *User's Guide*.) You add variables to the instruments list just as you do for the regressors: you can either type them directly into the box, or you can use the ... button to pop up a dialog box to manage the list. If we choose "Two Stage Least Squares", and use CONSTANT and SCHOOL as the Explanatory Variables and CONSTANT, EXPER and MALE as the Instruments, we generate the instructions:

```
instruments constant exper male
linreg(inst) wage
# constant school
```

The instruments are listed on the separate **INSTRUMENTS** instruction line. That allows you to do several **LINREG**'s off the same list without having to re-enter the instrument information. Two-stage least squares is done by adding the INST (short for INSTRUMENT) option to **LINREG**.

The next two choices for the "Method" box are "AR(1) Regression" and "AR(1)-Instrumental Variables". These will generate a different instruction: **AR1**, which is for linear regressions with first-order autocorrelated errors. See Section 2.4 of the *User's Guide* for more on handling autocorrelated errors. We won't look further at this here, but would like to point out that parts of the dialog change when you choose this (and all the remaining methods). In particular, the "Robust (HAC) Standard Errors" is replaced by a different popup box for the "AR1 Method".

The next choice down on the "Method" box is "Robust Regression", which generates an **RREG** (Robust REGression) instruction. This is still a linear regression, but uses a different estimation technique which is more "robust" to outliers. See Section 2.12 in the *User's Guide* if you're interested.

The final choice is "Stepwise Regression", which generates an **STWISE** instruction. Stepwise regression is nowhere near as popular as it was perhaps thirty years ago, but still has its place. If you're interested in this, see the description of **STWISE**.

Other Related Instructions

There are several other instructions which have similarities in form to **LINREG**, in the sense that the instruction looks like:

```
instruction(options)  dependent variable   start   end
# list of explanatory variables
```

but have option sets that are so different that they can't easily be fit into the same wizard. These are the next several choices on the *Statistics* menu:

The *Limited/Discrete Dependent Variables* wizard handles the **DDV** (Discrete Dependent Variables) and **LDV** (Limited Dependent Variables) instructions. **DDV** is used for techniques applied to data where the dependent variable is "discrete" (usually just values 0 and 1) such as logit and probit. **LDV** is used for techniques such as tobit which are applied when the dependent variable is continuous, but the observed range is somehow limited. These are covered in Chapter 12 of the *User's Guide*.

The *Panel Data Regressions* wizard generates a **PREGRESS** instruction, which includes special techniques for estimating the linear regression with panel data (often also known as longitudinal or cross section-time series data); techniques such as fixed effects and random effects. See Section 12.5 in the *User's Guide* for more.

The *Recursive Least Squares* wizard generates an **RLS** instruction. When applied over the same range as a **LINREG**, it will give exactly the same output—the point is that it does a whole set of calculations for reduced ranges that can be used to check for stability in the overall linear model. See Section 2.6 of the *User's Guide*.

There are quite a few other estimation instructions, but they have quite a different form. The instructions included here have models which are based upon a linear combination of a set of explanatory variables. If we need a non-linear function, we will need a different way to enter that than just a list of variables. We'll see one of these other instructions (**NLLS**, for non-linear least squares) in Section 1.7.

There are several other instructions which apply to systems of linear equations rather than just single equations: **SUR** (Section 2.7 of the *User's Guide*), which does fairly general systems, and **ESTIMATE**, which estimates the equations of a Vector AutoRegression (VAR)—the subject of Chapter 7 in the *User's Guide*.

Regression Format

"Regression format" is the manner which RATS uses to list regressors, either on the supplementary card for **LINREG** and related instructions, or on a list of instruments. It consists of a list of series and *lag fields*. You create a lag field by appending to the series name the list of desired lags in braces (**{}**). You can use the notation `{n1 TO n2}` to represent all lags between `n1` and `n2`. Leads are represented as negative lags. Examples of lag fields (using data from an earlier example with time series data) are:

M1{0 TO 4 8}	**RATES{1}**	**SEASON{-1}**
`M1` lags 0,1,2,3,4, and 8	`RATES` lagged once	One period lead of `SEASON`

1.6.7 Learn More: Error Messages

In a perfect world, we wouldn't need to talk about errors or error messages. Unfortunately, you will undoubtedly encounter an occasional error message in your work with RATS. In this section, we discuss how to interpret error messages and fix errors.

RATS Errors and Error Messages

Whenever RATS detects an error, it displays an error message describing the problem. Where possible, these messages provide you with specific information about the error.

Each error message begins with an *error code*: a one-, two-, or three-letter code indicating the general type of error, plus a numeric code identifying the specific error. For example, general syntax errors begin with the code "SX", syntax errors involving options have the code "OP" (for OPtion), and errors involving linear regressions begin with the code "REG".

There's a complete listing of RATS error and warning messages in the help. This includes some suggested causes and remedies.

Syntax Errors

Syntax errors probably appear most often. These occur while RATS is *reading* a line (as opposed to errors which occur as RATS is trying to *execute* a line it has read successfully). Syntax errors generally mean that RATS couldn't make sense of the line it was processing. Missing parentheses, misspelled instruction or option names, illegal characters, and incorrectly entered numbers are some common causes of syntax errors.

RATS displays two lines of information: the error message, and then the portion of the line which caused the error, enclosed in >>>> and <<<< symbols. RATS only prints the line *up to the point at which the error occurred*. It will display a maximum of 20 characters, counting back from the point of the error.

For example, suppose you tried the following:

```
set trendsq 1 50=t^2
```

This should have at least one blank space between the "*end*" parameter (50) and the =. Because that space is missing, RATS will produce the following error message:

```
Error SX15. Trying to Store Into Constant or Expression
>>>>set trendsq 1 50=t<<<<
```

As you can see, RATS stopped processing the line immediately after it detected the illegal assignment operation.

RATS often needs to look one character ahead to identify particular operators. For instance, if it sees an =, it must check the next character to see if the operator is ==. Thus, the true culprit may be one character before the one shown.

If the reason for a syntax error isn't immediately obvious, *check the entire line*, not just the point at which RATS stopped reading the line. RATS can often interpret the actual mistake in a way that allows it to go on reading a few more characters from the line.

For instance,

```
data (unit=input)  / gnpdata
```

produces the error

```
Error SX11. Identifier INPUT Is Not Recognizable
>>>>data (unit=input)<<<<
```

RATS reads all the way to the right parenthesis before it generates the error, but the problem is actually the (illegal) blank space between **DATA** and the left parenthesis that starts the option field. This blank causes RATS to interpret `(UNIT=INPUT)` as the *start* parameter, rather than as the option field. It is legal to use an expression like this for *start*. However, `INPUT` is not a recognizable variable name, so RATS produces the error message.

Other Types of Errors

Once RATS has successfully interpreted the line, it does not echo back the characters on the line. All you will see is the error message.

Show Last Error

If you are just typing in instructions one at a time, it will be clear which instruction caused the problem. Find it, fix it and re-execute the corrected line (hit <Enter>). However, if you select a block of lines and execute, the problem could be with any of the selected lines. If it's a syntax error, you might be able to figure out where the problem is by matching up the echoed characters, but that won't help if it's an execution error. In that case, try the *Edit—Show Last Error* operation, which will position the cursor on the instruction line that generated the error.

1.6.8 Learn More: Entry Ranges

The SMPL Instruction

The **SMPL** *instruction* lets you control the default range. It is an important instruction in EViews/TSP–like programs, but is less so in RATS because:

- You can set explicit entry ranges on individual instructions where necessary.
- You can use default ranges on most transformation and regression instructions.

SMPL *is* useful in situations where you want to run a sequence of regressions, forecasts, or other operations over a common fixed interval (other than the default range).

For instance, suppose you have data from `1922:1` through `1941:1`, but you want to run two regressions over the period `1923:1` to `1935:1`.

```
smpl 1923:1 1935:1
linreg foodcons
# constant dispinc trend
linreg foodprod
# constant avgprices
```

Once you set a **SMPL**, RATS uses the **SMPL** range as the default. To clear a **SMPL**, just issue a **SMPL** instruction with no parameters. We recommend that you do any preliminary transformations before you set a **SMPL**.

If you need to skip entries in the *middle* of a data set, use the `SMPL` *option* (pages Int–67 and Int–82).

Using Entry Numbers Instead of Dates

You can use hard-coded entry numbers rather than dates. For example, given a **CALENDAR** of

```
calendar(m) 1959:1
```

the instruction

```
linreg rate 1 24
# constant ip m1diff ppisum
```

is equivalent to:

```
linreg rate 1959:1 1960:12
# constant ip m1diff ppisum
```

because `1960:12` is the twenty-fourth entry given the **CALENDAR** setting. With time series data, you will usually use the dates, but using entry numbers is sometimes easier.

Dates are actually handled as integer-valued variables, which means that you can combine dates and integer entry numbers in an expression. For example, another way to do the regression above is:

```
compute start = 1959:1
compute end = start+23
linreg rate start end
# constant ip m1diff ppisum
```

This means that you need to be careful about using the proper format for dates when using them. Suppose accidentally you left off the `:1`, and wrote:

```
linreg rate 1959 1960
# constant ip m1diff ppisum
```

RATS would try to run the regressions using entries one thousand nine hundred and fifty nine and one thousand nine hundred and sixty. Clearly not what you intended, so be sure to include the "`:period`" anytime you are referring to a date.

Exceeding the Default End Period

The default end period is not a binding constraint—you can define series beyond this default limit as needed by using explicit date/entry ranges.

Selecting Subsamples: The SMPL Option

RATS provides two related methods for dropping observations out of the middle of a sample. We introduced the `SMPL` option available on many instructions on page Int–67. It allows you to include and omit selected observations within the *`start`* to *`end`* range of the instruction. The formal description is:

`smpl=`*`SMPL series or formula`*

The *`SMPL series`* is a series or formula with non-zero (or logical "true") values at the entries (between *`start`* and *`end`*) you want to include in the estimation, and zero values (false) at the entries to omit. It can be an existing series, or a formula like that used in a **`SET`** instruction. It's usually a dummy variable series of some form. It may be a dummy in the data set, or one constructed for this particular purpose.

You can also set a `SMPL` series which applies to *all* instructions that support the `SMPL` option. You do this with

`smpl(series=`*`SMPL series`***`)`**

We prefer to use the `SMPL` options in most situations. If you use the **`SMPL`** *instruction*, you must remember to reset it when you are done with the analysis which uses the reduced sample.

Missing Values

RATS will automatically omit observations which have a missing value in the dependent variable or *any* of the regressors, so you don't need a SMPL option for them. If you have a regression with lags, you could end up losing several data points since any observation which needs an unavailable lag of a variable will be dropped.

Skipping a Time Period

Suppose you want to leave the years 1942 to 1946 out of a regression over 1919 to 2009. You could construct a *SMPL series* for this with:

```
set notwar  1919:1  2009:1  = 1.0        Set all entries to one
set notwar  1942:1  1946:1  = 0.0        Set entries for war years to zero
linreg(smpl=notwar)  depvar
# regressors
```

The second **SET** only applies to the restricted range from 1942:1 to 1946:1, and so won't affect the values at other time periods. You could do the same thing with

```
linreg(smpl=t<1942:1.or.t>1946:1) depvar
# regressors
```

but the first way is easier to understand at a glance.

Subsample Based Upon Value

To run a regression for just those entries where a series exceeds some value or meets some similar criterion, use relational operators. This generates dummies for three subsamples based upon the series POP, then runs the regressions over those.

```
set small  = pop<2000
set medium = pop>=2000.and.pop<=6000
set large  = pop>6000
linreg(smpl=small)  depvar
# regressors
linreg(smpl=medium) depvar
# regressors
linreg(smpl=large) depvar
# regressors
```

Again, we could collapse these by using the SMPL option into three instructions like

```
linreg(smpl=pop<2000)  depvar
# regressors
```

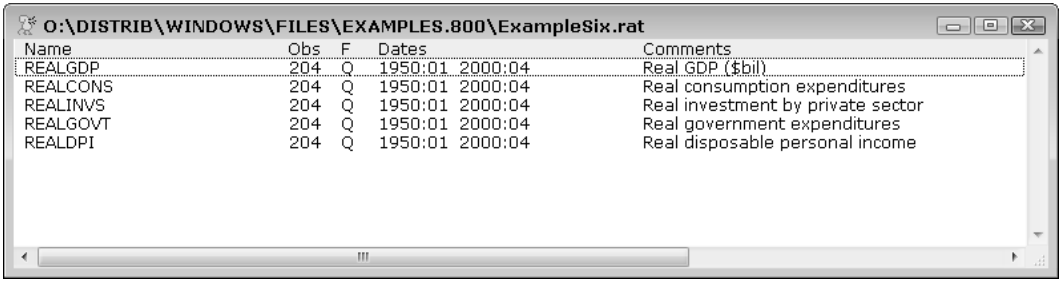
again, however, at a loss in readability. A few extra keystrokes is generally worth it if is produces a clearer program.

1.7 Example Six: Non-Linear Estimation

Our final example (file `ExampleSix.RPF`) is taken from Greene (2012), Section 7.2.5. This estimates a model of consumption of the form:

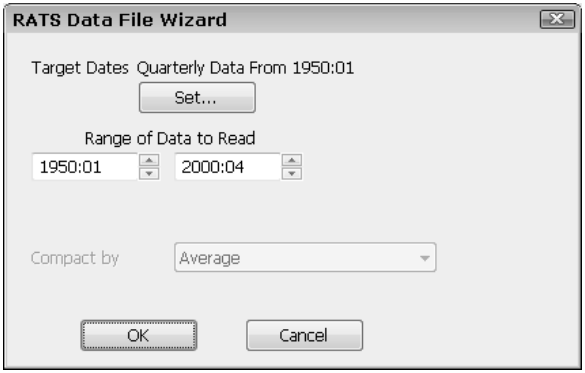
$$C_t = \alpha + \beta Y_t^{\gamma} + \varepsilon_t \quad (6)$$

where C is (real) consumption, Y is (real) disposable personal income and α, β and γ are unknown parameters. The data file is `ExampleSix.RAT`, which is a RATS data format file. If you open this (choose *File—Open*, pick "RATS Data Files" in the file type box, and select the file), you will see the following RATS Data File Window:



If you think that it looks a lot like a Series Window, you're right. They are formatted the same way, and share most of the same operations on the *View* menu and toolbars.

Of the five series on the file, we need only two: `REALCONS` and `REALDPI`. Select those two (click on one of the series, then `<Ctrl>+click` or `<Command>+click` on the second), and choose the *Data/Graphics—Data (RATS Format)* menu operation. This will pop up the following dialog box:



You'll notice that this is *much* simpler than the Data Wizard for the other formats. What this wizard will do is to look at the series that you selected, and guess that you want the coarsest frequency and the maximum common range. For instance, if we had both monthly and quarterly data, with some series starting in 1947 and some

in 1954, the guess would be quarterly data starting in 1954. If the common range is what you want, you can just OK the dialog. In this case, all the data are quarterly, and start and end at the same dates, so the only question is whether we want the whole range. If we wanted just 1960:1 to 1995:4, we could change the start and end dates in the dialog to request that range. We do want the full range, so we get

```
open data examplesix.rat
calendar(q) 1950:1
data(format=rats) 1950:01 2000:04 realcons realdpi
```

If γ were known, we could create a transformed variable and estimate α and β using **LINREG**; for instance, if it were 1.2, that would be done with:

```
set ypower = realdpi^1.2
linreg realcons
# constant ypower
```

Since it's unknown, we can't, and have to estimate this using the instruction **NLLS** (Non-Linear Least Squares).

Estimating linear models with least squares and related techniques is a fairly simple process, involving straightforward matrix computations. Non-linear estimation, however, is an inherently more general and complex task—the variety of different models you can estimate is virtually limitless, and fitting these models requires the application of complex iterative optimization algorithms, rather than simple computations.

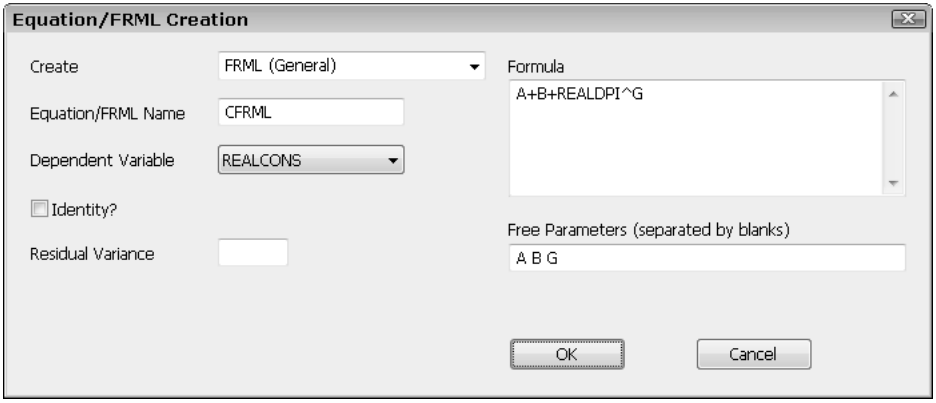
The estimation process can demand a fair bit of expertise and judgment on the part of the user, because the algorithms can be sensitive to initial conditions and are prone to converging to local, rather than global, optima. Thus, it should be no surprise that fitting non-linear models in RATS requires more effort and attention than fitting simple OLS models.

There are several steps we must do before we can even estimate the model. First off, we need to decide what we will call the free parameters; we can't use actual Greek characters, but we have to choose instead some legal RATS variable names.

Some other programs force you to use a specific way of writing this (perhaps `c(1)+c(2)*realdpi^c(3)`). RATS allows you to use symbolic names, which is much simpler, particularly if you ever decide to change the formula and add or remove some of the parameters. `alpha`, `beta` and `gamma` are an obvious choice here, but we'll choose the shorter `a`, `b` and `g`. If we were doing a **SET** instruction to calculate the right-hand side formula, we could write that as `a+b*realdpi^g`, which is, in fact, the way that we will write this.

We need two instructions to tell RATS what the free parameters will be called and to define the right-hand side formula. These, respectively, are **NONLIN** and **FRML**. You might find it easiest to put these in manually, but you can also apply the *Statistics—Equation/FRML Definition* Wizard:

Equation/FRML Definition Wizard



Choose "FRML (General)" from the "Create" popup menu. Choose the name that you want to assign to this formula in the "Equation/FRML Name" box; we'll use CFRML. Choose the dependent variable (REALCONS) in the "Dependent Variable" popup. Put in the names A B G in the "Free Parameters" box and the formula A+B*REALDPI^G in the "Formula" box. As you can see, since you have to type in both the formula and the parameter names, there isn't all that much that the wizard really does other than help you make sure you get all the information in. For instance, if you just put in the formula definition without providing the A B G names, you will get an error message that A is unrecognizable when you try to OK the dialog.

If you put everything in correctly, the wizard will produce:

```
nonlin a b g
frml cfrml realcons = a+b*realdpi^g
```

The **NONLIN** instruction tells RATS that on non-linear estimation instructions that follow, the free parameters will be A, B and G. The **FRML** instruction (FRML short for FoRMuLa) defines the expression that we'll be using—CFRML will be a special data type also known as a FRML. You don't *need* to include the dependent variable when defining a FRML; if you don't have one, just leave that part of the instruction out.

Guess Values

However, we're still not quite ready. (As we said, non-linear estimation is quite a bit harder than linear). Non-linear least squares is an iterative process; Greene, in fact, shows how to do (roughly) what RATS will do internally as a whole sequence of linear regressions. That sequence has to start somewhere, at what are known as the *guess values* or *initial values*. The default for RATS is all zeros, which sometimes works fine. Zeros *won't* work here, because if B is zero, the value of G doesn't matter: G is said to be *unidentified* (more than one value for G, here all values, give the same result).

An obvious choice to start are least squares results when G is one, which would just be a **LINREG** of REALCONS on CONSTANT and REALDPI. We can set this up with

```
linreg realcons
# constant realdpi
compute a=%beta(1),b=%beta(2),g=1.0
```

%BETA is a vector defined by RATS which has the coefficients from the last estimation instruction. %BETA(1) is the first coefficient and %BETA(2) is the second. We use those in a **COMPUTE** instruction to put our guess values into the three parameters.

The NLLS Instruction

Finally, we're ready. The model is estimated with

```
nlls(frml=cfrml)
```

Because of all the work required to prepare everything, the instruction itself is quite simple. **NLLS** has quite a few options, but most of them are to control the estimation process itself if the model proves hard to fit. The output is shown here:

```
Nonlinear Least Squares - Estimation by Gauss-Newton
Convergence in    65 Iterations. Final criterion was  0.0000003 <=  0.0000100
Dependent Variable REALCONS
Quarterly Data From 1950:01 To 2000:04
Usable Observations                        204
Degrees of Freedom                         201
Centered R^2                          0.9988339
R-Bar^2                               0.9988223
Uncentered R^2                        0.9997776
Mean of Dependent Variable         2999.4357843
Std Error of Dependent Variable    1459.7066917
Standard Error of Estimate           50.0945979
Sum of Squared Residuals            504403.21571
Regression F(2,201)                  86081.2782
Significance Level of F              0.0000000
Log Likelihood                       -1086.3906
Durbin-Watson Statistic                 0.2960

    Variable                       Coeff      Std Error      T-Stat      Signif
********************************************************************************
1.  A                         458.79905447  22.50138682     20.38981  0.00000000
2.  B                           0.10085209   0.01091037      9.24369  0.00000000
3.  G                           1.24482749   0.01205479    103.26415  0.00000000
```

It's not that much different from **LINREG** output. The only addition is the line describing the number of iterations. You would like this to indicate that the estimation has converged. If it doesn't, you will see in its place something like:

```
NO CONVERGENCE IN 30 ITERATIONS
LAST CRITERION WAS  0.0273883
```

When you get this message, or something like it, pay attention; it's telling you that something seems to be wrong. We get quite a few technical support questions where people are trying to interpret results (the bottom part of the output) while ignoring the message that the estimates might be wrong. This turns out to be a simple one to correct; we generated the warning message above by adding to **NLLS** the option ITERS=30 (which is less than the default of 100 iterations). Just taking off the restricted number of iterations (by omitting the ITERS option) gives us the proper result. In many cases, though, you have quite a bit more work to do to correct the reported problem. If you do non-linear estimation, you need to read (carefully) through the first few sections in Chapter 4 of the *User's Guide*.

Fitted Values

We introduced the **PRJ** instruction for computing fitted values on page Int–71. **PRJ** also works for many other types of estimations. However, it doesn't work after **NLLS**, or other instructions that aren't based upon a linear specification. Instead, you can apply your defined FRML in a **SET** instruction:

```
set fitted = cfrml
```

A FRML can be used within the **SET** expression (or, for that matter, in another FRML definition) almost exactly like a series, that is, CFRML by itself means the value at the current entry, CFRML{1} means the value the previous entry, etc. However, these types of expressions are the only places where you can use a FRML like a series. You can't use them for the dependent variable in a regression, or in the regressor list, or many other places. If you want to see the value produced by a FRML, do a **SET** as shown, and then examine the series created.

1.7.1 Learn More: Non-Linear Estimation Instructions

Many of the most important (and complicated) instructions in RATS do non-linear estimation. See Chapter 4 of the *User's Guide* on non-linear estimation techniques, which describes the various optimization algorithms, methods of creating and maintaining non-linear parameter sets and formulas, and includes descriptions of several of the basic instructions.

The following are similar to **NLLS** in that they require a **NONLIN** instruction and one (or more) FRML's to be defined before you can use them. Both are covered in Chapter 4 of the *User's Guide.*

MAXIMIZE does maximum-likelihood estimation, applicable to a very large variety of models.

NLSYSTEM estimates *systems* of non-linear equations, rather than just the one equation that **NLLS** handles.

The remaining instructions have either entire chapters or at least sizable sections in the *User's Guide* (all references are to that book) devoted to their use.

GARCH (Chapter 9) estimates univariate and multivariate ARCH and GARCH models.

DLM (Chapter 10) estimates and analyzes state-space models.

CVMODEL (Section 7.5) estimates covariance matrix models for structural VAR's.

NPREG (Section 13.1) estimates non-parametric regressions.

LQPROG (Section 13.2) solves linear and quadratic programming problems.

NNLEARN and **NNTEST** (Section 13.3) fit neural networks.

FIND (Section 4.13) can be used for just about any other type of optimization problem. If the function to be optimized can be written out using RATS instructions, **FIND** can be used to optimize it.

1.7.2 Learn More: Working with Matrices

Matrices (or arrays—we use the terms interchangeably) are very useful in RATS. For example, on page Int–87 we showed how to use the %BETA vector to access estimated coefficients. %BETA is just one of many arrays defined by various RATS instructions. For example, **LINREG** and many others also store the estimated standard errors and *t*-statistics in the vectors %STDERRS and %TSTATS, respectively. Arrays like these can be useful in reporting results, and for doing further computations, such as the Hausman test (*User's Guide*, page UG–95).

DISPLAY and COMPUTE

By now you should be familiar with **DISPLAY** and **COMPUTE**, and won't be surprised to learn they work with arrays as well as scalars. For example, after doing the **NLLS** instruction in our example, you could (re) display the estimated coefficients and standard errors (with fewer decimals) by doing:

```
display *.### %beta
display *.### %stderrs
```

COMPUTE is the primary instruction for doing matrix computations. RATS supports the standard arithmetic operators as well as several special matrix-specific operators. The program also provides dozens of built-in functions (such as inverse, transpose, and so on) for doing matrix calculations. See Section 1.6 of the *User's Guide* for more.

Defining Arrays

You can use **DECLARE** to create your own arrays, although you can often skip the declaration step. **COMPUTE**, **READ**, or **INPUT** can store values into arrays. If you need to construct an array from a set of series, you can do that using the **MAKE** instruction.

Other Types of Matrices

You can define arrays of labels, strings, series, equations, even arrays of arrays, and so on. These structures can be very useful in writing more sophisticated code, particularly for implementing complex, repetitive tasks. In Chapter 3 we make extensive use of arrays of strings for labeling graphs, while instructions like **FORECAST** use arrays of series to store results.

In-Line Matrix Notation

You can use "in-line" matrix"notation (*User's Guide*, Section 1.7) to provide the values of an array in an expression. Use two vertical lines (||) to mark the start and end of an array, and a single bar (|) for a row break. This is particularly useful for defining a matrix in the middle of an expression, such as in a non-linear formula. You can also use in-line matrix notation for options that accept an array for the arguments. For example, this estimates a Box-Jenkins model with AR terms on lags 1, 4, and 12:

```
boxjenk(ar=||1,4,12||) y
```

1.8 The RATS Programming Language

You can do quite a bit with the combination of the built-in instructions that we've introduced here, like **LINREG** and **NLLS**, plus the many others like **BOXJENK**, **GARCH** and **MAXIMIZE** that we've only mentioned in passing. But the real power of RATS comes from combining the sophisticated and carefully tested calculations of instructions like those with an extensive programming language. That's what makes possible everything from the hundreds of procedures that are included with RATS (Section 1.4.5), to the highly interactive cointegration program CATS, which is entirely written in the RATS language.

While you may never get to the point of writing your own procedures, you will almost certainly need to use some of the more basic programming features. The **DO** and **DOFOR** loops are relatively simple, and make many tasks much easier. Most of the early chapters in the *User's Guide* assume that you can use these, so we'll introduce them here. See Chapter 15 of the *User's Guide* for more on the programming features.

Loops

RATS offers five different looping instructions:

DO	is a standard loop instruction which loops over an integer index. It is like a FORTRAN DO or a BASIC FOR.
DOFOR	is less standard but often very useful. It loops over a list of items: series, matrices, numbers, etc.
WHILE	loops as long as an expression is true.
UNTIL	loops until an expression is true. It always executes the enclosed instructions at least once (**WHILE** may not).
LOOP	loops unconditionally. You need to use a **BREAK** instruction to get out of the loop.

A **DO** loop has the following structure:

```
do index = startvalue,endvalue,increment
  instructions executed for each value taken by index
end do
```

The index is a simple variable name, by convention, short names like I, J and K are often used. The *increment* is 1 by default, and the ,1 can be left out if it is.

A **DOFOR** loop repeatedly executes all of the instructions down to the matching **END**, with *index* taking a new value from the *list of values* on each pass through the loop. This continues until the list is exhausted. While the *index* of a **DO** loop must be an integer variable, a **DOFOR** *index* can be almost any type of variable.

```
dofor index =  list of values
   instructions executed for each value taken by index
end dofor
```

Examples

```
do time=1990:1,1992:12
   linreg y time time+47
   # constant y{1}
   set c1 time time = %beta(1)
   set c2 time time = %beta(2)
end do
```

This does a "rolling regression", using 48 data points for each regression. The first regression starts in 1990:1, the second in 1990:2, and so on. Dates in RATS are handled as integer entry numbers, so you can use dates in integer computations and in **DO** loops. On each trip through the loop, the **SET** instructions copy estimated coefficient values from %BETA (set by **LINREG**) into entry TIME of the series C1 and C2. When the loop is complete, these series will contain all 48 pairs of coefficient estimates.

```
dofor i = realgnp to realbfi
  set(scratch) i = log(i{0}/i{1})
end dofor
```

This does a percent change calculation for series from REALGNP to REALBFI. We use I{0} rather than just I so that I is treated as a series, not an integer value.

Conditional Execution

The **IF** instruction allows you to control the flow of program execution based on the condition of a logical expression. For example, in the following, "Condition is true" is displayed only if the variable TEST contains the value 1.

```
if test==1
   display "Condition is true"
else
   display "Condition is false"
```

Important note: The **IF** instruction only evaluates a *single* conditional expression—it does *not* include any sort of implicit looping over series or matrix entries. If you want to set values of series or an array based on a condition, you probably want to use the %IF() function, rather than using **IF** inside a loop. See page Int–63 for details.

IF-ELSE blocks are usually part of a larger programming structure, like a **DO** loop or a **PROCEDURE**. On the rare occasion that you need a stand-alone **IF** or **IF-ELSE**, enclose the whole set of lines in { }; for instance,

```
{
if test==1
   display "Condition is true"
else
   display "Condition is false"
}
```

This lets RATS know the limits of the programming code (what we call a *compiled section*) that needs to be treated as a unit.

1.9 What Next?

There is almost no limit to what you can do with RATS. It is only a question of learning how. You may need more information on the instructions we've already covered. Or, you may be interested in other types of analysis. With that in mind, we would like to offer some suggestions on how to find out what you need to know.

Yes, there *are* a lot of instructions, and the manuals *are* large. Don't worry! RATS is a very powerful and general program, and you may only need to use a fraction of its capabilities. Some suggestions for learning about RATS:

- First, look through the rest of this *Introduction*. You'll learn more about data handling and graphics, two things you will likely be using almost any time you work with the program.
- Take a look at the *Reference Manual* to get a feel for how it is organized and how details on each instruction are presented. (The on-line Help for instructions is formatted in a similar fashion). Look up some instructions you've already seen, like **LINREG**. When you see something new in an example, use the *Reference* or the help to find out exactly what is going on.
- Check out the *User's Guide* for in-depth information on various types of analysis. You can often jump straight to the topic of interest; if any of the previous chapters are needed to understand what's happening, we'll let you know that right away.
- Use the Tables of Contents and the Index (there's a combined index at the end of the *User's Guide*)! They are probably the quickest way to find the information you want. If yuou
- If you want to use a search engine, it usually helps to include "RATS" as one of the search keywords. If there's a RATS program that fits with the rest of the description, that will usually be at the top of the search list.
- There are two main lists of examples on the help. "Examples" are the main set that we use to demonstrate techniques. "Paper Replications" are more specifically for replications of specific papers.
- The "Index to the RATS instructions" has links to groups of RATS instructions by topic.

However, we can't emphasize enough that even if you find an example program that does exactly what you (think you) want, do not skip over the most important step which is to *Check Your Data*! In empirical work it's often the case that it takes longer to do something quickly than to do something carefully. We've had users waste countless hours trying to fit and interpret existing models adapted to their data set. The changes to the program were fine—it was the data set that was wrong. Make sure that you run the original program with the original data, and see whether your data have at least (somewhat) similar properties to the original.

2. Dealing With Data

Almost every time that you use RATS, you will need to bring data into the program. Since collecting and organizing your data set may be the single most time-consuming part of your analysis, you would obviously like RATS to accept your data with as little fuss as possible. We have provided you with a wide range of options for handling this important step.

You should note that RATS is not a "data set-oriented" program. Simply selecting a data file does not make the contents immediately available for analysis: you must use the **DATA** instruction or a Data Wizard to bring the data into working memory. This may seem to be an unnecessary extra step, but it offers numerous advantages. For example, you have the option of reading in only certain series and certain entries from the data set. Converting data from one frequency to another can be handled automatically as you read in the data.

RATS supports reading and writing data from a wide range of sources and in many different formats. The most popular are RATS format and Excel spreadsheets. The Professional version of RATS supports SQL commands and ODBC connections for reading many types of databases.

2.1 The Tools

Below, we discuss the instructions you can use to read data from a file into a RATS session and to write data from RATS out to a file. If you are new to RATS, please see the tutorial in Chapter 1 for additional information.

Getting Data In: The DATA Instruction and Data Wizards

Most of your projects in RATS will require reading data in from a file or files. The critical instruction for this is **DATA**, which reads data from a file into one or more series variables stored in memory.

You can write out the appropriate **OPEN DATA** and **DATA** instructions directly, or you can use one of the *Data Wizards* (on the *Data/Graphics* menu) to generate the necessary instructions.

The FORMAT option on **DATA** tells RATS what type of file you will be reading. Depending on the type of file, you may also need to use the ORGANIZATION option to specify whether the data series on the file are arranged in rows or columns. Note that the default format option for reading data is simple free-format text. RATS does not attempt to determine the format of the data file itself based on the filename extension—you need to set the FORMAT option yourself.

Getting Data Out: The COPY Instruction and File–Export...

The companion instructions to **OPEN DATA** and **DATA** are **OPEN COPY** and **COPY**, which write data series to a file. **COPY** can produce data files in a wide range of formats, including spreadsheets, RATS format files, text files, and binary files.

For RATS format files, **COPY** can only create a new file. To write data to an existing RATS file, use the instructions **DEDIT**, **STORE**, and **SAVE** (page Int–110).

If you prefer to use menu operations to write the data, you can:

1) Use *View—Series Window* to list the series available in memory.
2) Select (highlight) the series you want to write to a file from that list.
3) Do *File—Export...*, select the desired file format from the drop-down list, enter a name for the file, and click OK.

You can also open or create a RATS format file (using *File—New* or *File—Open*) and then drag and drop series from the Series Window onto the RATS file window.

Supported Formats

We list the formats in Section 2.2, divided into three broad categories. You can find detailed information on each format (indicating where it can be used and any special details) in the help.

Other Instructions for Reading and Writing Data

DATA and **COPY** can only read and write SERIES type variables. For other types of variables (such as scalar reals, integers, arrays, labels, strings), you can use the **INPUT** or **READ** instructions to read data into RATS, and the **WRITE** or **DISPLAY** instructions to write data to a file. The **REPORT** instruction can be useful for creating specialized reports of data and statistics, similar to those created by RATS instructions like **LINREG** or **STATISTICS**.

Moving Data Around

RATS allows you to write data in a wide range of forms using **COPY**. You can also open or create RATS format files using *File—New* or *File—Open*, and move data to and from memory one RATS format file to another (or from the working data series in memory to the RATS file) by dragging and dropping series from one window to another.

In fact, with RATS format used as an intermediate stop, you can easily translate data from, for instance, a poorly formatted text file to a well–organized spreadsheet.

The RATS file format is directly portable across different platforms (Mac, Windows, UNIX) with no translation required. You simply need to be able to transfer binary data. XLS, XLSX, WKS, PRN, DBF, DTA, DIF and the RATS Portable format are some other machine-independent formats which RATS supports.

Double-Check Your Data!

It is very important to be sure that data are being read in properly, particularly when reading from a data file for the first time. The tutorial in Chapter 1 includes a number of suggestions for doing this, particularly using quick views of the sample statistics. In a RATS program file, you can use the **TABLE** instruction immediately after the **DATA** instruction(s) to get the same type of quick statistics table.

2.2 Data/Copy Formats

RATS can read or write data in around twenty different formats. Some of these are strictly for input, some strictly for output, and some can be used for either. The details on each of these is provided in the help. The different formats generally fall into one of three broad categories.

Time Series Databases

Here, we look at file formats specifically designed to handle time series data. Data are organized by named series. Each of these series has its own calendar scheme and range of data. These are usually fairly complicated proprietary formats, since they have to be able to code up many types of date schemes, and must allow for very large amounts of data.

The most important of these is RATS format (**FORMAT=RATS**). A "text" version of the RATS format is the RATS Portable Format, useful mainly for archiving data in a "human-readable" form. We use **FORMAT=PORTABLE** for describing this format.

FRED® (Federal Reserve Economic Data) is a (free) on-line database provided by the St. Louis Federal Reserve Bank (*research.stlouisfed.org/fred2*). This requires an active internet connection. You can read series in directly with FORMAT=FRED, but you also have the option of browsing the database interactively—on the *Data/Graphics* menu, select *Data Browsers–FRED(online) Browser*. This displays a list of the main database categories in a new window. Double-click on a category (or sub-category) to see a Data File Window with the series available in that category.

The other formats listed below are available only in the Professional version of RATS, and some aren't included on certain platforms. These are all proprietary formats, so our ability to support them depends upon the type of support offered by the owner of the format. You will also need a subscription to a database or service provided by the owner in order to use them. These other formats are:

Citibase/DRI/Global Insight native format. We call this **FORMAT=CITIBASE**. This is available on all platforms.

CRSP® (Center for Research in Security Prices) data, available from the University of Chicago's Booth School of Business (*www.crsp.com*). We call this **FORMAT=CRSP**. It's available for Windows and some UNIX systems.

Fame® is the native format for the FAME database management program provided by SunGard. We call this **FORMAT=FAME**. This is available on Windows and certain UNIX systems.

Haver DLX is the native format for Haver Analytics (*www.haver.com*). The Haver USECON data which can be purchased from Estima are provided in both RATS format and in DLX format. We call this **FORMAT=HAVER**. It's available on Windows only.

Labeled Tables and Spreadsheets

These form a rectangular table of data with each column representing a series and each row representing an observation. Each column is labeled with the series name. Each series has at least nominally the same range, and same date scheme (if any) though there might be missing values in some of the series.

The most commonly used of these is one of the Excel formats, though they also can be in the third, less structured category. For formats through Excel 2003, use the option **FORMAT=XLS**. For Excel 2007 and later, use **FORMAT=XLSX**. You can pull data off more than one worksheet within an Excel workbook (using the SHEET option on **DATA**), but can only access one worksheet per **DATA** instruction—use additional **DATA** instructions to get data from more than one worksheet.

RATS accepts several "delimited" text formats. **FORMAT=PRN** will accept data fields which are separated by commas or "white space" (spaces or tabs). For writing data, you can also use **FORMAT=CDF** (comma delimited format) or **FORMAT=TSD** (tab separated data) to get specific field separators.

EViews® workfiles fall into this category. RATS can process data series (but not other objects). This uses **FORMAT=WF1**.

DTA is the native format for Stata® data files. Use **FORMAT=DTA** for that.

If you have the Professional level of RATS, you can also read data using SQL queries on databases which support ODBC (such as Oracle and Access). This is a bit more complicated than other formats since you have to provide the SQL query that sets up the table of interest. This is **FORMAT=ODBC**.

DIF is a (rather bulky) text format for transmitting the content of a simple spreadsheet in a text, rather than binary form. It's rarely used now, except as a copy-paste format for data to and from spreadsheets. It uses **FORMAT=DIF**.

WKS is an older spreadsheet format used by the Lotus spreadsheets. It's perfectly adequate as a storage format for data, and quite a few legacy data sets are saved in it. As with Excel, it can also be in the third category. This uses **FORMAT=WKS**.

DBF is the database format used by dBase and compatible programs. It uses **FORMAT=DBF**.

Unlabeled Text and Spreadsheets

These either don't have labels for individual series, or have "labels" which have illegal characters for RATS series names (such as spaces or parentheses).

The most important of these is free-format, which is just a text file with numbers, delimited with commas or "white space" (spaces, tabs, line breaks). While we would never recommend saving your own data in this, there are a large number of existing data files done in such an unstructured way. Also, if you ever have to scan data out of a book, you will likely end up with this. This uses **FORMAT=FREE**.

Another unlabeled text "format" is the collection of FORTRAN formats. This is likely only of value if you have a very old text data file which, in order to keep the size down, squeezed data into undelimited fields. (For instance, first five positions are the first number, second five are the next number).

MATLAB is the native data format for the MATLAB programming language. This could also fall into the "Time Series Databases" category, if each series is a separate one-column matrix. If, instead, you have a single matrix with multiple columns forming your dataset, it falls into this group. This uses **FORMAT=MATLAB**.

The least-recommended of all the formats is native binary. *Never* save data in this format—there's no identifying information on the file, you can't read the data without the proper program and it isn't necessarily portable from one type of computer system to another. It's **FORMAT=BINARY**.

Any of the "spreadsheet" formats (XLS, XLSX, WKS, DIF, PRN) can also be treated this way if you have extra comment lines or missing or unusable labels.

Series Labels on the Files

The advantage of the Time Series Databases and the Labeled Tables is that they have series labels already on the file, so you can pick and choose which series you want. If the name on the source is either not directly usable as a RATS variable (for instance, it includes spaces) or is too cryptic to be useful (the commercial databases need naming conventions to distinguish thousands of series), you can redirect the source name to a RATS name with a field myseriesname<<"fileseriesname". For instance

```
cal(m) 1990:1
data(format=fred) / cudur<<"CAPUTLGMFDS" cunondur<<"CAPUTLGMFNS"
```

reads the current information for capacity utilization for durable and for non-durable manufacturing from FRED and maps them to shorter names for use in RATS.

Formats for Input Only

CITIBASE, CRSP, DTA, FRED, MATLAB, ODBC, WF1

Formats for Output Only

RATS supports two types of specialized output formats for generating "tables". These are **FORMAT=TEX**, which generates a TeX tabular environment, for inclusion in a TeX document and **FORMAT=HTML**, which generates an HTML table for inclusion on a web page.

2.3 Where Are Your Data Now?

The suggestions below are designed to help you figure out the best approach to getting your data read into RATS so you can make use of them. Details on the various file formats supported by RATS follow later in this chapter and in *Additional Topics*.

In a Time Series Database

Except for FRED, these can all be read using the proper sequence of **CALENDAR**, **OPEN DATA** and **DATA** instructions. See Section 2.7 for more on the use of RATS format; otherwise, check the description of the format in the help. Again, note that most of these are available only for the Professional level of RATS.

RATS, FRED, Fame and Haver DLX all have "browser" modes, which are Data File Windows. You can use that to navigate through the contents of the file, using the *View—Reset List* menu item or "Change Layout" toolbar icon () to restrict the list based upon frequency, start or end year, comments or names (or any combination). You can drag and drop series you select onto the Series Window, or onto a RATS Data File Window, or export them directly to some other type of file (such as a spreadsheet). You open the browser for a RATS data file by opening it with the menu *File—Open*; the other browsers are submenus for *Data/Graphics—Data Browsers*.

If you are working with the browser for a RATS file, you can also select the series you want and use the *Data (RATS Format)* Wizard. That will look at the ranges and frequencies of the series that you selected, and come up with a common data range.

In a Labeled Table

These are all included in the *Data (Other Formats)* Wizard, which we would recommend you use the first time you try to work with a particular file. The wizard will always read all the series from the file, which might not be what you want. However, you can always edit the **DATA** instruction produced by the wizard to eliminate the unwanted series. Save the edited program file, and use that as the basis for further work.

See Section 2.8 and the next paragraph for more on the "spreadsheet" formats (`XLS`, `XLSX`, `WKS`, `DIF`, `PRN`). Because there are many ways to pull data out of these, when you use them as a Labeled Table, you need to include the option `ORG=COLUMNS`.

In a (Less Structured) Spreadsheet

Spreadsheets can often include additional information besides the data and labels (and possibly dates) that are required for treatment as a Labeled Table. If you have a well-structured labeled table *within* the spreadsheet, you probably can read it as a Labeled Table by using the `LEFT` and `TOP` options to restrict the attention of the **DATA** instruction to the area that you want. For instance, if you have ten lines of descriptive headers, you would add the option `TOP=11`. The *Data (Other Formats)* Wizard can help you set those options.

If you have a properly labeled table, except that the series are arranged by rows, rather than columns, you can use the option ORG=ROWS. You can combine that with the TOP and LEFT options if needed to isolate the data table from other information. There are several other options for reading data out of less-structured spreadsheets which are covered in Section 2.8. Note, however, that in many cases, the simplest approach is simply to pull the data into a spreadsheet program and edit out the unwanted parts of it.

In a Text File

What you should do depends on the format that you have inherited and how big the data set is. If the data is a well-structured Labeled Table, you can just read it as described above.

If the file is fairly well-organized in columns and rows, but without series names, you can probably still process the file using the *Data (Other Formats)* operation. In most cases, you can just select "Text files (*.*)" as the file type. You'll just have to provide labels for the series. If the data file is fairly stable, we recommend that you then write the data from RATS into a RATS format or spreadsheet format file for later use.

See Section 2.9 for more tips on reading text files.

In a Database File

If the data are in a database you can access using SQL commands, you should be able to read it with the Professional version of RATS, using **DATA** with FORMAT=ODBC. See the description of the FORMAT=ODBC in the help.

On Paper

If you have quite a bit of data in printed form, you're probably best off scanning it to create a text file, and continuing as described under "In a Text File." Since character recognition software is still imperfect, you'll have to be extra careful in checking that the data came in properly. If you need to key in your data, you can:

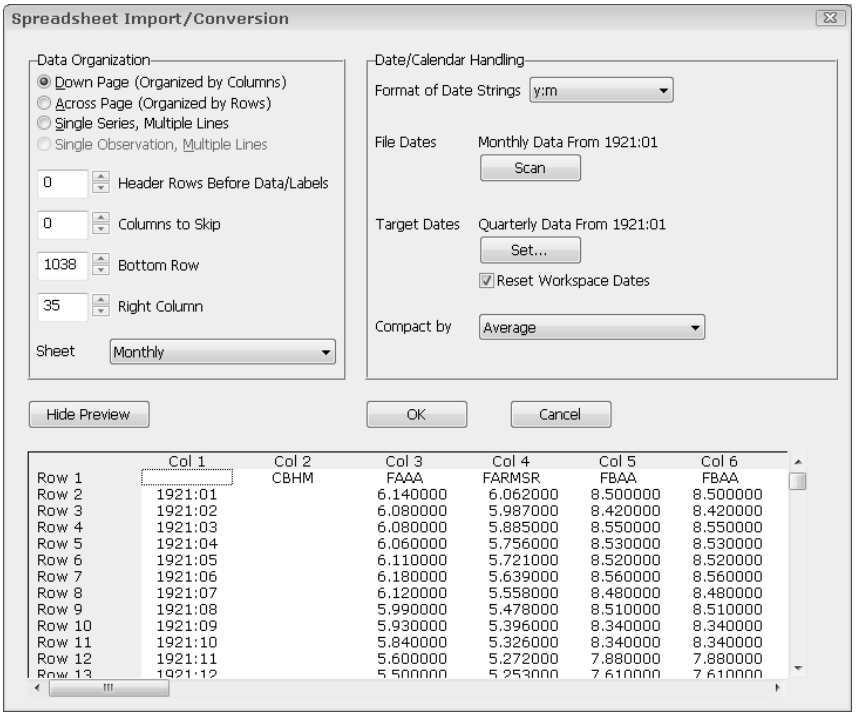
- type the data into a spreadsheet, a database application, or a text file (using RATS or any other text editor) and read the data using the steps described in the relevant section above.
- Use the *Data—Create Series* operation and type the data directly into RATS memory, and then use *File—Export...*.
- Open or create a RATS format file (using *File—Open* or *File—New*), and then use *Data—Create Series* to type the data directly into a RATS format data file.

The RATS data editor used by the *Create Series* operation is designed to handle time-series data sets easily, where you deal with one series at a time. If your data set is more "case-oriented," a spreadsheet or data base editor is likely to be more convenient, since you can edit across series.

Whatever method you use, be sure to check the data after reading it into RATS.

2.4 The Data (Other Formats) Wizard

For most users, the bulk of your work will probably involve reading data from Excel spreadsheets or text files, and as demonstrated in Chapter 1, the *Data Wizards (Other Formats)* operation is usually the easiest way to go about this. Because you'll use it often, let's take another look at a sample Data Wizard dialog:



	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
Row 1		CBHM	FAAA	FARMSR	FBAA	FBAA
Row 2	1921:01		6.140000	6.062000	8.500000	8.500000
Row 3	1921:02		6.080000	5.987000	8.420000	8.420000
Row 4	1921:03		6.080000	5.885000	8.550000	8.550000
Row 5	1921:04		6.060000	5.756000	8.530000	8.530000
Row 6	1921:05		6.110000	5.721000	8.520000	8.520000
Row 7	1921:06		6.180000	5.639000	8.560000	8.560000
Row 8	1921:07		6.120000	5.558000	8.480000	8.480000
Row 9	1921:08		5.990000	5.478000	8.510000	8.510000
Row 10	1921:09		5.930000	5.396000	8.340000	8.340000
Row 11	1921:10		5.840000	5.326000	8.340000	8.340000
Row 12	1921:11		5.600000	5.272000	7.880000	7.880000
Row 13	1921:12		5.500000	5.253000	7.610000	7.610000

To get to this point, we've done the following:

- Selected the *File—Data Wizard (Other Formats)* operation, selected "Excel 2007 Files (*.xlsx)" from the list of file types, and opened the file `HAVERSAMPLE.XLSX`.
- Used the "Sheet" button to preview the two sheets on this file. Here, we've selected the "Monthly" sheet.
- Clicked on "Scan" to have RATS determine the frequency and dates of the data.
- Used the "Set" button to tell RATS to read the data at a *quarterly* frequency, rather than at its native monthly frequency.

Because we've set our Target Dates field to a lower frequency than the source data, the "Compact by" button is active. You can use this to choose the compaction method RATS will use. Here, we've selected the default choice—taking simple averages of the subperiods (page Int–104). If we didn't want the Wizard to generate a **CALENDAR** instruction at all (for example, if we had already set a quarterly **CAL** earlier in the session), we could just turn off the "Reset Workspace Dates" switch.

2.5 Changing Data Frequencies

The **DATA** instruction can automatically convert data from one frequency to another. For this to work:

- the source file *must* contain dates that RATS can process or you must be able to provide the source date information yourself.
- you must set the **CALENDAR** instruction to the desired (new) frequency. You can do this by typing in **CALENDAR** directly, by using the *Data/Graphics—Calendar* operation, or by setting the "Target Dates" field in the Data Wizard (page Int–103).

Given a mismatch between the frequency of the data on the file and the **CALENDAR** seasonal, RATS will automatically compact or expand the data to match the **CALENDAR** setting. This works for any of the file formats for which RATS can process dates (any of the Time Series Database formats and most of the Labeled Table formats).

Compacting Data

To compact from a higher frequency to a lower frequency, just follow these steps:

1) As noted above, make sure the source data file contains valid date information at the original (higher) frequency. See page Int–106 for tips if you don't already have date information on a file.
2) If you're using a *Data Wizard*, just make sure that the "File Dates" field accurately reflects the higher frequency of the source data. Then, set the "Target Dates" fields to the desired lower frequency. Click on OK to read the data.

   If you are typing in commands directly, set the **CALENDAR** to the target (lower) frequency. For example, if you are compacting monthly data to a quarterly frequency, specify a quarterly **CALENDAR**. Then use **OPEN DATA** and **DATA** to read the file.

RATS will automatically compact the data to match the **CALENDAR** frequency using the method specified by the COMPACT or SELECT option, or the "Compact by" field on the Data Wizard. The default is COMPACT=AVERAGE.

Compact and Select

The COMPACT and SELECT options allow you to select from several compaction methods. Note that the two options are mutually exclusive. If you try to use both, RATS will honor the SELECT choice. The choices are:

compact=[average]/sum/geometric/first/last/maximum/minimum

AVERAGE	simple average of the subperiods
SUM	sum of the subperiods
GEOMETRIC	geometric average of the subperiods
FIRST/LAST	first or last entry of each subperiod, respectively
MAX/MIN	maximum value or minimum value from each subperiod

select=*subperiod to select*
compacts data by selecting a single subperiod from within each period.

Suppose you have a quarterly **CALENDAR** and you want to read a monthly data series. **DATA** with SELECT=3 will read in the third month from each quarter (that is, the March, June, September, and December observations from each year). With the default option of COMPACT=AVERAGE, each quarterly value will be the average of the three months which make up the quarter.

It is more complicated when you move between dissimilar frequencies: weekly to monthly, for instance. If you use SELECT=2, it will select the second *full* week within the month. Suppose you have the weekly observations shown below. Since weekly data are classified in RATS according to the end of the week, March 13, and not March 6, will end a full week for this data set. If you compact data using the other methods, RATS will give 6/7 of the March 6 value to March, and 1/7 to February. If the input data are:

Week ending:	March 6	March 13	March 20	March 27	April 3
	15.0	11.0	13.0	18.0	20.0

the value for March (with COMPACT=AVERAGE) is:

(6/7x15.0 + 11.0 + 13.0 + 18.0 + 4/7x20.0)/(6/7 + 3 + 4/7) = 14.97

Expanding Data to Higher Frequencies

The steps are the same as those described under Compacting Data on page Int–104.

If RATS detects that the data on the file being read are at a *lower* frequency than the current **CALENDAR** setting or Data Wizard target frequency, it will automatically expand the data to the higher frequency by setting each subperiod equal to the value for the full period. For example, if you set a quarterly **CALENDAR** and read in annual data, RATS will set the value of each quarter in a given year to the annual value for that year.

For dissimilar frequencies, the value of a period which crosses two periods of the lower frequency data is a weighted average of the two values. For instance, in moving from monthly to weekly, the value for the week ending March 6 will be:

1/7 x February + 6/7 x March

For a more complex interpolation, you would first read the data as described above to expand to the higher frequency, and then apply the desired interpolation routine to the expanded data. RATS ships with several interpolation procedures you can use, but we primarily recommend the **@DISAGGREGATE** procedure, as it provides options for choosing from several models and techniques.

The following example reads in quarterly GDP data at a monthly frequency. It uses **@DISAGGREGATE** to produce a monthly series from the "step" function that **DATA** generates. Since GDP is ordinarily quoted in annual rates, we want to maintain the aver-

age at the higher frequency; if we had a series that quoted in totals (sales figures, for instance), we would use MAINTAIN=SUM.

```
calendar(m) 1947
open data haversample.rat
data(format=rats) 1947:1 2006:4 gdp
set trend = t
@disaggregate(factor=3,model=loglin,maintain=average) gdp / gdpm
# trend
```

No Dates on Source File?

If you have a data set that you want to compact or expand, but your data file doesn't include date information, you can either

- add dates to the original file directly (see page Int–119 for suggestions if using a spreadsheet), or
- use the CALENDAR option on **DATA** to describe the date scheme of the source file. This is created using **CALENDAR** with the SAVE option.

For example, suppose the quarterly GDP data from the previous example was provided in a plain text file which we'll call gdpnodates.txt. You know that the data are quarterly, and start in 1947Q1, but the text file doesn't contain any date information. The following defines a **CALENDAR** scheme for the quarterly data, then reads that into a monthly workspace **CALENDAR** scheme. It then uses **@DISAGGREGATE** to take the crudely expanded data to a better estimate of monthly GDP.

```
calendar(q,save=q1947cal) 1947
open data gdpnodates.txt
calendar(m) 1947
data(format=free,org=columns,calendar=q1947cal) 1947:1 2006:12 gdp
set trend = t
@disaggregate(factor=3,model=loglin,maintain=average) gdp / gdpm
# trend
```

2.6 Missing Data

Coding

Internally, RATS represents missing data with a special value. On most machines this is the value "+infinity." On output, a missing value is denoted as "NA" for Not Available. In a RATS expression, this value is available as %NA. Thus,

```
set fixx = %if(x<0.0,%na,x)
```

will set FIXX to be missing for entries where X<0 and to X otherwise. You can test whether a value is missing using the %VALID function: %VALID(x) is 0.0 if X is missing and 1.0 otherwise. You can also use X==%NA.

RATS has special ways of handling missing values for each of the formats.

Spreadsheet Files

The "+infinity" coding is the same as is used by the major spreadsheet programs for N/A's. Thus, if a cell contains the explicit missing value function (NA() in Excel), RATS will read a missing value at that cell. When you export data to the spreadsheet, RATS passes the code value (as a number, not a function), so it will display as NA or #N/A on the spreadsheet.

RATS will also interpret blank cells within the spreadsheet as missing values.

Spreadsheet-Style Text (PRN) Files

Missing values are a problem if you have to save the data in a space or tab separated text format (PRN or TSD). You can't simply leave the cell blank in the text file, as you can with an actual spreadsheet, because a blank cell is indistinguishable from the blanks used to separate data items. Instead, you'll have to use one of the codings described in the next paragraph.

Text Files

RATS will accept the following as codes for missing values in text-format data files:

- The characters "NA" or "#N/A" (upper or lower case)
- A decimal point, followed by zero, one, or two non-numeric characters (for example, . or .NA)

Note, however, that you cannot use these within a RATS expression: use %NA for that.

RATS will interpret *any* block of non-numeric characters as a missing observation. However, if the characters don't fit the description above, RATS will issue a warning message that it has encountered invalid input.

Numeric Codes

If you have data which use a specific numeric value (–999 or something similar) to indicate a missing value, use the option MISSING=*missing value code* on **DATA**. *Whenever possible, use integers for missing value codes*. The realities of numerical precision in the computing world mean that RATS may not be able to match exactly a decimal-valued code such as –999.99.

Suppose your data set uses –9999 for missing values. You could use something like the following:

```
data(format=prn,org=columns,missing=-9999) / sales revenue
```

This would read the series SALES and REVENUE, and convert any values of –9999.0 to missing values.

If you have several missing value codes (such as –999 = no response, –998 = invalid response), you have two options:

- Edit the file and replace the two codes by a single code.
- Read the data and alter it within RATS.

This is an example of the latter:

```
data(format=free,org=col) / income charitable mortgage
set income = %if(income==-999.or.income==-998, %na, income)
```

The %IF function tests whether the current entry of INCOME is equal to either of our two missing value numeric codes. If so, the function returns the %NA missing value code, which is then stored in INCOME. Otherwise, the existing value of INCOME is retained.

To apply this procedure to multiple series, we can use a **DOFOR** loop:

```
dofor i = income charitable mortgage
   set i = %if(i{0}==-999.or.i{0}==-998, %na, i{0})
end dofor
```

Note that the index variable I will actually be an integer variable containing the series number associated with the current series. As a result, we need to use the lag notation {0} (specifying the zero lag) on the right hand side of the **SET** instruction. This tells RATS to treat I as a series, not as an integer number.

Skipped Dates

Suppose that you have a (daily) data set in which holidays and other non-trading days are skipped. If you have no date information on the file, there is little you can do: you will have to treat it as an irregular time-series—omit the **CALENDAR** entirely, or use **CAL(IRREGULAR)**.

If you *do* have dates on the file, **DATA** (and **STORE** with CONVERT) will code entries corresponding to the skipped dates as missing values. For example, consider the following portion of an XLS file, which skips the (U.S.) Thanksgiving holiday on November 23, 2000:

```
             SALES_DATA
2000:11:20    3590.50
2000:11:21    4256.05
2000:11:22    2987.23
2000:11:24    6799.87
```

You might read the data with the following instructions:

```
calendar(d) 2000:1:2
open data sales.xls
data(format=xls,org=col) 2000:1:2 2000:12:31 sales_data
```

The data in the SALES_DATA series for the week of Thanksgiving would be:

```
2000:11:20    3590.50
2000:11:21    4256.05
2000:11:22    2987.23
2000:11:23         NA
2000:11:24    6799.87
```

If you really want to skip the holidays without having gaps in your data (in the sense that you want to treat the data for November 22 and November 24 as adjoining entries), you cannot treat the data set as "Daily" because RATS insists that daily data be five days a week. If you use **CALENDAR(IRREGULAR)**, RATS will ignore the dates on the data file and read the data from the file into consecutive entries. There is little difference between the two ways of handling the holidays, unless you do some type of time-series analysis that uses lags or leads.

Another option for these situations is to go ahead and read the data using a DAILY or SEVENDAY **CALENDAR**, as in the Thanksgiving example above, then use the **SAMPLE** instruction to create "compressed" versions of the data without the missing-values:

```
sample(smpl=%valid(sales_data)) sales_data / sales_nomissing
```

You can use the uncompressed data for printing or generating graphs (where you want to include date information), and use the compressed series for estimating Box–Jenkins, GARCH, and other similar models that don't accept missing values.

If you want to patch over a missing value like this with the previous value (which would make no sense with the sales data, but might if you have, for instance, price data), you could do that with something like:

```
set patched = lastprice=%if(%valid(price),price,lastprice)
```

This will use the value in the series PRICE if PRICE isn't missing, and will use the last non-missing value if it is.

2.7 RATS Format

RATS format is designed for the special problems of time series data. It has many advantages over other formats:

- Data access is more flexible: you can set up a single file with many series and select only the ones you need for a particular application. You can also select a specific set of entries—you don't have to read in entire series.
- The data series do not need to have identical structures. For example, you can mix monthly, quarterly and daily series on one file and can have series running over different intervals.
- RATS can automatically compact or expand data to match the current **CALENDAR** frequency. See Section 2.4.
- There are many ways to add data to a file or edit series already on a file.
- Data retrieval is extremely fast.

RATS also allows a certain amount of flexibility in reading the Labeled Table files (page Int–99): you can select specific series off a file and restrict yourself to a subset of entries, and, under certain circumstances, you can compact and expand.

Stored Information

For each series on the file, RATS retains the following information:

- the series name
- the frequency of the data (annual, quarterly, monthly, panel, etc.)
- the dates (or entries) for which the series is available
- up to two lines of comments
- the data itself

Reading Data into RATS

To work with data from a RATS format file, you can:

- Open the file using **OPEN DATA**, and read the data using **DATA(FORMAT=RATS)**, or
- Open the file using *File—Open*, select (highlight) the series you want, and use the *Data (RATS Format)* wizard from the *Data/Graphics* menu to read in the data.
- Open the file using *File—Open* and drag and drop series from the RATS file onto the Series Window (displayed using *View—Series Window*).

With RATS format, you can read any combination of series from the file. You can also read specific sets of observations from the full data set by using the *start* and *end* parameters on the wizard or the **DATA** instruction. This is one of the most important advantages of RATS format.

If you omit the *list of series* when using **DATA**, RATS will read *all* the series on the file. Be cautious when working with large files, or you might read in many unneeded series.

To change the name of the series as stored on the file, you can either open the file using *File—Open* and use the "Rename" toolbar button, or else open the file with **DEDIT** and use the **RENAME** instruction.

Instructions for Creating and Editing RATS Files

RATS provides special instructions for creating and editing RATS format data files. *Never try to make changes to a* RATS *data file using a text editor*. The instructions for working with RATS format files are:

COPY

When used with the option FORMAT=RATS, this writes a set of series to a *new* RATS format file. This is the easiest way to write data to a RATS file, but cannot be used to append data to an existing file—use **DEDIT**, **STORE**, and **SAVE** for that.

DEDIT

initiates the editing of a new or existing file. Once you have opened a file using **DEDIT**, you can use the following instructions. Several of these are rarely used now, since it's easier to make changes on-screen using the RATS Data File Window.

STORE	adds specified series to the file, or converts the contents of a data file of another format.
SAVE	saves changes to the file.
CATALOG	produces a directory of series on the file.
PRTDATA	prints one or more of the data series stored on the file.
QUIT	ends an editing session *without* saving the changes (do **SAVE** then **QUIT** if you want to save changes and then close the file).

The following are no longer documented in the *Reference Manual*, but are included in the help.

EDIT	performs on-screen editing of an existing or new series.
INCLUDE	adds a single series to the file.
UPDATE	makes changes to a small set of entries of an existing series.
DELETE	deletes a series from the file.
RENAME	renames an existing series.

RATS Data File Windows

When you open an existing RATS file using *File—Open*, or create a new file using *File—New—RATSData Window*, RATS displays the contents of the file in a window. From this window, you can:

- Read data into memory by selecting the series you want from the data window and then doing *Data/Graphics—Data (RATS Format)*.
- Read data into memory by dragging and dropping series *from* the RATS file window to the Series Window (displayed using *View—Series Window*).
- Write data *to* the RATS file by dragging series from the Series Window and dropping them on the RATS Data File Window.
- Use *File—Import* to bring data from other files into the RATS format file.
- Use *File—Export* to export data from the RATS format file.
- Double-click on a series to view or edit the data in that series.
- Select *Data/Graphics—Create Series (*or click on the icon ) to create a new series on the file.
- Rename a series by selecting the series and clicking on the 

icon.
- Use the *View—Reset List* menu item or  toolbar icon to restrict the list based upon frequency, start or end year, comments or names (or any combination)
- Use the *View* menu operations or corresponding toolbar icons to display time series graphs, histograms, and box plots, generate a table of basic statistics, or compute a set of autocorrelations for the selected series.

You can also right-click (or `<Command>+Click` on the Mac) on a series (or list of series) and select operations like "Cut", "Copy", and "Export..." from the pop-up menu.

Examples of DATA

Suppose the data file `MYDATA.RAT` has `REAL_GDP`, quarterly from 1947:1 to 2018:1, and `TBILLS`, monthly from 1946:1 to 2018:5.

```
cal(q) 1947:1                                  Sets a quarterly CALENDAR
open data mydata.rat
data(format=rats) 1947:1 2018:1 real_gdp  tbills
```

will read `REAL_GNP` and quarterly averages (the default compaction option) of `TBILLS`, over the range 1947:1 through 2018:1.

```
cal(m) 1954:1                                  Sets a monthly CALENDAR
allocate 2016:12
open data mydata.rat
data(format=rats) / tbills
```

will read the (monthly) data for 1954 through 2016 for `TBILLS`.

Examples of Editing

```
dedit company.rat
cal(q) 1950:1
open data ascii.dat
data(format=free) 1950:1 2017:4 order invent shipment backlog
store order invent shipment backlog
save
```

reads four series from a free–format file and saves them on the RATS file COMPANY.RAT.

```
cal(d,save=dailycal) 1991:1:2
dedit(new) options.rat
open data options.wks
store(convert=wks,org=col,calendar=dailycal)
save
```

converts all data from the (undated) WKS file OPTIONS.WKS, creating OPTIONS.RAT. The data will be stored as daily data, beginning with entry 1991:1:2.

The RATSData Utility Program

RATS ships with a stand-alone, menu-driven utility program called RATSData. Most of the functionality of this program has now been incorporated directly into RATS, using the *New*, *Open*, *Import* and *Export* operations on the *File* menu, and the various operations (and associated toolbar icons) on the *View* menu.

You may still find RATSData useful for some tasks. Windows users will find shortcut icons for RATSData in the WinRATS folder on the Start menu and desktop. Mac users will have a RATSData icon in their MacRATS folder. Linux and UNIX users can run the `ratsdata` executable file. See the RATSData help for details on using the program.

Using Older RATS Format Files

RATS Version 11 uses the same RATS file format as Versions 4 through 9, so files can be interchanged seamlessly among these versions. It can also read from files created with Versions 2 and 3 of RATS, which should cover any RATS format file created since around 1987.

2.8 Spreadsheet and Delimited Text Formats

Supported Formats

Spreadsheets and comma, tab, or space delimited text files are popular tools for data storage, and many commercial database vendors allow users to download data in these formats. RATS supports several formats associated with such programs, with these choices for the FORMAT option on **DATA** and **COPY**:

FORMAT Option	File Type
XLSX	Microsoft Excel 2007 and later XLSX spreadsheets
XLS	Microsoft Excel (2003 and earlier) XLS spreadsheets
WKS	Lotus 123 worksheets (WK3, WKS, WK1, WRK and WR1 files)
PRN	Print Files (text files with a spreadsheet–style layout)
CSV	Comma separated text files
TDF	Tab delimited text
DIF	Data Interchange Format

XLS/XLSX are typically the best choices from among these.

RATS supports these formats in several ways:

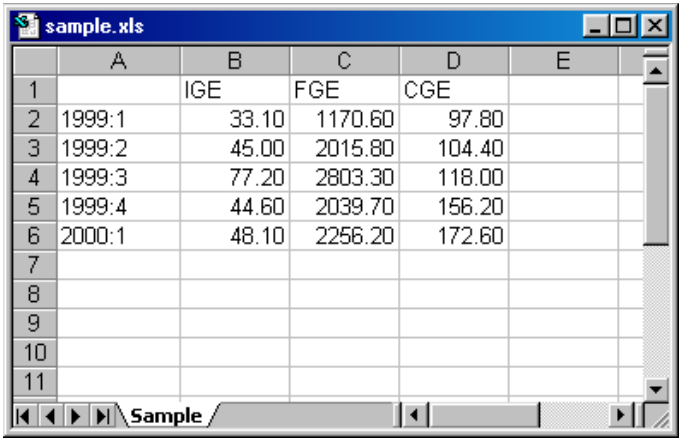
- You can read data from a file using the *Data Wizard (Other Formats)* operation, the instruction **DATA**, or by importing into a Series Window.
- You can write data to a file using the instruction **COPY** or by exporting from the Series Window.
- You can convert data from a spreadsheet directly to a RATS format file using the CONVERT option of **STORE** or by importing into a RATS Data File Window.
- You can read and write matrices using **READ** and **WRITE**.
- You can export data from Report Windows using the export operations.

As Labeled Tables

All of these can be treated either as Labeled Tables or as Unlabeled Data, depending upon the form that the information takes. To be handled as a Labeled Table, it's necessary for the data to look something like the sample at the top of the next page. We refer to this as organization by *column* (series arranged in columns, one observation per row). The series can also run *across* the rows: dates (if present) in the first row and each row of data beginning with the series label in column A. This is *organization by row* or *variable*.

In either case, the block of data should be in the upper left hand corner of the worksheet (although options like TOP and LEFT can be used to ignore data above and to the left of the data block).

A third organization is for use when a single series is itself a block of rows *and* columns. That's ORG=MULTILINE. With that, you can only read a single series at a time, with the TOP, LEFT, BOTTOM and RIGHT options used to isolate the data.



	A	B	C	D	E
1		IGE	FGE	CGE	
2	1999:1	33.10	1170.60	97.80	
3	1999:2	45.00	2015.80	104.40	
4	1999:3	77.20	2803.30	118.00	
5	1999:4	44.60	2039.70	156.20	
6	2000:1	48.10	2256.20	172.60	
7					
8					
9					
10					
11					

Dates

The column (or row) of dates is optional. Dates permit more flexible use of the data. If your file includes dates, the date information must appear in the *first* column or row of the data block. You can enter the dates as strings or as functions or values. If you *do* enter them as functions or values, be sure to format the cells with a date format.

If you use strings, the dates must be have *some* type of delimiter between the year and period, or year, month and day; for instance, 1999:1, 1999Q1 or 1999-3-31 are acceptable, 199901 isn't. If the dates aren't "yearDperiod" or "yearDmonthDday" ("D" being some non-numeric delimiter), you need to use the DATEFORM option on **DATA** to provide the form being used. For instance, if you're using the notation qq-yyyy (01-1999, 02-1999, etc.), include the option DATEFORM="qq-yyyy". The Data Wizard will generally be able to properly detect any such non-standard coding for the dates.

With annual data, you can use just the year number in the data file (2010 rather than 2010:1), but RATS will only recognize this as a date if it is entered as a string rather than a numeric value. Remember that in your RATS programs themselves, you *must* include the ":1" for annual date references.

Labels

The labels at the beginning of each series are *not* optional if you want to treat the file as a Labeled Table. RATS uses the series labels to determine which series to read from the file. Labels should normally appear in row 1 (if organized by columns) or column A (if organized by rows). If you have additional rows of header information before the row of series labels (on data arranged by column), you can use "Header Rows before Dates/Labels" field in the Data Wizard or the TOP option directly on **DATA** to skip those.

Variable names must start with a letter or _, consist of letters, digits, _ , $ or %, and be sixteen characters or less in length (they can actually be longer, but only the first sixteen are significant). If you don't have the option of changing the names (for in-

stance), you can either use a redirection on **DATA** or skip the automatic label processing. For instance, suppose the sample file had labels of "Investment (GE)", "Value (GE)" and "Capital (GE)". Neither the spaces nor the parentheses are permitted in a RATS variable name. To redirect, use fields `myseriesname<<"fileseriesname"`:

```
open data sample.xls
cal(q) 1999:1
data(format=xls,org=col) 1999:1 2001:1 $
  ige<<"Investment (GE)" fge<<"Value (GE)" cge<<"Capital (GE)"
```

To skip label processing, use the option NOLABELS on **DATA**, combined with the TOP option to skip the line(s) with the (unusable) labels. So we could read this with:

```
open data sample.xls
cal(q) 1999:1
data(format=xls,org=col,nolabels,top=2) 1999:1 2001:1 $
  ige fge cge
```

Since we skipped the label processing, we have to provide our own labels for the series, and we have to read all the series from the file.

Reading Data

You can read data from a spreadsheet or delimited text file using the *Data (Other Formats)* Wizard or by typing in a **DATA** instruction with the appropriate FORMAT and ORG options. For multi-sheet workbooks, you can use the SHEET option to tell RATS which worksheet you want to read. By default, it will read data from the first worksheet on the workbook's list of sheets.

To use the Wizard, select *Data/Graphics—Data (Other Formats)*, choose the format of the file you want to read (such as "Excel 2007 Files (*.XLSX)") from the drop-down list in the dialog box, select the file you want to read, and then use the Wizard dialog box to read the file. If the source file has more than one worksheet, this has a popup box to allow you to select the worksheet you want.

If you are typing the instructions manually, use **CALENDAR** or the *Calendar* operation to set the frequency and start date (if using dated data), then use **OPEN DATA** to specify the file to be read, followed by a **DATA** instruction with the appropriate options. You can list the specific series you want to read on the **DATA** instruction, or omit the list and let RATS read all the series on the file.

Whether you can select specific observations out of the file depends in part on whether or not the file includes a date column:

- If there is no date column or row on the file, RATS assumes the first observation on the file corresponds to the first observation you request on **DATA**. You can ask only for entries at the beginning of the file, never a block in the middle or end. If you need to be able to read in a subset of the data, you could first convert the file to RATS format.

- If there *are* dates on the file, and the frequency on the file matches the current **CALENDAR** seasonal, **DATA** will locate the entries requested and fill in any skipped dates with missing values.
- If the frequency of the current **CALENDAR** does *not* match the frequency of the data on the file, RATS will compact or expand the data as it reads it in. See Section 2.4.
- If you don't use a **CALENDAR** instruction, RATS will read observations from the beginning of the file. Any dates on the file are ignored.

Examples

Make sure that you use the proper extension (XLSX, XLS, or PRN for example) on the file name. RATS *will not add it automatically*.

```
cal(q) 1999:1
open data sample.xls
data(format=xls,org=col)  1999:1 1999:4 cge fge
```

reads series CGE and FGE, for the dates 1999:1 through 1999:4, from the sample worksheet earlier in this section.

```
cal(q) 1999:1
open data sample.xls
data(format=xls,org=col) 1999:1 2000:3
```

This reads all the series on the sample file. Because there are no data for 2000:2 and 2000:3, those entries are filled with NA.

```
cal(m) 2002:1
open data daysales.xlsx
data(format=xlsx,org=col,compact=sum) 2002:1 2009:12
```

This reads data from an Excel 2007 XLSX file, summing the daily data on the file to create monthly data in working memory.

```
cal(q) 1950:1
open data sales.xls
data(format=xls,org=col,sheet="quarterly",verbose) 1950:1 2006:4
```

This reads data from an Excel workbook containing several spreadsheet pages. Here, we are reading from the sheet entitled "Quarterly". The VERBOSE option produces information about the data file, including the frequency and starting date. It will also indicate if any frequency conversion is being performed to match the **CALENDAR** frequency.

ORG=MULTIROWS

Consider the following. It has three years of quarterly data for two firms. This could be in any of the spreadsheet formats.

```
FIRM A
2007          11.3      11.6      10.9      12.3
2008          13.0      12.8      11.5      12.5
2009          12.9      13.0      13.2      13.6

FIRM B
2007          22.5      21.9      24.3      25.6
2008          21.9      21.8      22.6      23.5
2009          22.5      25.0      24.5      25.4
```

We can't easily convert this to a Labeled Table because the data for a single series span multiple rows. One approach (probably the simplest in this case) is to eliminate the "FIRM A" and "FIRM B" lines and delete the "2007", "2008", and "2009". If this is in a spreadsheet, you could just delete the first column, and save what's left as text, and read it as free-format (see Section 2.9) with

```
cal(q) 2007
open data firms.txt
data(format=free,org=rows) 2007:1 2009:4 firma firmb
```

An alternative is to use a set of options on **DATA** to isolate just the information that we need. All the spreadsheet formats have a third choice for the ORG option, which is ORG=MULTIROWS. This is for precisely this type of situation, where a data series covers several lines. In order to use this, however, we need to know (in advance) exactly how many data points there are in each series, since the limit on a series isn't defined by the end of the data file. The data for the first series starts at column 2 of row 2 and has twelve data points while the second starts at column 2 of row 7. We have to read each series separately with something like:

```
open data firms.xls
cal(q) 2007
all 2009:4
data(format=xls,left=2,top=2,nolabels,org=multi) / firma
data(format=xls,left=2,top=7,nolabels,org=multi) / firmb
```

Both the date scheme (quarterly 2007:1) and end date (2009:4 or twelve entries) need to come from us since those won't be recognized from the file.

Writing Data

There are several ways to write data to a spreadsheet file. The most common is to use the **COPY** command to write data series. Begin by opening a file for output with **OPEN COPY**. Then use **COPY** to write the data to the file. We recommend that you use the ORG=COL option. You can include dates on the file with the DATES option. Do a **CLOSE COPY** command if you want to be able to open the file in another application right away. For spreadsheets, this creates a new file (overwriting any existing file of the same name), so you *must* write the entire file using a single **COPY** instruction—you cannot use multiple **COPY** commands to append data to existing spreadsheets.

You can also write series to a spreadsheet by opening the Series List Window using *View—Series Window*, selecting the series you want to export, and doing *File—Export*.

```
open copy sample.xls
copy(format=xls,org=col,dates)  1999:1 2000:1  ige  fge  cge
```

produces the sample worksheet from earlier in the section.

Adding Dates to a Spreadsheet

One way to add dates to a spreadsheet file which doesn't have them is to read the data into RATS, set the desired **CALENDAR**, and write the data back to a new file. If you do that with **COPY**, use the DATES option; if you export it, make sure you check the "Label Observations with Dates" box.

If you prefer to add dates directly to a spreadsheet, here's an easy way to add monthly or quarterly date labels to a spreadsheet file. First, create a blank column (or row if the data are organized that way) and format it as one of the built-in date formats. Next, enter the date number corresponding to your first date in cell A2 (or B1 for ORG=ROWS) using the appropriate date function. In Excel, for example, you could enter the date June 1, 1999 with the formula:

```
=date(1999,6,1)
```

To fill the rest of the date column or row with monthly dates, enter the following formula in cell A3 (or C1):

```
=date(year(a2+31),month(a2+31),1)
```

If ORG=ROWS, use +b1+31 in the formulas, rather than +a2+31.

Finally, copy this formula to the rest of the cells in the date column or row. For quarterly dates, just use +92 in place of +31.

2.9 Text Files

Free Format

We use the term "free format" to refer to text files that contain only numbers, with no series names, date labels, or other alphanumeric labels. Numbers on the file can be separated by blanks, tabs, commas, or "new lines".

You can use the *Data (Other Formats*) operation to read data from free format files. Just select "Text Files" or "Comma Delimited" from the file-type list before selecting the file you want to read. When you complete the Data Wizard dialog box, RATS will prompt you for a name for each series being read in.

If typing in the **DATA** instruction directly, use FORMAT=FREE and the appropriate ORG option (ORG=COLS or ORG=ROWS), and include a list of the names you want to assign to the series. See below for more on reading free format files.

PRN and Other Delimited Formats

PRN format refers to a text file structured like a spreadsheet. The name dates back to the days of Lotus 123, where it described a printable (text) version of a spreadsheet. To be read as a Labeled Table, PRN files need to have data structured as described on page Int–114. Series should be arranged in columns or rows, and the file *must* contain names for each series. PRN files can also contain dates, in the form of date-format strings such as "yyyy/mm/dd" or "yyyy-qq". Note that you can't use any type of numerical coding for the dates. Numbers on the file can be separated by blanks, commas, tabs, and/or carriage-returns. The columns don't need to be nicely aligned since it's the delimiters that determine where one column ends and the next starts.

You can read the file using the *Data (Other Formats*) operation, or by typing in a **DATA** instruction with FORMAT=PRN and the appropriate ORG option.

RATS also offers two related choices: FORMAT=CDF for comma-delimited files and FORMAT=TSD for tab-delimited files.

For reading data, CDF, TSD, and PRN are interchangeable. Regardless of which you use, RATS will accept commas, tabs, or spaces as separators. For creating a file with **COPY**, the option will determine the type of separator used: commas for CDF, tabs for TSD, and spaces for PRN.

Free Versus PRN

Free format files can be very convenient if you want to type in a short data set, because you can create them with any editor that can save files as plain text. They are also the most natural format for data scanned in from printed material.

However, because these files do not contain series labels or date information, they are not good for long-term use, and RATS cannot read them with much flexibility. Usually, you will want to convert free-format files to RATS format files, or to one of the spreadsheet formats.

PRN format is a generally more useful and reliable. The presence of series names reduces the chance of mistakes identifying data, and also give you the option of only reading in selected series. The ability to include date labels is clearly an advantage for dated time series data.

If you have a text file that *does* have variable labels, or other non-numeric characters you could:

- read the file using FORMAT=PRN. This allows RATS to process the series labels and dates (if any).
- import the data into a spreadsheet and read that file into RATS (see Section 2.8). Spreadsheet programs have very sophisticated "parsing" dialogs for taking text and dividing it into spreadsheet cells.

If you have an otherwise well-formatted text file which does *not* include series labels, you could:

- edit out all the non-numeric characters and read the file as FORMAT=FREE.
- edit the file to add series labels and read it as FORMAT=PRN.

Reading Free Format Files

When reading a free format file, RATS reads series line by line, according to the following:

- If your data are organized by row, each variable must *begin* on a new line. The data for a given series can, however, extend over more than one line—just make sure that the data for the *next* series begins on a new line.

  If **DATA** does not find enough entries on a line to fill a series, it automatically moves on to the next line and continues reading numbers.

  Extra rows of spaces are fine (for instance, separating data for two series). FORMAT=FREE will just keep scanning until it hits the next set of numbers.

- If your data are organized by column, the data for each new observation must begin on a new line. As above, data for a particular observation can extend over multiple lines.

  If **DATA** does not find enough entries on a line to fill all the series for an observation, it automatically moves on to the next line and continues reading numbers. When it has read data for all series for that observation, it drops to the next line to start reading in the next observation.

- RATS interprets as a missing value the characters "NA" or "#N/A" (upper or lower case), or a period followed by zero, one, or two non-numeric characters. If it encounters any other non-numeric characters, RATS will interpret them as a missing observation and display a warning message.

Troubleshooting

Free format allows data for a single observation (or a single variable) to cover several lines. The disadvantage of this is that it becomes more difficult to pinpoint errors. If you forget one number early in the file, **DATA** will automatically pull in the next line to fill the omitted value, and throw the data off.

If you get the error "`Unexpected end of file...`" when reading with `FORMAT=FREE`, it means that RATS reached the end of the data file before it filled all of the data series. To determine what happened, do the following:

- Check your **ALLOCATE** setting or the *start* and *end* parameters on **DATA** to make sure they are set correctly. Note that if, for example, you have **ALLOCATE 2018:5:14**, you can do **DISPLAY 2018:5:14** to see how many data points are implied by the **ALLOCATE** range.
- Make sure that you have the proper `ORGANIZATION` option.
- Check that the data file has the number of observations and variables that you expect, at least at first inspection.
- If all seems to be correct and it's a small enough file, you can do a quick check for typographical errors.

If all this fails to locate the problem, you will have to let **DATA** help you find the problem. For illustration, suppose you have a file that looks like this:

```
1,2,3
10.20,30
100,200,300
1000,2000,3000
```

This is supposed to have 4 observations on each of three series, but the second line has a decimal point between the 10 and the 20, where it should have a comma, so the line only contains two values (10.20 and 30) rather than three. If we read this with

```
data(org=col) 1 4 first second third
print
```

we will get the error message,

```
Unexpected end of file while processing line 5. (series FIRST entry 4).
```

We can tell the following from this:

1. **DATA** thinks it needs a fifth line to read the requested data. Our data set is supposed to have four lines—this tells us that, in fact, the file has as many lines as we think.
2. RATS was trying to read the fourth observation from a fifth line, so for some reason we are precisely one line off at the end of the file.

We can then examine the values of the series with **PRINT**, looking for values which have ended up in the wrong series, or locating where a series gets off sequence.

2.10 File Handling Tips and Tricks

Can't Open a File?

If you have created a data file in RATS and are having trouble opening the file in another application, you need to make sure that RATS doesn't still have the file open. For example, if you are working in interactive mode and do:

```
open copy test.dat
copy(format=prn,org=columns) / x y z
```

and then immediately try to open TEST.DAT in another application while RATS is still running, you will probably get an error message. And, if you view a directory listing of TEST.DAT using Windows Explorer or a similar utility, it will appear as a zero-byte file. That's because RATS still has the file open. In order to access the file in another application, just close the file by issuing the command:

```
close copy
```

or else by quitting the RATS application.

Similarly, RATS may not be able to read data from a file that is currently open in another application.

File Units

In RATS, you open a file by associating a file name with an input/output "unit." For example, the statement:

```
open data myfile.txt
```

simply associates the filename "myfile.txt" with the DATA unit, which is one of the reserved unit names in RATS. Every RATS instruction that can read or write data has a UNIT option which allows you to specify the source or destination unit for that operation.

Fortunately, this business of file unit names is generally transparent to the user, because all of the relevant instructions have default settings for the UNIT option. For example, the **DATA** instruction applies to the "DATA" unit by default. So, you can read data from a file by simply doing something like:

```
open data sample.rat
data(format=rats)
```

Because DATA is the default unit for the **DATA** instruction, it will automatically read the data from the SAMPLE.RAT file, which has been associated with the DATA unit by the **OPEN** command. If no file has been associated with the DATA unit, RATS will prompt you for a filename.

In some cases, however, you may find it helpful to make use of the UNIT option, particularly when you want to have several data files or output files open simultaneously. This is made easy by the fact that RATS lets you define your own unit names.

For example, suppose you want to read data from two different files in your program. You can either:

- Use an **OPEN DATA** command to associate the first file with the DATA unit, and then read the data in using **DATA**. Then, repeat this procedure for the second file:

```
open data first.rat
data(format=rats) / x
open data second.rat
data(format=rats) / y
```

- Or, you can define your own custom unit names, and associate one file with each name. You can then read from the files in any order by specifying the appropriate unit name. For example:

```
open data1 first.rat
open data2 second.rat
data(format=rats,unit=data2) / y
data(format=rats,unit=data1) / x
```

  Because you are using different unit names for each file, both files remain open, so you can go back and read from either one without having to repeat the **OPEN** command.

3. Graphics

RATS offers four main graphing instructions: **GRAPH** for time-series graphs, **SCATTER** for x-y graphs, **GCONTOUR** for creating contour graphs, and **GBOX** for creating box plots. All of these instructions offer a very large collection of options, allowing you to create publication-quality graphs easily.

In addition to a variety of basic graph types (such as line graphs and bar graphs), RATS offers a number of special effects, such as overlay graphs (displaying data in two different forms or with two different scales in a single graph) and arrays of smaller graphs.

You can also define graphics style sheets, which allow you to customize various attributes of your graphs, including colors, patterns, line thicknesses, and more.

Finally, graphs created can be exported in several well-known formats (such as PostScript or PDF) for placement in documents.

3.1 Graphics

The section provides general information on working with graphs in RATS. We will cover the instructions used to generate graphs, as well as the parts of the RATS interface which deal with graphs, including exporting graphs for use in other applications.

Overview of the Graphics Instructions and Wizards

RATS has four main instructions for creating graphs:

- **GRAPH** creates time series plots, or, more generally, plots of a series which is arranged as a sequence. The *Data/Graphics—Graph* wizard generates a **GRAPH** instruction. You can produce line graphs, filled line graphs, several types of bar graphs, and high-low-close graphs.
- **SCATTER** creates scatter (x vs. y) plots. In addition to series vs. series, it does graphs where the x axis is a grid on the real line, either equally or unequally spaced. The *Data/Graphics—Scatter(X-Y)* wizard generates a **SCATTER** instruction.
- **GCONTOUR** (Section 3.13) produces contour plots.
- **GBOX** (Section 3.14) produces box (or "box and whisker") plots.

These are very flexible instructions which allow you to create a wide variety of graphs. The auxiliary instructions **SPGRAPH**, **GRTEXT**, and **GRPARM** further enhance the abilities. With RATS, you can:

- use color, dashed line patterns, or shades of gray to distinguish series. RATS can automatically translate color into patterns or grayscale for output to black and white printers.
- generate a graph with series having different scales (which we refer to as "overlay graphs") using a single **GRAPH** or **SCATTER** instruction (Section 3.7). The overlay options also allow you to mix different styles on a single graph. For example, you could plot one series as a line and another using a bar-graph style.
- create pages of equally sized "postage-stamp" graphs. This is done using **SPGRAPH** (Section 3.8).
- add explanatory text within the graph, done with **GRTEXT** (Section 3.9).
- use graph style sheets to set the color and thickness of lines, choose from a range of fill patterns for bar graphs, and more (Section 3.16).
- change fonts and font sizes, done with **GRPARM** (Section 3.18)

The **GRAPH** and **SCATTER** instructions were introduced in the tutorial in Chapter 1. For complete details on these and the other instructions listed above, see the remainder of this chapter and the relevant sections in the help or *Reference Manual*.

3.2 Working with Graph Windows

Resizing Graph Windows, Setting Proportions

We already briefly discussed Graph Windows in Section 1.5.5. As mentioned there, when you resize a Graph Window, the contents resize with it. This is the standard behavior that you would get when you insert a graph into a document. However, the windows that you see on the screen are not (in general) in the proportions used if you export or copy the graph to insert into a publication. Instead, RATS does that in a roughly "golden ratio" with the width being 1.5 times the height. If you paste that into a document in standard orientation, the graph will generally cover about 1/3 of the page.

You can force RATS to use different proportions by first resizing the window to get the appearance that you want, then using the FIX toolbar icon. If you resize the window further, you'll see that the graph's shape doesn't change, just the size. That shape will be maintained if you copy, print, or export the graph.

Color or Black and White

RATS can render a graph either in color, or in black and white. Color is almost always the easiest to view on the screen, so that's the standard way that graphs are shown, but most (print) publications still require graphs in black and white. If you need a grayscale graph, click on the toolbar icon:  . If you copy or export the Graph Window, whether that is done in color or black and white will depend upon how you are showing the graph at the time. If you want to change back to color, just click on the same toolbar icon, which will now be in color. Grayscale graphs can be hard to read if you have lines that cross many times. If you have such a graph, you may need to either change the style numbers (page Int–130), or define your own style sheets (page Int–149).

Saving and Opening Graphs

If you want to save a graph so you can reload it later, you need to save it in RGF format (RATS Graphics Format). To open a saved RATS graph, select *File—Open...*, choose "RATS Graphics" from the list of file types, and then open the desired graph file. This will show the graph in a new Graph Window, exactly as if it has just been generated by the program.

RGF is designed for *temporary* storage of graphs. Its specification changes as we add features to our graphics system, so you should only open files with the same version of RATS that created them. It is, however, portable across systems, so a file can be shared by Windows, Mac and UNIX users as long as they are running the same version number of RATS.

Printing

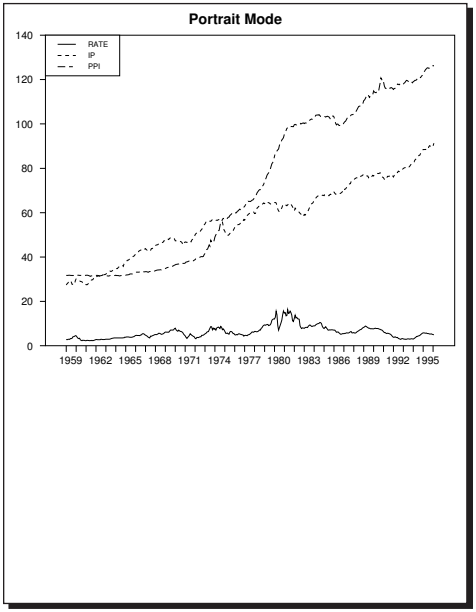
To print a Graph Window, do the following:

1. Make sure the desired graph window is active.
2. Select *File—Print*, or click on the toolbar icon.

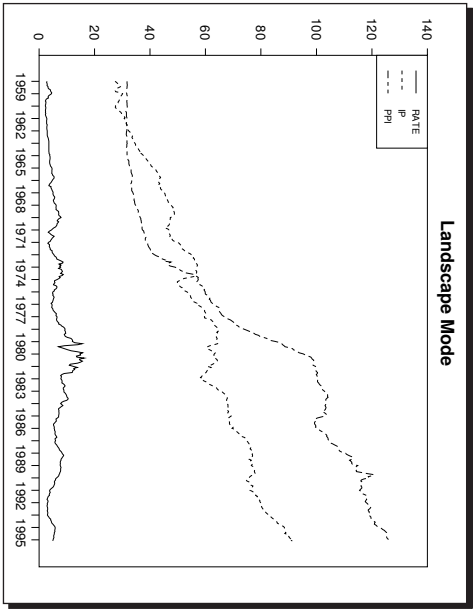
If you have a black and white printer, your graph will print with colors replaced by the black and white patterns, whether or not that is showing in your window. If it's a color-capable printer, then the graph will print in color if you're showing the graph in color, or in black and white if that's what you're showing.

Printers will allow you to choose between portrait and landscape mode. The results of the two choices are shown below. When you're printing a graph, you're more likely to use landscape, since it uses the full page. Portrait generally will only give a partial page since the graph itself is wider than it is tall, but the paper, in that orientation, is the reverse.

Portrait:



Landscape:



3.3 Preparing for Publication

You have three ways to include RATS graphs in a publication:

- Export the graph in one of the standard graphics file formats and then import the graph into your word-processor or page-layout program.
- Use *Edit—Copy* to copy a graph to the system clipboard, and then paste it into your word-processing application.
- Print a copy of your graph directly from RATS (for instance, as a PDF file) and include this in your publication.

The first two are obviously more flexible, since most word processors will allow you to size and shape the graph to fit your document.

Make sure that you have the desired settings for the proportions and color as discussed on page Int–127. Those will be applied when exporting or copying the graph.

Exporting/Saving

To export (save) to disk a graph that you have already generated, you should:

1. Make sure the desired graph window is active.
2. Choose the *File—Export* or *File—Save As* operation from the menu or the  toolbar icon. Select the desired format from the drop-down list.

Another option is to use the **GSAVE** instruction *before* generating your graphs to have RATS save them automatically. See page Int–153 for details.

PostScript is the preferred format for saving graphs because it gives the most accurate translation of the graph, and works very well if you will be producing your final output as a PDF file or on a PostScript printer. For PostScript, you can choose from two orientations, Portrait and Landscape. When you place the graph file on a page, these will have the same orientations shown on the previous page for printing. Portrait is the standard here, and is what we use in all the graphs in this manual other than samples of landscape.

PDF format should also work very well for any applications that support importing PDF files. WMF (Windows Metafile) is available on Windows, but might not be acceptable for all publications, as it's specific to Windows. PNG is a portable "bit-map" format, which also is often not acceptable for publication because it doesn't resize well, but is a good choice for (for instance) a web page.

Copying and Pasting

To copy a graph to the clipboard, make sure the desired graph window is active and then choose the *Edit—Copy* operation from the menu or click on the  toolbar icon.

Windows versions copy both bitmap and Windows metafile versions of the graph to the clipboard, while Macintosh versions copy in PDF format. To paste the graph, switch to the other application and select the *Paste* or *Paste Special...* operation from the *Edit* menu (*Paste Special...* allows you to select the format that is pasted).

3.4 Graph Styles and Style Numbers

Selecting Colors, Lines, and Patterns

RATS graphs data using lines, fill patterns, or symbols, depending upon the settings you choose for the STYLE and OVERLAY options: for instance, the "painted" styles like bar, stacked bar, and polygonal all use fill patterns. For each category (line, fill, symbol) RATS supports up to thirty different styles for color mode, and a corresponding thirty for black and white mode.

Styles are identified and selected by number, from 1 to 30, plus style 0 which is used by the SHADE option (Section 3.11). The default styles used by RATS have been chosen to be fairly easily distinguishable roughly for style numbers 1 through 10. You can use Graph Style Sheets (Section 3.16) to create your own definitions for the style numbers.

By default, RATS uses style number one for the first series being graphed (the series listed on the first supplementary card), style number two for the second series, and so on. You can use the *stylenum* parameter on the supplementary cards of **GRAPH** and **SCATTER** to select different styles. For example:

```
graph 2
# x1 / 4
# x2 / 2
```

selects style number four for the X1 series and style number two for the X2 series. Style numbers above 30 simply wrap around so 41 is the same as 11.

Color and Black and White Styles

RATS normally displays graphs in color, using different colors to distinguish series. In color mode, with the default style definitions, lines are drawn as solid lines in different colors, fill patterns as solid fills in different colors, and symbols using the same symbol in different colors. These usually are easy to distinguish.

RATS switches to the corresponding black and white (grayscale) styles if you: print to a black and white printer; use the () toolbar button; or use the PATTERNS option on the graphing instruction. With the default styles, lines are drawn in black using different dash patterns to distinguish series. Fills are drawn using different "hatch" patterns, and different symbols (in black) are used.

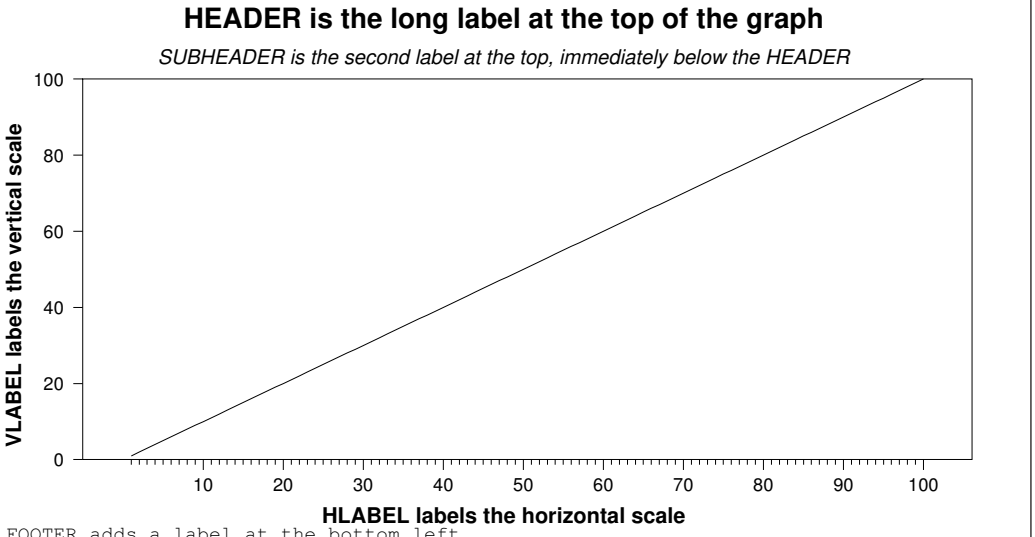
If you have a graph with several intertwined lines, it can be hard to see the detail of the lines when you can't use color. The solid black (style 1) isn't the problem; it's the dashed lines which lose detail and may seem to disappear at times. If you have one series which is fairly smooth and one which is more volatile, list the volatile one first so it gets style 1. If that won't work, you may need to use thicker solid grayscale lines—styles 8, 9, 10 and 11 are defined that way (this may look a bit odd in color). However, it may not be possible to make the graph work if it's too busy, and you may need to think of a different way to show the data.

3.5 Labeling Graphs

To help you produce publication-quality graphs, RATS gives you a great deal of flexibility in labeling graphs. We look at these in the paragraphs below.

Major Labels

The four main graphing instructions, as well as **SPGRAPH**, all offer HEADER, FOOTER, SUBHEADER, HLABEL, and VLABEL options. Their functions are shown in the graph below. In addition, in its title, the Graph Window will assume the name of the HEADER or the FOOTER (HEADER if you use both).



Other Labels

When the main labels are used on **SPGRAPH** (Section 3.8), they are placed at these locations relative to the full graph page, and not to any single graph on it. You can use the XLABELS and YLABELS options on **SPGRAPH** to label the rows and columns of the graph "matrix", or you can add the standard label types to the individual graphics instructions, which will add (smaller) labeling in the zone assigned to that graph.

The **GRTEXT** instruction (Section 3.9) is a very flexible tool that allows you to add strings of text anywhere inside or in the margins of a graph. You can select the exact location, font, size, and style of the text string.

SCATTER and **GCONTOUR** (Section 3.13) offer an XLABELS option to supply your own list of labels for the X-axis tick marks, while **GBOX** (Section 3.14) provides a LABELS option for labeling each of the series in the graph.

Supplying Labels

You can supply either a literal string of text enclosed in quotes, or a LABEL or STRING variable defined ahead of time (some options require a VECTOR of STRINGS). For example, the following **GRAPH** commands produce the same result:

```
graph(header="US Real GDP")
# rgdp

compute hlab = "US Real GDP"
graph(header=hlab)
# rgdp
```

Constructing Strings

Quoted strings are fine for single graphs, but if you need to generate many standardized graphs, it can be rather tedious to duplicate code and hand edit it. Fortunately, all these options will permit you to use a STRING type variable instead of a quoted string. These can be constructed, or input (see below).

```
compute header="Coherence of "+%l(x1)+" and "+%l(x2)
graph(header=header,noticks)
# coherenc

compute header="Autocorrelations "+diffs+" x "+sdiffs
graph(header=header,max=1.0,min=-1.0,style=bargraph)
# corrs
```

Note: %L(series) is a special function which returns the label of a series.

Reading Strings From a File

If you have a standardized program, but the headers or labels can't be constructed easily as they have no obvious pattern, make up a separate file with a list of headers (one per line) and read them in using the instruction **READ**:

```
open data labels.rgf
declare vector[string] header(230)
read header
do i=1,230
  ...
  graph(header=header(i),...)
  #  ...
end do i
```

Line Breaks in Strings

You can use two backward slash characters (\\) to insert line breaks in strings used in graphs. See the **GRTEXT** examples in Section 3.9.

3.6 Keys (Legends)

The KEY option on **GRAPH** and **SCATTER** allows you to add a key (legend) at a choice of several locations. Four of these are outside the graph box (KEY=BELOW, ABOVE, LEFT or RIGHT). Four of these are inside the graph box (UPLEFT, LOLEFT, UPRIGHT, LORIGHT). This second group doesn't take space away from the graph itself, but you have to be careful to choose a location (if it exists) which doesn't interfere with the graph. There is also KEY=ATTACHED, which puts the graph labels on the graph itself at locations where the labeling will be as clear as possible. This is available only for a few types of graphs (line, step, and symbols time series graphs).

By default, RATS uses the series names for the key labels. If these are cryptic database names, you can provide your own labels on the graph using the KLABELS option. KBOX controls whether a box is drawn around the key, while KHEIGHT and KWIDTH allow you to control the size and shape of the key. If you only want to see text in the key box (perhaps you don't need to show a sample line because you are graphing only one series), use NOKSAMPLE.

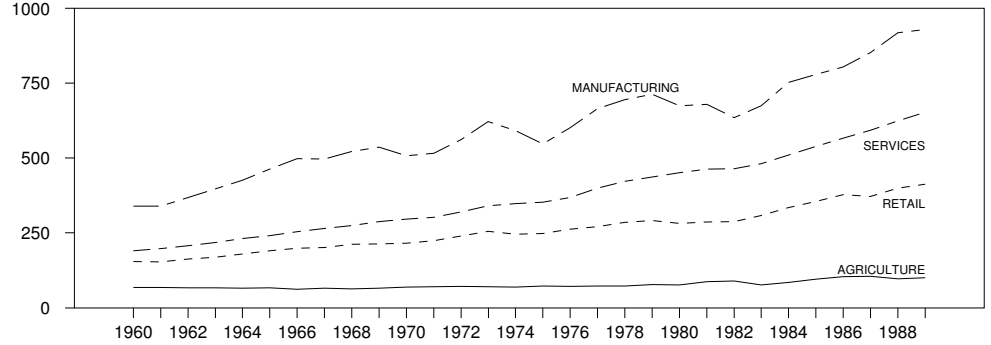
This example uses descriptive titles rather than the original series names for the key. The key will be placed below the graph.

```
graph(key=below,klabels=||"30 Year Bonds","3 Month Bills"||) 2
# ftb30y 1998:1 1999:12
# ftbs3  1998:1 1999:12
```

This uses KEY=ATTACHED, and also KLABELS, using in-line matrix notation (page Int–90)
to provide the VECTOR of STRINGS.

```
graph(style=line,key=attached,header="Figure 3.15",$
klabels=||"AGRICULTURE","RETAIL","SERVICES","MANUFACTURING"||) 4
# ga8gff
# ga8gr
# ga8gs
# ga8gm
```

Figure 3.15



3.7 Overlay (Two-Scale or Two-Style) Graphs

The OVERLAY option makes it easy to do a **GRAPH** or **SCATTER** plot with two different vertical scales. It also allows you to combine two different styles. For example, you can graph some series using a line style and others using a bar graph style.

By default, the scale for the first set of series appears on the left side of the graph, while the scale for the second set of series appears on the right side of the graph. You can use the OVSAMESCALE option (along with OVERLAY) if you want to use the same scale for both axes, but with different styles for different series.

Creating Overlay Graphs

Here is the basic procedure for creating an overlay graph:

- Use the OVERLAY option on your **GRAPH** or **SCATTER** instruction to tell RATS to do an overlay graph. OVERLAY offers the same choices as STYLE, but it sets the style used for the overlaying (right-scale) series. The STYLE option itself applies only to the left-scale series.

- If you are only graphing two series (or two pairs of series for **SCATTER**), you don't need to do anything else. Just list the series on separate supplementary cards, as you would for an ordinary graph. The second series (or pair) are graphed using the OVERLAY style and scale.

- If graphing more than two series, use the OVCOUNT option to tell RATS how many series should be graphed using the right-side (overlay) scale. For OVCOUNT=*n*, RATS graphs the last *n* series (or pairs) listed using the right-side scale.

Notes

When you use the OVERLAY option, RATS ignores the AXIS, EXTEND, and SCALE options on **GRAPH** and the AXIS, EXTEND and VSCALE options on **SCATTER**. This is because running an axis line or extending grid lines across the graph is likely to be very distracting, as they will apply to only one of the two scales. All other options, such as HEADER, HLABEL, and GRID, work normally.

You can use the OVLABEL option to label the right-side scale. Use the standard VLABEL option if you want to label the left-side scale.

The OMAX and OMIN options set the maximum and minimum values for the right-side scale. They function just like the MAX and MIN options, which will control only the left-side scale in a two-scale graph.

If you want to use one of the "painted" styles, such as POLYGONAL or BARGRAPH, you should probably use that as the STYLE (left-scale), as these are drawn first. If you use a paint style for the overlaying series, they may cover up the first block of series.

You can use the OVRANGE option to offset the two scales so they aren't directly across from each other. OVRANGE=*fraction* gives each scale the given fraction of the verti-

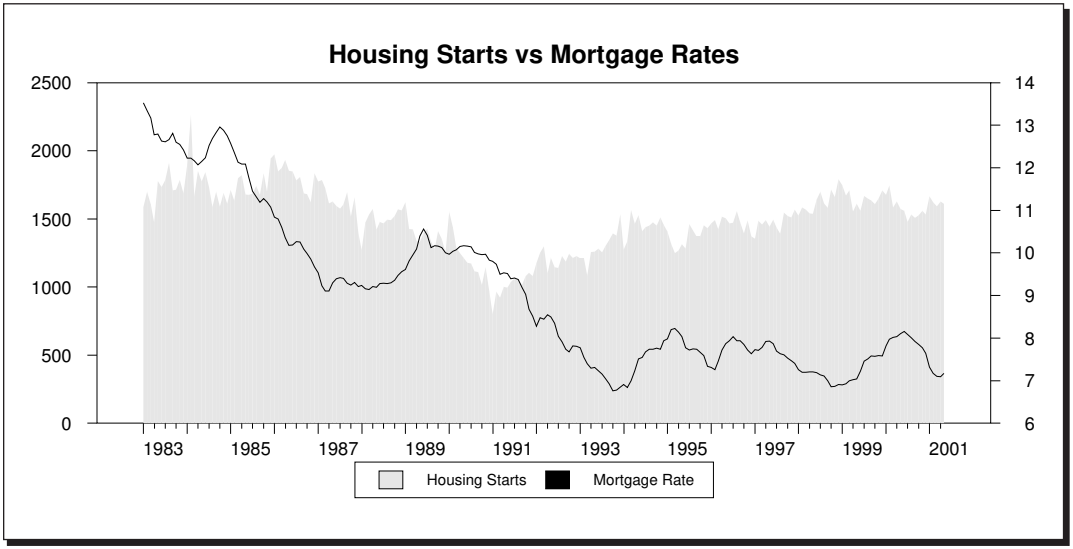
cal size (*fraction* should be in the range 0.5 to 1.0). OVRANGE=.6 for instance will have them overlap only slightly (the left scale getting the bottom 60% and the right getting the top 60%).

Example

This simple example (from GRAPHOVERLAY.RPF) graphs two series. The series HSF (housing starts) is graphed using the POLYGONAL style, and its scale appears on the left side of the graph. The series FCME (mortgage rates) is graphed as a LINE, and uses the right-side scale.

You will probably need to experiment with the *stylenum* parameters to get overlay graphs to look good. Here, we have used fill pattern 2 (medium gray) for the HST series, and line type 1 (solid black) for the FCME series. Without these, HST would be graphed using a solid black pattern (pattern 1), and FCME, which would be a dashed line, would not be as visible. We also used KLABEL to supply custom key labels.

```
open data haversample.rat
calendar(m) 1972
data(format=rats) 1972:1 1988:12 fcme hst
graph(key=below,header="Housing Starts vs Mortgage Rates",$
  klabel=||"Housing Starts","Mortgage Rate"||,$
  style=polygonal,min=0.0,overlay=line) 2
# hst  / 2
# fcme / 1
```



3.8 SPGRAPH—Multiple Graphs on a Page

You can put two or more individual graphs on a single picture using **SPGRAPH** with the HFIELDS (horizontal fields) and VFIELDS (vertical fields) options. For example:

```
spgraph(vfields=2,hfields=2)
   four graph, scatter, or gcontour instructions
spgraph(done)
```

This **SPGRAPH** divides the available space in half in each direction, creating four zones of equal size. The graphs (by default) fill the zones by rows beginning at the top left. Note, however, that the characters on the small graphs are sized to be proportional to what they would be on a full page; that is, each quarter graph will look like a reduced version of the same graph done by itself. The labeling (particularly on the scales) can sometimes get too small to be readable, particularly if you get to four or more graphs in either direction. Depending upon the situation, you may want to increase the character sizes using **GRPARM**, or you might want to suppress the axis labeling altogether. To do this, use NOTICKS and SCALE=NONE on **GRAPH** and VSCALE=NONE and HSCALE=NONE on **SCATTER** or **GCONTOUR**.

You also need to be careful that you do not leave false impressions by allowing each graph to find its own range. This could inflate the appearance of what may be, in reality, some very small effects. Most of the work in doing a matrix of graphs is, in fact, just preparation.

The following (from example SPGRAPH.RPF) graphs the exchange rate of the dollar versus four foreign currencies. These are first given a common reference by converting them to percentage appreciation of dollars per foreign unit since the start of the sample. As all but the UK series (GBRUSXSR) are originally stated in foreign currency per dollar, they have to be flipped. We then use **TABLE** to get the maximum and minimum values attained across *all* the countries, and use these as the MAX and MIN on each **GRAPH**. This ensures that the very modest movements relative to the Canadian dollar aren't exaggerated relative to the much larger movement versus the yen.

The **LABELS** instruction is used to give a more readable label to the series, naming them by their country instead of the database coding. Their new labels are used in the **GRAPH** instructions within the loop on the HEADER option.

```
cal(m) 1996:1
open data oecdsample.rat
data(format=rats) 1996:1 1998:12 canusxsr frausxsr $
   jpnusxsr gbrusxsr
```

Flip to dollars per foreign unit and assign descriptive labels

```
set canusxsr = 1.0/canusxsr
set frausxsr = 1.0/frausxsr
set jpnusxsr = 1.0/jpnusxsr
labels canusxsr frausxsr jpnusxsr gbrusxsr
# "Canada" "France" "Japan" "UK"
```

Convert to percent appreciation

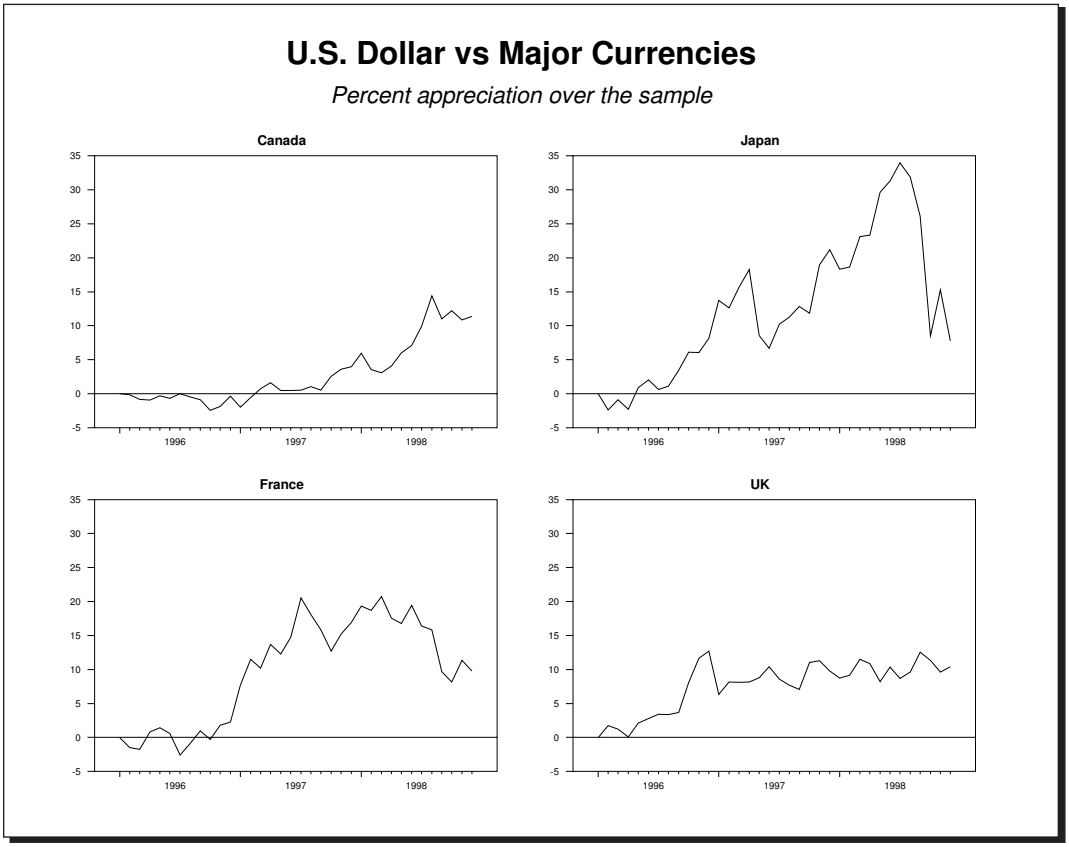
```
dofor i = canusxsr frausxsr jpnusxsr gbrusxsr
   compute base=([series]i)(1996:1)
   set i = 100*(i{0}/base - 1.0)
end dofor i
```

Use TABLE to get the maximum and minimum across all series

```
table(noprint) / canusxsr frausxsr jpnusxsr gbrusxsr
```

Set up the SPGRAPH for a 2×2 matrix

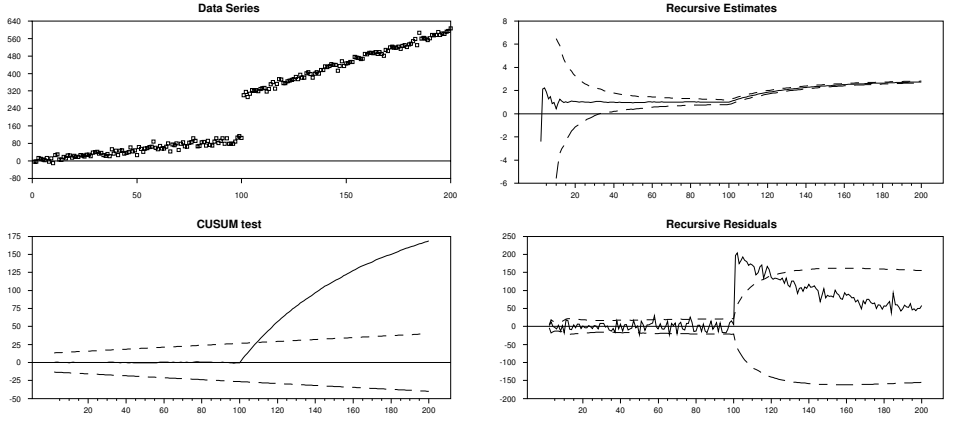
```
spgraph(vfields=2,hfields=2,$
  header="U.S. Dollar vs Major Currencies",$
  subheader="Percent appreciation over the sample")
dofor i = canusxsr frausxsr jpnusxsr gbrusxsr
   graph(max=%maximum,min=%minimum,header=%l(i)) 1
   # i
end dofor
spgraph(done)
```



Here is another example using **SPGRAPH**. This is taken from an example in Diebold (2004). The example program file (DIEB3P235.RPF) is included with RATS.

```
spgraph(hfields=2,vfields=2,$
header="Figure 9.17 Recursive Analysis: Breaking Parameter Model")
scatter
# x y3
rls(cohist=cohist,sehist=sehist,sighist=sighist,csum=cusum) $
  y3 / resids
# x
set upperco = cohist(1)+sehist(1)*2.0
set lowerco = cohist(1)-sehist(1)*2.0
graph(header="Recursive Estimates") 3
# cohist(1)
# upperco 10 *
# lowerco 10 *
set upperres = 2.0*sighist
set lowerres = -2.0*sighist
graph(header="Recursive Residuals") 3
# resids
# upperres
# lowerres
set cusum = cusum/sqrt(%seesq)
set upper5 startr end = .948*sqrt(%ndf)*(1+2.0*(t-startr)/%ndf)
set lower5 startr end = -upper5
graph(header="CUSUM test") 3
# cusum
# upper5
# lower5
spgraph(done)
```

Figure 9.17 Recursive Analysis: Breaking Parameter Model

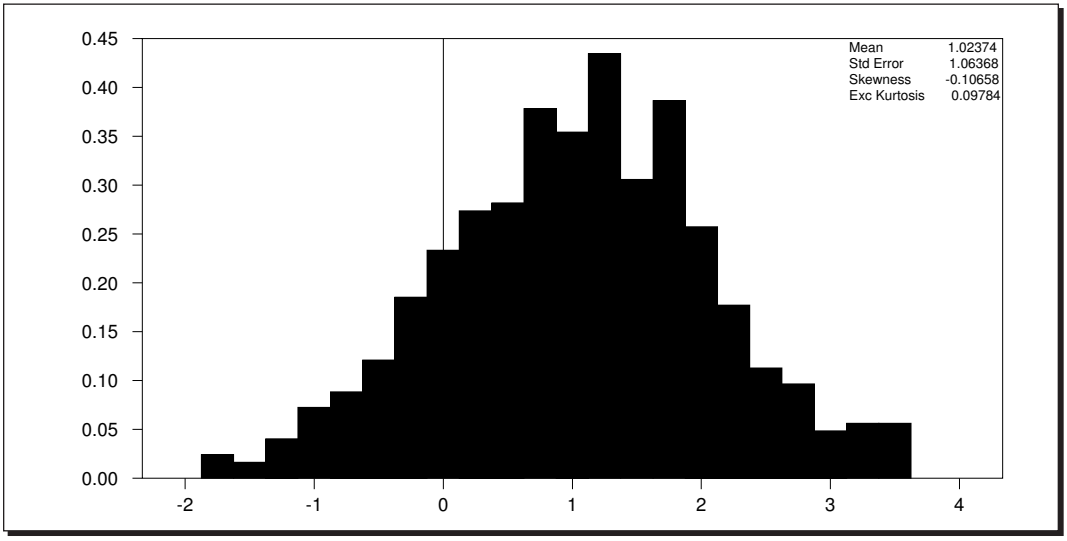


3.9 GRTEXT—Adding Text to Graphs

The **GRTEXT** command allows you to add text anywhere inside a graph image or in the margins of the graph, with options allowing you to specify the position of the text, the alignment, and the font style. To use **GRTEXT**, you need to enclose the graphing and **GRTEXT** command(s) inside an **SPGRAPH** block.

The code excerpt below is from the HISTOGRAM.SRC procedure included with RATS, which generates histogram plots. We use **GRTEXT** to include computed statistics on the graph. Note the use of "\\" symbols to put line breaks into a single string—this is often easier than using multiple **GRTEXT** commands to achieve the same effect.

```
spgraph
 display(store=s) "Mean" %mean "\\Std Error" sqrt(%variance) $
  "\\Skewness" %skewness "\\Exc Kurtosis" %kurtosis
 if distrib==2 {
   set nx = 1.0/sqrt(%variance)*$
            %density((fx-%mean)/sqrt(%variance))
   scatter(style=bargraph,overlay=line,ovsamescale) 2
   # fx dx
   # fx nx
 }
 else {
   scatter(style=bargraph) 1
   # fx dx
 }
 grtext(position=upright) s
spgraph(done)
```

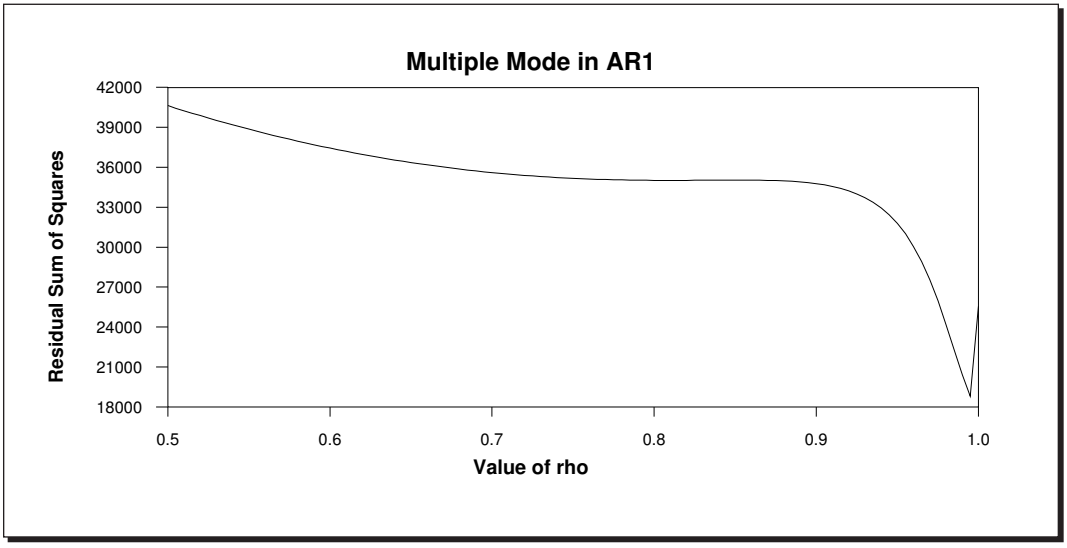


3.10 Graphing Functions

If you want to graph a function $y = f(x)$, use the instruction **SCATTER**. Create a grid series with x values and a corresponding series of y values. **SCATTER** with the option STYLE=LINE will create the graph. You can also use other style choices, such as BAR and POLYGONAL. In general, you should not need more than about 100 grid points on the x-axis for a smooth function. Note that the LINE, BAR and POLYGONAL styles on **SCATTER** only work correctly if the x series is sorted into increasing order.

The following computes a series of **AR1** regressions for a grid of values for ρ between .5 and 1.0 and graphs the residual sum of squares against ρ. This is from example file GRAPHFUNCTION.RPF.

```
@GridSeries(from=.5,to=1.0,size=.005,pts=gpts) rhos
set rss 1 gpts = 0.0
do i=1,gpts
   ar1(noprint,rho=rhos(i)) invest 1950:1 1985:4
   # constant ydiff{1} gnp rate{4}
   compute rss(i)=%rss
end do i
scatter(style=lines,vlabel="Residual Sum of Squares",$
   hlabel="Value of rho",header="Multiple Mode in AR1")
# rhos rss
```



3.11 Highlighting Entries

The options GRID and SHADING permit you to draw attention to particular entries. GRID does this by drawing a vertical line from top to bottom of the graph at specific entries. SHADING paints a shaded box over any set of consecutive non-zero entries.

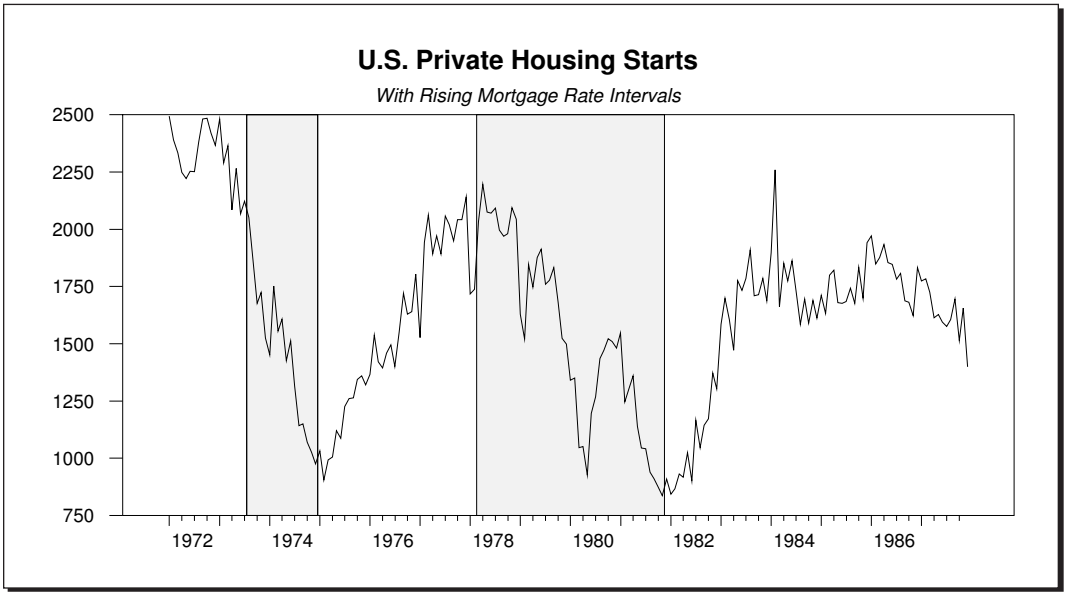
Shading or grid lines are done before any of the series. If the graph uses one of the painted styles (POLYGONAL, BAR, STACKED or OVERLAP), it may cover the highlighting, so be careful if you want to use those.

SCATTER and **GCONTOUR** have separate options for grids or shading in each direction. Note that, unlike **GRAPH**, where the highlighting information is provided in data series, for **SCATTER** the grid lines are provided using a VECTOR in the HGRID and VGRID options and the shading zones are in a $N\times 2$ RECTANGULAR for HSHADE and VSHADE.

```
set raterise = t>=1973:8.and.t<=1974:12.or.$
               t>=1978:3.and.t<=1981:11
graph(shading=raterise,header="U.S. Private Housing Starts",$
  subheader="With Rising Mortgage Rate Intervals") 1
# hsf 1972:1 1987:12
```

is from GRAPHOVERLAY.RPF. It puts shading over 1973:8 to 1974:12 and 1978:3 to 1981:11.

You *might* find that you need to darken the shading for publication. To do that, redefine FILL_BW_0 (see page Int–149). The default value is .90; if that's too light, you might try .85.

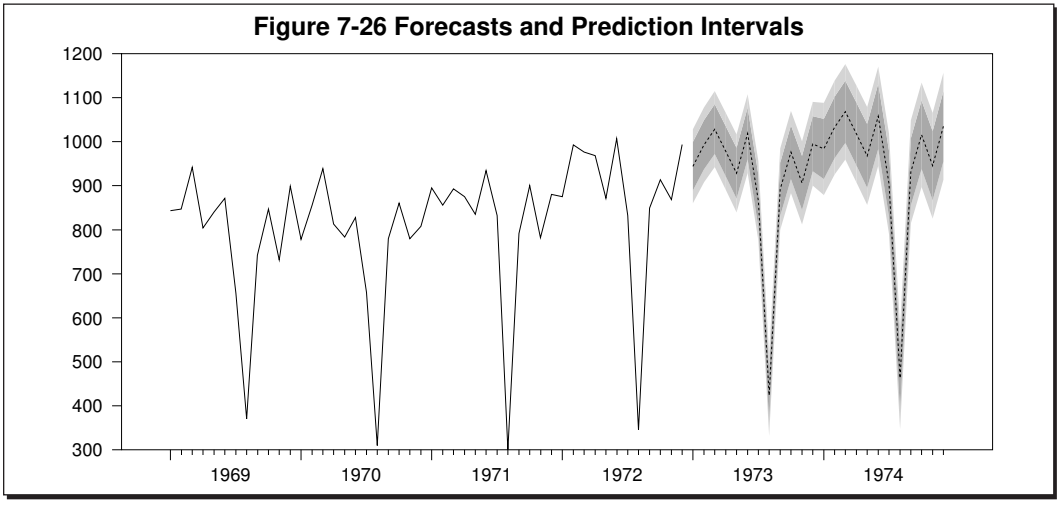


3.12 Fan Charts

Fan charts provide a useful way to display confidence bands, particularly with multiple levels. The confidence bands are displayed in shades of the same color, which are darkest nearest the center and growth fainter towards the outer bands. With the **GRAPH** instruction, it's usually best to do these as an overlay, with the point forecast being done as a line. Make sure that you use OVSAME with OVERLAY=FAN so the bands are located in the correct location.

This is from Makradakis, et. al (1998), part of textbook example MWH3P366.RPF. It computes point forecasts and upper and lower 80% and 95% confidence bands. The pre-forecast data and the point forecasts are done as line graphs; the confidence bands as an overlay. The order of listing of the series covered by the fan isn't important. At each point, the values are ordered from bottom to top. With four series, there are three bands. The center is done in a darker gray (for black and white) and the outer ones in a lighter shade.

```
uforecast(equation=weq,stderrs=stderrs) wfore 1973:1 1974:12
set lower95 1973:1 1974:12 = wfore+%invnormal(.025)*stderrs
set upper95 1973:1 1974:12 = wfore+%invnormal(.975)*stderrs
set lower80 1973:1 1974:12 = wfore+%invnormal(.1)*stderrs
set upper80 1973:1 1974:12 = wfore+%invnormal(.9)*stderrs
graph(footer="Figure 7-26 Forecasts and Prediction Intervals",$
  ovcount=4,overlay=fan,ovsame) 6
# writing 1969:1 1972:12
# wfore
# lower95
# lower80
# upper80
# upper95
```



3.13 GCONTOUR—Contour Graphs

The **GCONTOUR** instruction generates contour plots. Its design is quite different from **GRAPH** and **SCATTER**, because it needs the *x* and *y* values to be in the form of a grid, and the function values must be a matrix with dimensions size of *x* grid by size of *y* grid. This matrix is usually created using the instruction **EWISE**, which fills an array based on a function of the row and column numbers. See **EWISE** in the help or *Reference Manual* and Section 1.7 of the *User's Guide* for details.

The following is taken from the file GCONTOUR.RPF. This analyzes the log likelihood of a model with a break point in the mean as a function of the location of the break point (shown on the y axis) against the mean of the post break process (x axis).

Set up the grids for BREAKS (1,...,100) and MUS (.2,.4,...,20)

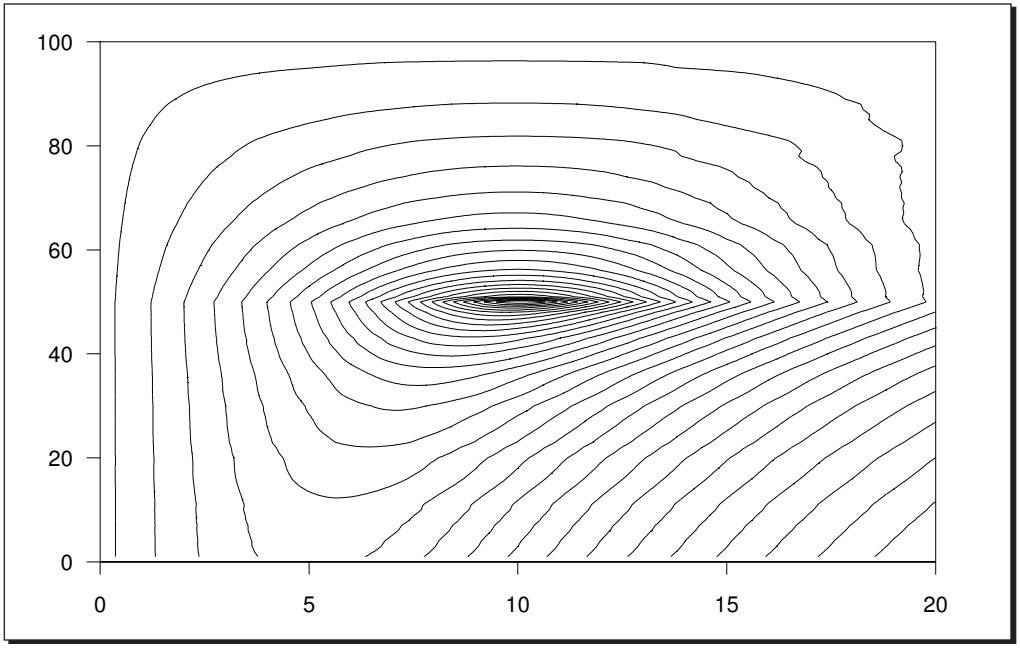
```
compute breaks=%seqa(1.0,1.0,100)
compute mus   =%seqa( .2, .2,100)
```

Generate the log likelihood for all the combinations of MUS *and BREAKS.*

```
dec rect f(100,100)
ewise f(i,j) = -50.0*log((x2-2*mus(i)*over(fix(breaks(j)))+$
               (100-breaks(j))*mus(i)^2)/100.0)
```

Do the contour graph with a grid line across the actual break (50)

```
gcontour(x=mus,y=breaks,f=f,vgrid=||50||)
```



3.14 GBOX—Box Plots

Box plots (also known as box and whisker plots) provide a quick way to examine some basic statistical properties of one or more data series, and they can give you a visual indication of how the observations in the data are distributed.

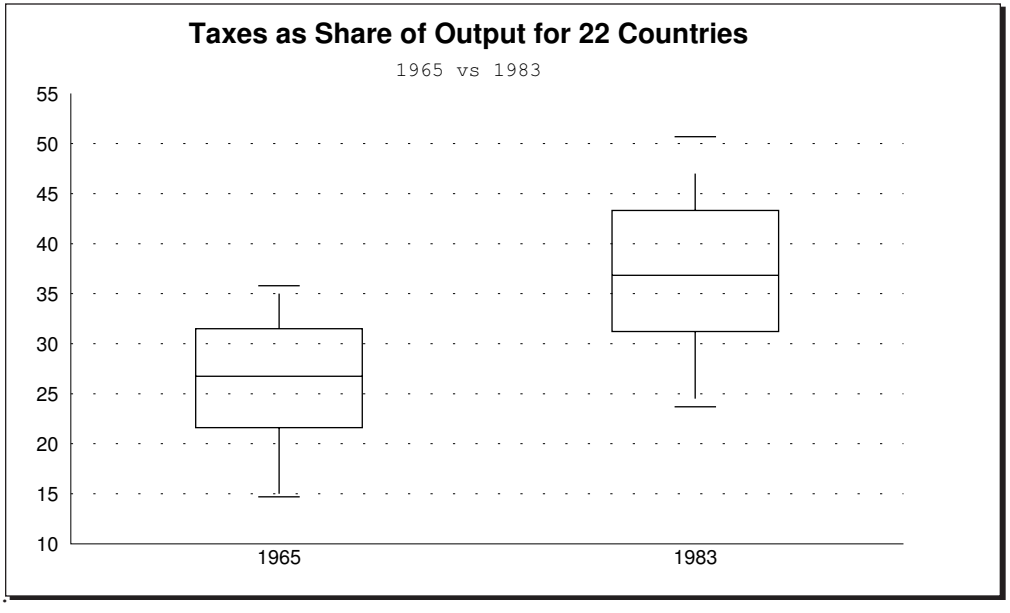
This simple example (from GRAPHBOXPLOT.RPF) draws box plots for two data series which list taxes as a share of total output for 22 countries in two different years. The first series contains the data for 1965, while the second series contains data for 1983:

```
open data CountryTaxData.xlsx
calendar(panelobs=22)
data(format=xlsx,org=columns) 1//1 1//22 Country Tax1965 Tax1983
gbox(header="Taxes as Share of Output for 22 Countries",$
     subheader="1965 vs 1983",labels=||"1965","1983"||,$
     frame=half,extend) 2
# tax1965
# tax1983
```

Here, we've used the LABELS option to tell **GBOX** to use "1965" and "1983" as the labels for the two series, rather than the series names. In this case, we use in-line matrix notation to provide the VECTOR of STRINGS. This could also be done by storing the labels into a variable of type VECTOR[STRING] ahead of time:

```
compute [vector[string]] gboxlab = ||"1965","1983"||
gbox(header="Taxes as Share of Output for 22 Countries",$
     subheader="1965 vs 1983",labels=gboxlab,$
     frame=half,extend) 2
```

etc.



3.15 Miscellaneous Graph Types

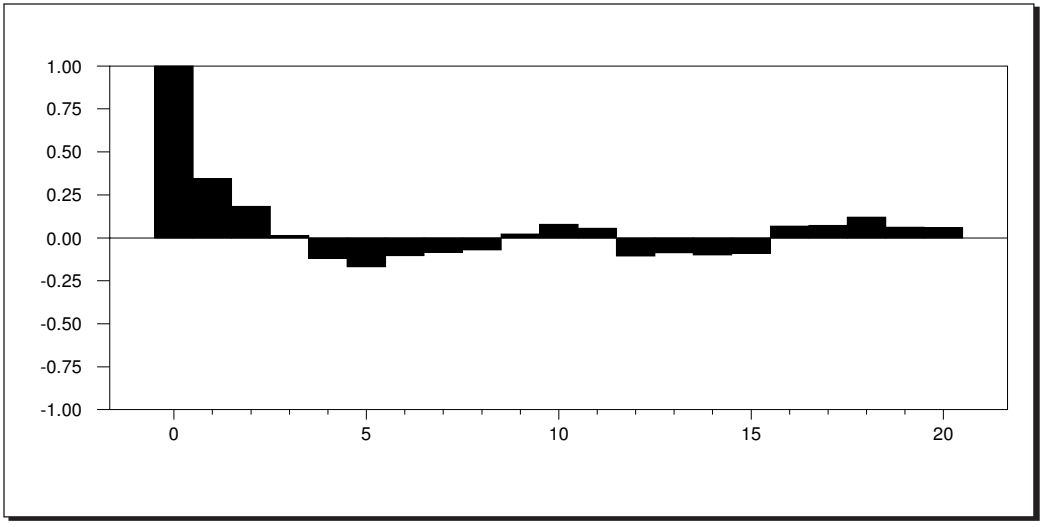
Autocorrelations

Autocorrelations and partial autocorrelations are bounded between −1.0 and 1.0. Graphing them without fixing the plot limits can produce confusion, because the plot ranges for different sets of correlations will probably be quite different. We have found the following set of options to be very helpful:

```
graph(number=0,style=bargraph,max=1.0,min=-1.0) 1
# corrs
```

NUMBER=0 causes the time axis to be labeled with 0,1,2,etc. STYLE=POLYGONAL also works well for correlations, and STYLE=SPIKE can be useful if you have to graph a large number of correlations. See Section 6.4.1 of the *User's Guide* for more details on computing and graphing correlations.

This shows a very basic graph of correlations. If you use any RATS procedures which compute and graph autocorrelations (such as the **@BJIDENT** procedure included with RATS), you are likely to run into some graphs that supply additional information. For instance, it's possible to highlight the correlations which are statistically significant using the SHADING option, which we demonstrate in another context on page Int–141.

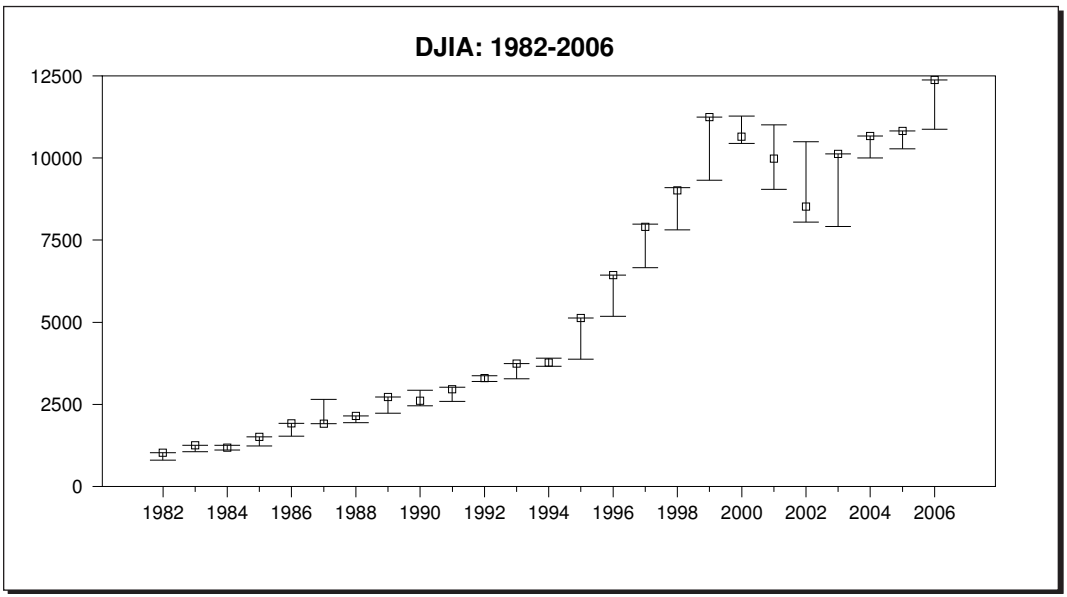


High-Low-Close Graphs

You can use the option STYLE=VERTICAL for high-low-close graphs and the like. At each entry, it draws a line connecting the highest and lowest values and puts hash marks at the locations of the series. If there are three or more series, it puts a filled circle on the first series so it can be picked out easily.

The example GRAPHHIGHLOW.RPF pulls monthly values of the Dow Jones Industrial Average into annual series. It uses three **DATA** instructions, one to get the maximum, one the minimum and one the final value during the year. Since each **DATA** instruction will reset SPDJI, the data are copied to a new series name after each. The "close" series should be listed on the first supplementary card so it gets tagged with a special symbol. For this graph, we look at results from 1982 through 2006.

```
open data haversample.rat
calendar(a) 1982:1
all 2006:1
data(format=rats,compact=max) / spdji
set djmax = spdji
data(format=rats,compact=min) / spdji
set djmin = spdji
data(format=rats,compact=last) / spdji
set djlast = spdji
graph(style=vertical,header="DJIA: 1982-2006") 3
# djlast
# djmin
# djmax
```



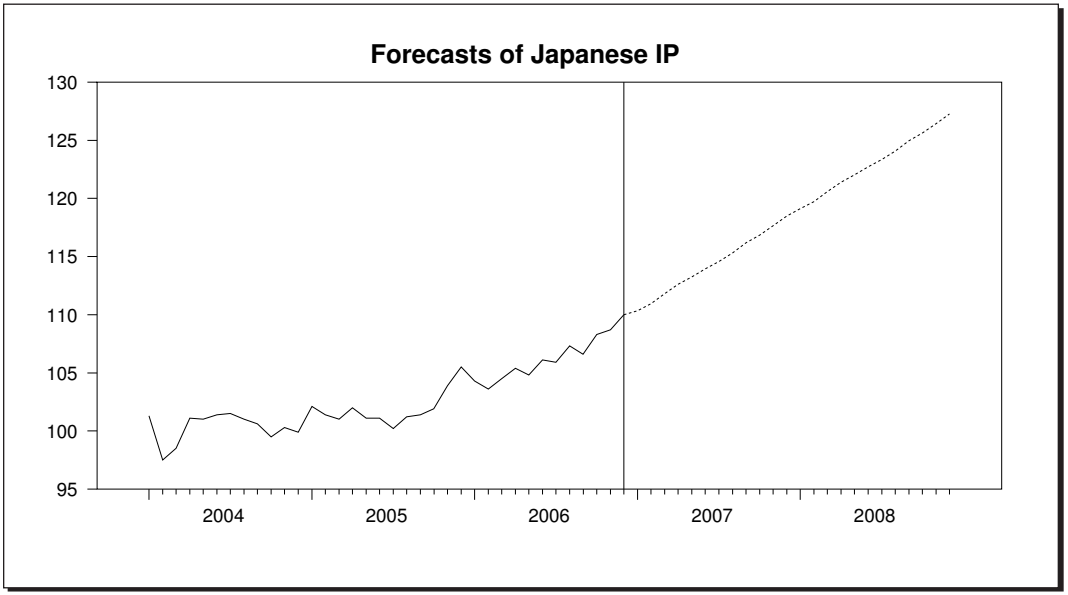
Forecasts

If you graph forecasts with the last few observations of historical data (as separate series), you will find there will be a break in the graph between the end of the historical data and the start of the forecasts. One way to improve the appearance is to add the final historical value to the beginning of the forecast series. That way, its line will connect smoothly with the historical series.

The following (from GRAPHFORECAST.RPF) uses **ESMOOTH** (exponential smoothing) to forecast from 2007:1 to 2008:12. The final actual data value from 2006:12 is added at that entry to the forecast series (JPNFORE). A grid line is added to the graph at 2006:12.

```
cal(m) 1960:1
open data oecdsample.rat
data(format=rats) 1960:1 2006:12 jpniptotrs

esmooth(trend=select,seasonal=select,$
     forecast=jpnfore,steps=24) jpniptotrs
set jpnfore 2006:12 2006:12 = jpniptotrs
graph(header="Forecasts of Japanese IP",grid=t==2006:12) 2
# jpniptotrs 2004:1  2006:12
# jpnfore    2006:12 2008:12
```



This is another example, taken from pages 349-360 of Diebold (2004). It includes upper and lower confidence bands, and shades the "forecast" area. The full example program (DIEB3P348.RPF) and associated procedure files are included with RATS.

Set dummy variable FOREZONE to 1 for 1995:1 and later:

```
set forezone * 2010:12 = t>=1995:1
```

Estimate the model through 1994:12

```
boxjenk(const,diffs=1,ar=1,define=diffeq) logyen * 1994:12 resids
```

Use UFORECAST to generate forecasts and standard errors:

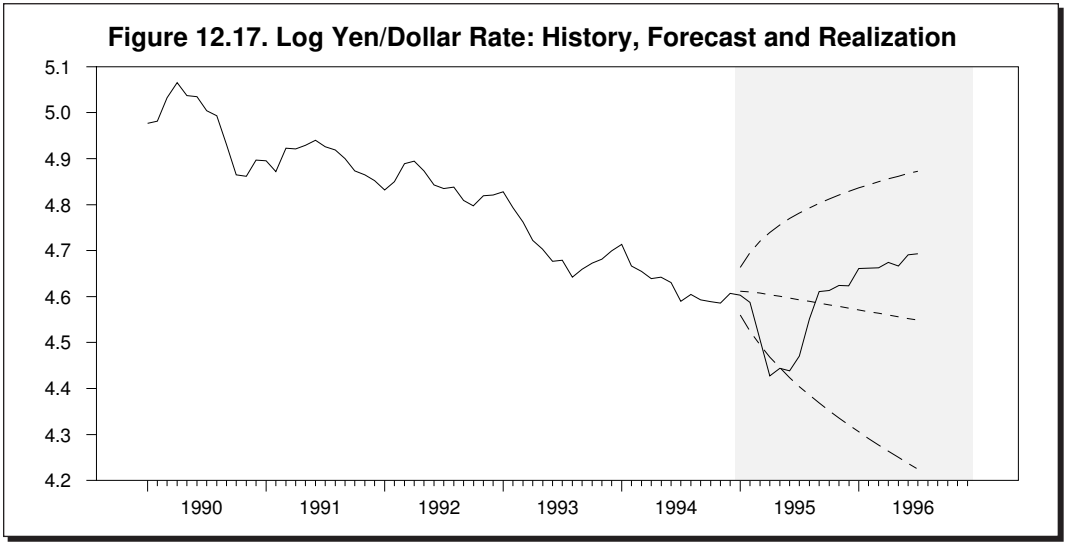
```
uforecast(equation=diffeq,stderrs=sefore) fyen 1995:1 1996:7
```

Generate upper and lower bound series from results:

```
set upper 1995:1 1996:7 = fyen+2.0*sefore
set lower 1995:1 1996:7 = fyen-2.0*sefore
```

Graph the actual values, forecasts, and upper and lower bounds. Shading is applied where FOREZONE is non-zero. The "stylenum" parameter is used to specify line style 3 for both the upper and lower series:

```
graph(header=$
"Figure 12.17. Log Yen/Dollar Rate: History, Forecast and Realization",$
shading=forezone) 4
# logyen 1990:1 1996:12
# fyen 1995:1 1996:7
# upper 1995:1 1996:7 3
# lower 1995:1 1996:7 3
```



3.16 Graph Style Definitions

The default styles are designed to work well in most circumstances, but you can also define your own custom styles to suit your preferences or organizational standards. You do this using Graph Style Definitions. RATS defines 30 different styles for color lines, grayscale (black and white) lines, color patterns, grayscale patterns, color symbols, and grayscale symbols. (If you have a graph which uses more than 30 styles, these repeat, so 31 is the same as 1). You can override any of those.

You have control over the following:

- For lines in color graphs: the pattern (solid line or one of seven dashed patterns), the line color and the thickness.
- For lines in grayscale graphs: the pattern, the grayscale level and the thickness.
- For fill patterns (as used in bar graphs and similar styles) in color graphs: the pattern, with seven patterns from which to choose (solid and six types of "hatch" patterns) and the color.
- For fill patterns in grayscale: the hatch pattern and the grayscale level.
- For symbols in color graphs: the shape (twelve choices), the color, and whether or not the symbol is filled or not filled.
- For symbols in grayscale graphs: the shape, the grayscale level, and whether or not the symbol is filled or not filled.

You're most likely to want to redefine the black and white styles, since those will be used most often in publications.

Style Definitions

A style definition will take one of the following forms, depending upon what you want to change.

```
LINE_COLOR_NN=pattern,color,thickness
LINE_BW_NN=pattern,gray,thickness

FILL_COLOR_NN=pattern,color
FILL_BW_NN=pattern,gray

SYMBOL_COLOR_NN=pattern,color,filled
SYMBOL_BW_NN=pattern,gray,filled
```

On the left side of the =, the first part of the definition specifies the type of representation you are defining (`LINE`, `FILL`, or `SYMBOL`). The second part (`COLOR` or `BW`) tells whether it is a color or black and white style. The third part (`NN`) is the style number that you're defining. It should be between 0 and 30. Style 0 is reserved for fill shading performed using the `SHADE` options, so if you want to adjust the pattern or gray level used for shading, define the `FILL_BW_0` style.

The style to be used is described using the following:

pattern	a number indicating the pattern choice—solid or dashed for lines, hatch pattern for fills, and symbol shape for symbols. See Pattern Definitions for the available choices.
color	is represented as a 24 bit (six digit) hexadecimal number (the "RGB" code for the color). The first two hexadecimal digits are the level of red (00=no red to FF=red fully on), the next two are the level of green and the final two the level of blue.
gray	is a real number between 0 and 1 representing the degree of "grayness" (fraction of white). 0 means black, 1 means white. Note that it's much easier to distinguish the lighter end of this (near 1) than the darker end: 0 and .25 look very similar, .90 and .95 look quite different. The default values for the first four black and white fills are solid black, solid .90 gray, solid .50 gray and solid .80 gray.
thickness	is a real-valued scale factor where 1.0 represents a standard line thickness. To make a line three times the standard thickness, use 3.0.
filled	is 0 for not filled (outline only) and 1 for filled.

Examples

The line graphs shown in Chapter 1 used a style set with these definitions:

```
LINE_BW_01=0,0.00,1.0
LINE_BW_02=0,0.60,1.0
LINE_BW_03=0,0.80,1.0
```

Here, we are redefining the styles for the first three black and white line styles. The first value after the = selects line pattern 0 (a solid line—see Pattern Definitions) for all three. The second parameter sets the gray scale value. Here, we use solid black, 60% gray, and 80% gray. (Note again that in gray scale, a higher number is lighter) The third parameter sets the line thickness—we're using the default size of 1.0.

We're only changing the black and white styles, so color versions of the graphs will use the default styles (solid lines, with black, blue, and green as the colors for the first three styles). But in black and white mode, the graphs will use solid lines in black, 60% gray, and 80% gray, respectively, rather than default of a solid line and two dashed line styles (all in black).

GRPARM(DEFINE option)

The easiest way to let RATS know about the desired style definitions is with the **GRPARM** instruction with the DEFINE option. The argument to DEFINE is a (quoted) definition string taking the form shown, one per instruction. For instance,

```
grparm(define="LINE_BW_01=0,0.00,1.0")
grparm(define="LINE_BW_02=0,0.60,1.0")
grparm(define="LINE_BW_03=0,0.80,1.0")
```

will do the three line definitions in the example above. If you have a whole set of styles that you commonly use, you can create a **SOURCE** file including these and any other graphics style options, like fixing the representation with **GRPARM(PATTERNS)** or the size with **GRPARM** with the HEIGHT and WIDTH options). For instance, the following will pull in the definitions from the file QReviewStyles.src.

```
source QReviewStyles.src
```

You can turn these off easily by just putting a comment * at the start of the line while you get the program working, then take it off to create final graphs in the desired style.

GRPARM(IMPORT option)

The older method is to create an external text file. For instance, the three line definitions from the example would be on a file with just the definitions:

```
LINE_BW_01=0,0.00,1.0
LINE_BW_02=0,0.60,1.0
LINE_BW_03=0,0.80,1.0
```

Suppose we call this file graphstyles_00_60_80.txt. Then you would use something like

```
open styles graphstyles_00_60_80.txt
grparm(import=styles)
```

This is quite a bit clumsier than the newer **GRPARM(DEFINE=..)** method, so we would recommend switching if you've been using the older method.

Background Color/Shading

The default background color for the main graph box is white. You can redefine this color or pattern to be used by using the instruction

```
GRPARM(BACKGROUND=stylenum)
```

which chooses a (color) fill style number. Similarly, the shading used by the SHADING option (on **GRAPH**) or VSHADE or HSHADE (on **SCATTER** and **GCONTOUR**), which, by default is a very light solid gray, can also be redefined. With

```
GRPARM(SHADING=stylenum)
```

you can pick a different color fill style. Finally, the grid lines used by the GRID or

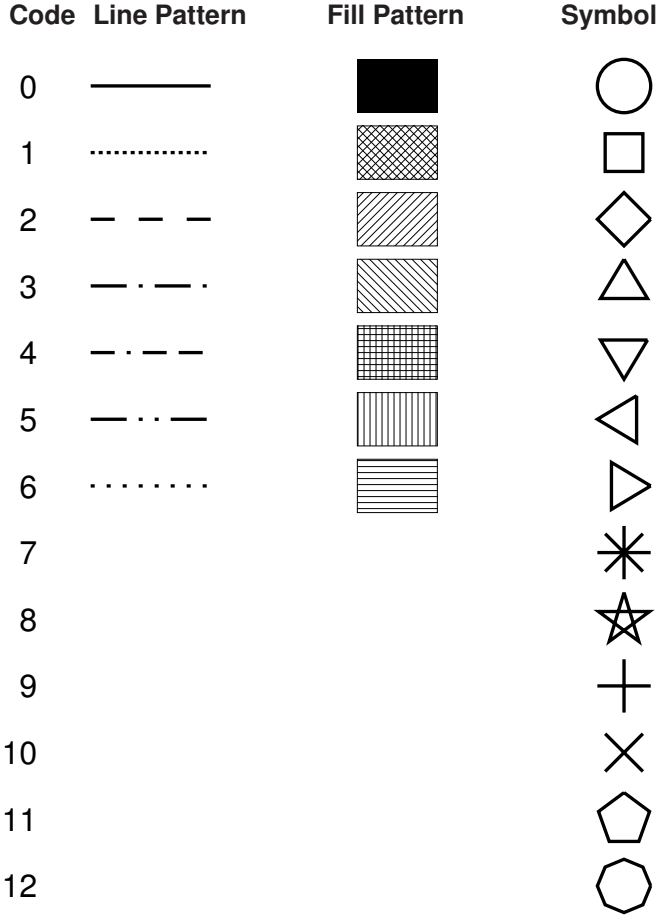
VGRID options (on **GRAPH**), VGRID or HGRID (on **SCATTER** and **GCONTOUR**) or VGRID (on GBOX) can be redefined with

GRPARM(GRID=stylenum)

The default is a "hairline" (very thin) solid black. Here the stylenum is a color line style.

Pattern Definitions

The available line, fill, and symbol choices are shown below. To select a line, fill, or symbol for a given style, use the number in the left-hand column as the *pattern* parameter. For example, "SYMBOL_COLOR_2=1,FF0000,1" defines color symbol style number two as a red, filled square (1 being the pattern code for a symbol).



3.17 Batch Graph Generation

Generating and saving graphs one at a time can be slow and error-prone if you are producing many of them, or if you have a standard program that you need to run on a regular basis. In such cases, you may prefer to have your program save your graphs automatically. There are two ways to do this, as described below.

Color Versus Grayscale

Because you aren't saving each one manually, you can't switch them to over to black and white mode if that's what you need. Instead, use the instruction

```
grparm(patterns)
```

before you generate the graphs if you want them to be drawn in black and white. (You only need to do this once). This changes the default appearance of all types of graphs from color to black and white (patterns).

GSAVE(GFORMAT=*format*) *template*

The **GSAVE** instruction is your best choice for automatically saving multiple graphs into separate files (one graph per file), and for automatically saving files in formats other than RGF, such as PostScript. You just supply a filename "template", and RATS saves any subsequent graphs using that template plus a sequence number for the filename.

The *template* will usually be a filename string that includes an asterisk (*) symbol. For each graph you generate, RATS will construct a filename by replacing the asterisk with a sequence number, and save the graph using the constructed filename. Omit the asterisk if you want to save a single graph under a specific name.

By default, this saves graphs in RGF format. You can use the FORMAT=*format* option to select other formats. Choices for FORMAT include: RGF, PORTRAIT, LANDSCAPE, PDF, LPDF, WMF, and PNG, where PORTRAIT and LANDSCAPE are PostScript format, saved in portrait and landscape orientations, respectively (see page Int–129) and LPDF is PDF saved in landscape.

The availability of some formats is platform-dependent.

The code below plots a series of impulse response graphs and saves them in a set of PostScript format files named ImpulseResp1.EPS, ImpulseResp2.EPS, and so on.

```
gsave(format=portrait) "ImpulseResp*.eps"
list ieqn = 1 to neqn
do i=1,neqn
   graph(header=header(i),key=loleft,number=0) neqn
   cards impblk(i,ieqn)
end do i
```

GSAVE(NOHEADER/NOFOOTER)

GSAVE also has NOHEADER and NOFOOTER options which can be used to suppress the (outer) header or footer on a graph being saved (either using **GSAVE** or standard single-file operations). This can be handy if the graph will be inserted into a document which will have its own captioning for the graph. You can keep the header or footer on the graph so it will still be easy to examine while you're working on it with RATS, but then strip it off when it's being saved for import into the final document.

GSAVE(PATTERNS)

GSAVE has a PATTERNS option which can be used to downgrade graphs to black and white from color, but it does that only when graphs are exported; it does not change the screen appearance the way **GRPARM(PATTERNS)** does.

OPEN PLOT *filename*

If you issue an **OPEN PLOT** instruction, any *subsequent* graphs you generate will be saved to the specified RGF format file, until you either do a **CLOSE PLOT** instruction, or use another **OPEN PLOT** instruction to open a different graph file.

This works in both interactive and batch modes, and offers the option of saving multiple graphs to a single file.

There is no problem with putting several pages of graphics on a single file if you just intend to print them. However, if you are going to reopen the graphs later for viewing or for translation to another format, you should put only one graph on a file. If your program generates several graphs, you can either use a separate **OPEN PLOT** instruction for each graph as shown below, or use the **GSAVE** instruction.

```
open plot gdp.rgf
graph(header="US Real GDP")
# gdp90
open plot consump.rgf
graph(header="Consumption of Durable Goods")
# gcd
```

Printing Graphs Automatically

If you want RATS to print graphs automatically, include the instruction

```
environment printgraphs
```

in your program (prior to the graphing instructions). This will spool the graphs to your default printer as they are generated.

3.18 Choosing Fonts

Font Handling

The FONT option on **GRPARM** (and **GRTEXT)** allows you to choose the font for the specified label(s). With the exceptions noted below, the fonts you select will be used when displaying, printing, and saving the graph. You can select from any font installed on your system (use the "Fonts" folder on the Windows Control Panel or Macintosh System folder, or a font-handling utility, to see a list of the installed fonts.)

You must type the font name exactly (although case does not matter). On a Windows system for example, you might use FONT="Times Roman" or FONT="Arial".

If you export a graph to PostScript from Windows, RATS will automatically insert dashes between words in multi-word font names, as PostScript does not accept spaces in font names. It will also substitute standard PostScript fonts for their Windows counterparts (Helvetica for Arial, for example). If you want to change or add font substitutions, and are familiar with PostScript, you can edit the PROLOG.PST file (found in your RATS directory).

You can also supply exact PostScript names. RATS may not be able to use those fonts when displaying the graph, but will use them if you export the graph to a PostScript file. Be sure to use the full PostScript name. For example, the regular version of Adobe Garamond would be "AGaramond–Regular" rather than "AGaramond".

On the Macintosh, font names are generally the same for both display and PostScript devices. This means you can use the same font name for displaying output to the screen, printing on a PostScript printer, or exporting to a PostScript file. UNIX systems are generally similar to Macintosh systems—use exact PostScript names for both display and printing purposes, as well as exporting to a file.

Examples

```
grparm(font="Symbol") axislabels 14

compute [vect[strings]] flabel=$
   ||"0","p/6","p/3","p/2","2p/3","5p/6","p"||

scatter(style=line,vlabel="Coherence",vmin=0.0,vmax=1.0,$
   xlabels=flabel)
# freq coher
```

The axis labels will be done in 14 point "Symbol" font. Although this choice affects both the horizontal and vertical axes, the numbers on the vertical axis will still be shown as numbers, since the Symbol font doesn't remap the numerical characters. However, the p's in the XLABELS option strings will be shown as π's. The seven character strings in FLABEL will be spaced equally (on center) from one end of the x-axis to the other.

4. Resources

In addition to the information and examples presented here in the RATS manuals, there are numerous other resources available that can help you get the most out of the program. We describe some of those here in this chapter.

4.1 Installing RATS (if not already done)

For UNIX/Linux Users

Please see the UNIX RATS installation guide included with your software.

For Macintosh Users

RATS is distributed as a zipped installer file. Open the zip (it may be done automatically by your browser) and double-click on the installer package to install the software. (You might have to Ctrl+click and "Open" rather than double-click---it depends upon your system settings). RATS will be installed into a `RATS xxx` folder within your `Applications` folder where xxx is a description of your specific MacRATS version. (Different releases will go into different folders).

For Windows Users

If you downloaded the software, it's in a single installer file which just needs to be double-clicked to install the software. If you need to install RATS from a CD:

1. Insert the CD in your CD ROM drive.
2. In most cases, the installation program still start automatically. If the installation program does not start automatically, you can run it manually by selecting the *Run* operation from the *Start* menu, typing in "`d:setup`" (where "`d`" is the drive containing the CD), and clicking on "OK"

File Locations

By default, the core WinRATS application files will be installed in the subdirectory:

`C:\Program Files (x86)\Estima\WinRATS xxx\`

where **xxx** is a description of your specific WinRATS version. (Different releases will go into different directories). The "user" files, including the example programs, procedures, and PDF documentation files will be a user directory. By default, this will usually be

`C:\Users\Public\Documents\Estima\WinRATS xxx\`

which you can access using the "Documents" shortcuts. Again, different releases will go into different directories.

If you want to install the files into a different locations, use the **Custom** button displayed on the **Setup Types** screen in the installer. You can control the locations of both the application files and the user files. For example, you may wish to install all of the user files in an easy-to-find location, such as

`C:\RATS`

The Custom installation button also allows you to choose which components you wish to install.

4.2 Additional Documentation

PDF Files

RATS includes a set of three manuals in Adobe PDF format. These are this *Introduction* and the *User's Guide* and *Reference Manual*. You will find these files in the "Manuals" subdirectory of your RATS user directory (see page Int–158). Windows users will have shortcuts to these in the WinRATS folder on your Start menu.

In addition, you will find the following files providing additional information:

- An *Additional Topics* manual, covering some additional statistical topics and features not included in the *Introduction*, *User's Guide* and *Reference Manual*

The Help System

The on-line help is provided as a collection of linked HTML files. You can open up the Help under Windows using the ***Help-Contents*** menu operation. From there, you can navigate to other help topics. You can also use the `F1` help key, which will open the help directly to the topic that is most likely to be useful. For instance, if you have a wizard dialog on the screen and do `F1`, it will bring up the description of that dialog.

On the Macintosh, you can use the ***Help-RATS Help*** menu operation to get the main help page, or ***Help-Search*** to locate a particular page. The contextual help for windows and dialogs can be obtained using the ***Help-Quick Help*** operation.

The help for the most current version of RATS is maintained on our web site at www.estima.com/webhelp. Note that the links to the help in the manuals all go to the web site.

The RATSletter

The *RATSLetter* is a newsletter for registered users. We distribute it (approximately) twice a year, generally by e-mail. It includes new product announcements, answers to common questions, bug reports, tips on the use of the program, lists of contributed procedures, among others. We would appreciate questions of general interest and suggestions.

Help—News...

The *News...* menu item on the *Help* menu links to major news about the software, such as updates, newsletters, important new paper replications, etc.

4.3 Examples and Procedures

RATS ships with nearly a thousand example programs. Referring to these can help you learn the program, and they can also provide frameworks you can modify to suit your own needs.

We also include hundreds of RATS procedures that greatly extend the capabilities of the program, allowing you to implement complex tasks with a single procedure call. See page Int–32 for details.

You will find the main set of examples and procedures located in your RATS user directory (page Int–158). Programs and data sets for replicating the examples from many popular econometric textbooks are provided in the `TextbookExamples` subdirectory. Similarly, the `PaperReplicationExamples` subdirectory offers program, data, and procedure files for replicating the results from many important papers.

There are lists of the examples, procedures and paper replications in the help and on the web site.

Using the Website

All of the examples and procedures shipped with RATS are also available on our website, along with dozens of others written by RATS users around the world. You may find that the "Procedure Browser" and "Search" tools on the web site are the easiest ways to locate files of interest.

You can find links to the examples and procedures by going to:

www.estima.com

and clicking on the "Resources" tab, then on "Procedures/Examples".

The "Textbook Examples" link takes you to lists of the example programs for the various textbooks, grouped by book.

The "Procedure Browser" link takes you to a very handy tool for searching through the available files (excluding the textbook examples). You can also get to the Procedure Browser directly using this address

https://estima.com/cgi-bin/procbrowser.cgi

The Browser allows you to search for files related to a particular subject area, or that reference a particular author's work or techniques. You can also search for recently added files, or limit the search to files that work with specific (older) versions of RATS.

Alternatively, you can use the "Search" tab on the site to do a general keyword search. This is particularly handy for locating relevant textbook examples (as these are not covered by the Procedure Browser) and applicable posts on our web forum.

4.4 RATS Forum and Online Courses

The RATS Software Forum

We host a discussion forum on our website, at:

www.estima.com/forum

Here, registered RATS users can exchange ideas, ask statistical and programming questions, share example code and procedures, and get answers to technical questions from Estima staff. We also frequently post new procedures and examples on the forum.

Please note that only users with legal, licensed copies of RATS are allowed to register as users on the forum. You don't have to register to view topics and posts, but registration is required to post any messages on the forum.

If you ask a technical support question on the forum, please do not email the same question to Estima's support email address, as this will lead to an unnecessary duplication of effort on our part. We are happy to provide support through either medium, but please choose one or the other.

Online Courses and Course Materials

Estima also offers online courses delivered via private sections of the user forum on our website. Please see the

www.estima.com/courses.shtml

for details on any upcoming courses.

We also offer for sale the packages of course materials from previous courses. These packages include the PDF handbooks providing the instruction materials along with example programs and procedures developed for the course. These have been very popular, as they go into much greater detail than we can provide in our main documentation on individual topics. They help you get the most out of your software. The courses are e-mailed so you can get them right away. See

https://estima.com/courses_completed.shtml

4.5 Technical Support

Support is not cheap, but we think it is an important part of what you have paid for. The next three pages describe the procedures for obtaining support and the level of support which you can expect from us.

Your Serial Number

Depending upon how you received your software, your serial number will be on the invoice, on a license agreement, or printed on a label on the CD jacket. After the software has been installed, the serial number will generally be visible at the top of the *Preferences* dialog (on the *File* menu in Windows or UNIX, or the RATS menu on the Mac) and in the "About RATS":.

Contacting Estima Technical Support

If you have access to e-mail, this is probably the best way for you to obtain technical support. You can provide us with a copy of your program and data set, and other very specific information about the nature of your question. This makes it much easier for us to resolve the problem or question, and to provide detailed answers. We try to respond to questions within one business day.

Please be sure to use a descriptive subject line on your email. A message with a subject like "Please help" is all too likely to end up in a spam-filter folder.

When you contact technical support please be able to provide the following:

1. Your name. While this seems obvious, it's remarkable how many questions come from `me@gmail.com` without any attempt to identify the sender. 100% of the time, the reply to such an email will be a request for your identity.
2. Your serial number, or, if you have unserialized software (trial or classroom), a description of where and when you got the software.
3. The product name and version number.
4. As much detail about your question as possible. In particular, if you are getting an error message you don't understand, be sure to include that error message in your e-mail. RATS produces fairly detailed error messages, so providing the specifics of any errors you encounter will make it much easier for us to diagnose the problem. Ideally, we would like the program and data file if it's about an error.

As noted earlier, please don't submit the same technical questions via both email and the forum—choose one medium or the other.

You can contact technical support at:

E–mail: support@estima.com
Voice: +1 847-864-1910

Statistical Questions

RATS has many capabilities which may be unfamiliar to some of you. If you decide to explore some new territory, we will be happy to steer you to some good references or to explain how RATS does particular computations. However, while we can help to clear up basic misunderstandings about the use of the RATS instructions, we cannot give involved statistics lessons. If you are interested in discovering new techniques, check the list of e-courses. These explain in considerable detail both the theory and implementation using RATS of many key statistical methods.

Can RATS Do ...?

Users often ask us whether or not RATS code exists for implementing a technique described in an econometrics paper that has caught their interest. In many cases, the answer is that such code does already exist. Make sure you check the main index in the *User's Guide*. Also, check the examples (main set plus textbook and paper replications). Using ***Help—Find in Files*** and looking for a keyword for the technique or an author's name is usually the quickest way to search the examples on your computer.

And we do "take requests" on a fairly regular basis by writing, or helping write, code for papers that users have asked about. We are most likely to work on papers that would be of general interest to the RATS user community. And we are *much* more likely to take up requests if the author(s) can provide both the data and the program(s) they used to produce the original results.

Having the original code (in whatever language was used) is extremely helpful, because papers often omit or are unclear about important empirical details, such as data transformations, sample ranges, and so on.

Programming mistakes are also fairly common, and having the original code makes it possible to identify discrepancies between what a paper says, and what the accompanying code actually does.

While access to the original programs is very helpful, access to the original data is usually *essential*. Without the data, it is usually impossible to know whether you have correctly reproduced the original technique, or to identify the source of any obvious differences in the results.

With rare exceptions, if the authors can't even provide the original data used in the paper, we probably won't have any interest in trying to reproduce the results (and we would question whether the paper is really of any use). Even worse is a paper with no empirical example at all. If the *author(s)* couldn't find a real-world use for it, then it isn't really worth pursuing.

Bugs and Potential Bugs

In a program as complex as RATS, there are undoubtedly some bugs remaining. In addition, because RATS has many features of a programming language, it is quite possible for you to experience problems due to errors in your own code. The more complex your program, the more likely it is that the latter is true. If your program is not running correctly, you should do the following:

- Check carefully that you are using the proper syntax on the instruction(s) causing the problem. See the *Reference Manual* or the on-line help in particular.
- If you are doing extensive operations with loops and **COMPUTE** instructions, put in some debugging statements (**DISPLAY** and **PRINT** are the most useful for this) to see where things go awry.
- Check the list of frequently asked questions, and the list of known bugs, available on our web site (www.estima.com).
- If, after all this, you have a strong suspicion that you have located a bug, contact Technical Support. If you have done a thorough job on the preceding steps, you can often ask a direct question such as "Is there a known bug in ....?", and we may be able to give you a quick answer. If you have not been able to isolate the problem, we will almost certainly have to ask you to send us the input file and data and as much other information as you can supply.

Updates

We typically have a "major" update every two to three years, where we change the main version number and redo the documentation. Between these we have several "minor" updates, where we add features to the program, revise procedures, add new textbooks and paper replications, etc. Many of the most important features get added first in these minor updates since that's when we do most of the programming. If you want to stay up-to-date automatically, we have update subscription programs where you pay in advance to receive the updates at one fixed rate. See

https://estima.com/ratsupdatesubs.shtml

Bibliography

Diebold, F.X. (2004). *Elements of Forecasting, 3rd Edition*. Cincinnati: South-Western

Greene, W.H. (2012). *Econometric Analysis, 7th Edition*. New Jersey: Prentice Hall.

Hill, R.C., W.E. Griffiths, and G.C. Lim (2008). *Principles of Econometrics, 3rd Edition*. New York: Wiley.

Makridakis, S., S.C. Wheelwright, and R.J. Hyndman (1998). Forecasting, Methods and Applications, 3rd Edition. Hoboken: Wiley.

Pindyck, R. and D. Rubinfeld (1998). *Econometric Models and Economic Forecasts*, *4th Edition*. New York: McGraw-Hill.

Verbeek, M. (2008). *A Guide to Modern Econometrics, 3rd Edition*. New York: Wiley.

Index

Symbols

A

B

C

D

Q

R

S

T

U

V

W